Tiedostojen sähköinen allekirjoitus ohjelmistokehityksessä

Rossi, Petteri

2018 Laurea

Laurea-ammattikorkeakoulu

Tiedostojen sähköinen allekirjoitus ohjelmistokehityksessä

Rossi, Petteri
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Joulukuu, 20182018

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelmaTietojenkäsittelyn koulutusohjelma
IT-Tradenomi

Rossi, Petteri Petteri Rossi

Tiedostojen Sähköinen Allekirjoitus OhjelmistokehityksessäTiedostojen sähköinen allekirjoitus ohjelmistokehityksessä

Vuosi 20182018 Sivumäärä 36

Tämän opinnäytetyön toimeksiantajan työntekijät ovat saaneet luoda Visual Basic for Applications (VBA) makroja toimeksiantajan ympäristössä vapaasti. Toimeksiantaja pitää tätä nyt uhkana, sillä VBA-makrojen seassa saattaa olla viruksia, kuten Ransomware -ohjelmia. Toimeksiantaja haluaa ottaa käyttöön sähköisen allekirjoituksen ratkaisuna tähän uhkaan, mutta sen käyttöönotossa ilmenee ongelmia. Etenkin allekirjoitettavien tiedostojen määrä on ongelma, joka tekee tiedostojen yksittäisestä allekirjoituksesta vaikeaa.

SignServer on ohjelma, joka automatisoi sähköisen allekirjoituksen, joten toimeksiantaja haluaa ottaa SignServerin käyttöön heidän ympäristöön. Toimeksiantaja ei ole kuitenkaan varma, sopiiko SignServer heidän ympäristöönsä. Tämä opinnäytetyö tutkii, sopiiko SignServer toimeksiantajan ympäristöön ja täyttääkö se toimeksiantajan muut vaatimukset sille.

Opinnäytetyön tavoite oli rakentaa SignServerille hiekkalaatikkoympäristöön palvelin ja suorittaa SignServerille testejä. Jos SignServer läpäisee nämä testit hyväksyttävästi, niin toimeksiantaja sijoittaa enemmän resursseja SignServerin käyttöottoon. Toimeksiantaja haluaa myös selvityksen muista SignServerin toiminnallisuuksista, joista voisi olla heille hyötyä.

SignServerin arkkitehtuurin mallinnus pohjautui Microsoftin Code Signing Best Practices -viitekehykseen. SignServer-palvelin rakennettiin seuraamalla LAMP-mallia, joka soveltuu web-sovellusten palvelimien rakentamiseen mainiosti. Tämän työn tuotokset kehitettiin seuraamalla inkrementaalista mallia. SignServerille suoritettujen testien vaatimukset asetti toimeksiantaja.

SignServer läpäisi vaaditut testit hyväksyttävästi. SignServer suorittaa sähköisen allekirjoituksen onnistuneesti, ja sen arkisto ja loki toimivat hyvin. Yhtä testiä ei voitu suorittaa, koska SignServer-palvelin oli hiekkalaatikkoympäristössä. Toimeksiantaja ottaa SignServerin käyttöön tulevaisuudessa. Todettiin myös, että SignServer voi parantaa toimeksiantajan jo olemassa olevaa käännösautomaatiota. Tärkein kehitysehdotus on luoda SignServerille palvelin ympäristöön, josta se pääsee Internettiin. Tällöin kaikki SignServerin ominaisuudet voidaan testata ja SignServerin integraatio toimeksiantajan välityspalvelimeen voidaan aloittaa.

Asiasanat: Sähköinen allekirjoitus, Sovelluskehitysympäristö, PKI, Inkrementaalinen kehitys malli, Tietoturva

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's thesis

Rossi, Petteri Petteri Rossi

Digital Signing of Files in Software Development

Year 20182018 Pages 36

The employees of the commissioner of this thesis have been allowed to create Visual Basic for Applications (VBA) macros freely in the commissioner's environment. Now the commissioner sees this as a threat, as there may be viruses among the VBA-macros like Ransomware. The commissioner wants to introduce digital signatures as a solution to the threat, but there are issues with its implementation. Notably the number of files to be signed is an issue that makes it difficult to sign files one by one.

SignServer is a program that automates digital signings, thus the commissioner wants to implement SignServer in their environment. However, the commissioner is not certain whether SignServer is suitable for their environment. This thesis examines whether SignServer is suitable for the commissioner's environment and if it fulfils the commissioner's other requirements.

The aim of this thesis was to build a server for SignServer in a sandbox environment and run tests for SignServer. If SignServer passes the tests successfully, the commissioner will invest more resources on the implementation of SignServer. The commissioner also wants a report about any other functions of SignServer, which could be useful to them.

The architectural modelling of SignServer was based on the Microsoft Code Signing Best Practices document. The server for SignServer was built by following the LAMP-model, which is well suited for building servers for web applications. The results of this thesis were developed by following the incremental model. The requirements of the tests for SignServer were set by the commissioner.

SignServer passed the required tests successfully. SignServer can perform a digital signature successfully, and its archive and log work well. One test could not be performed because the server for SignServer was in a sandbox environment. The commissioner will implement SignServer in the future. It was also discovered that SignServer can enhance the commissioner's existing build servers. The most important suggestion for further development is to create a server for SignServer in an environment where it can access the Internet. Then all the features of SignServer can be tested, and the integration to the commissioner's proxy server can be started.

Keywords: Digital signatures, Software development environment, PKI, Incremental development model, Information security

Sisällys

Käsitteet

Active Directory	Windows-toimialueen käyttäjätietokanta, joka sisältää tietoa käyttäjistä ja heidän käyttöoikeutensa
Allekirjoitusprosessi	SignServerin sisäinen prosessi, joka suorittaa sähköisen allekirjoituksen
Authenticode	Sähköinen allekirjoitusmetodi, jolla allekirjoitetaan Microsoft PE tiedostoja
Client-yhteys	Tietokoneiden välisessä kommunikaatiossa client on asiakas, joka pyytää palvelimelta jotakin
DevOPS	Yhdistelmä sanoista ”development” ja ”operations”, DevOPS on toimintamalli, joka parantaa ohjelmistokehitystä yhdistämällä sovelluskehityksen ja ylläpidon
Hash-arvo	Tiedostosta kryptografisella tiivistefunktiolla yksisuuntaisesti laskettu tiiviste, jota käytetään muun muassa tiedoston todentamiseen. Sha256 on esimerkki tiivistefunktiosta
Käännösautomaatio	Sovelluskehityksessä käytetty palvelin, jossa ohjelmisto rakennetaan ja ajetaan
Makro	Ohjelma, joka suorittaa tehtäviä käyttäjän puolesta
SignServer	PrimeKeyn kehittämä ohjelma, joka suorittaa sähköisen allekirjoituksen automaattisesti
Varmenne	Sähköinen dokumentti, joka osoittaa sen omistajan omaavan julkisen avaimen
Varmenteiden haltija	PKI-rooli, ulkoinen yleisesti luotettu taho, johon tiedostojen julkaisijat ja käyttäjät voivat luottaa
Välityspalvelin	Toisin tunnettuna Reverse proxy, ympäristön sisäinen välittäjä, joka tarkistaa ympäristön tietoliikenteen, Käytetään muun muassa virustentorjunnassa ja todennuksessa

Lyhenteet

CA	Certificate Authority, varmenteiden haltija
CIA	Confidentiality, Integrity, Availability
CMS	Cryptographic Message Syntax
CN	Common Name
CRL	Certificate Revocation List

DB	Database, tietokanta
EJBCA	Enterprise Java Bean Certificate Authority
Exe	Executable, suoritusohjelma
HSM	Hardware Security Module
LAMP	Linux, Apache, MySQL, PHP
Microsoft PE	Microsoft Portable Executable
Msi	Windows Installer
PDF	Portable Document Format
PKI	Public Key Infrastructure, julkisten avainten infrastruktuuri
RHEL	Red Hat Enterprise Linux
SSL	Secure Sockets Layer
SQL	Structured Query Language
TSA	TimeStamp Authority
UI	User Interface
VBA	Visual Basic for Applications
VM	Virtual Machine
XML	Extensible Markup Language

1 Johdanto

Tämä opinnäytetyö tehtiin yhteistyössä Suomessa toimivan finanssialan toimijan kanssa. Toimeksiantaja ei halua julkaista nimeään tähän opinnäytetyöhön, joten nimeä ei mainita.

Toimeksiantaja käyttää ympäristössään Visual Basic for Applications (VBA) makroja. VBA on ohjelmointikieli Microsoft Office -tuotteille. VBA:lla voi luoda ja automatisoida monia hyödyllisiä ja vaativia Office -tuotteiden toimintoja, joita ei voi suorittaa Office -tuotteiden perustoiminnoilla. (Microsoft 2018a.) Toimeksiantajan ympäristössä VBA-makroja käytetään eniten Microsoft Excel -ohjelman kanssa, mutta kaikenlaisia VBA-makroja esiintyy.

Toimeksiantajan työntekijät ovat saaneet luoda vapaasti VBA-makroja. Tämä on tietoturvallisesti ongelmallista. Tietoturvan tavoitteet voidaan jakaa kolmeen eri periaatteeseen. Ne ovat luottamuksellisuus, eheys ja saatavuus, ja niiden yhteisnimitys on CIA-kolmio. Toimeksiantajan politiikka rikkoo eheyden tavoitetta. Eheyden periaate on, että tietoa lisäävät, muokkaavat tai päivittävät vain luotetut henkilöt (Vacca 2014, 48 – 49). Eheys varmistaa, että tietojärjestelmässä olevaan tietoon voidaan luottaa. Koska VBA-makroja on voitu luoda vapaasti, niiden seassa voi olla haittaohjelmia. Haittaohjelmia on monenlaisia, muun muassa Ransomware -haittaohjelmat lukitsevat tietokoneen täysin, jonka jälkeen haittaohjelman luoja kiristää uhrilta rahaa (Microsoft 2018b). VBA-makrot ovat siis toimeksiantajalle hyödyllisiä, mutta myös uhkia.

Toimeksiantaja haluaa siis vähentää heidän ympäristöönsä kohdistuvia uhkia. Toteuttaakseen tämän, toimeksiantaja on päättänyt ottaa käyttöön sähköisen allekirjoituksen. Sähköinen allekirjoitus varmistaa, että tiedostoa ei ole muutettu allekirjoituksen jälkeen, ja se sitoo tiedoston sen allekirjoittajaan (Microsoft 2007, 5). Tällöin toimeksiantaja voi sähköisesti allekirjoittaa VBA-makrot, joihin toimeksiantaja luottaa, ja kieltää kaikki tuntemattomat VBA-makrot. Tällöin haittaohjelmat eivät voi esiintyä VBA-makroina.

Manuaalisen sähköisen allekirjoituksen käyttöönotossa ilmenee kuitenkin kaksi ongelmaa. Sähköisen allekirjoituksen suoritus ei ole nopea prosessi. Allekirjoituksia voi normaalisti tehdä vain yksitellen ja jokainen allekirjoitus pitää varmistaa erikseen. Tämän lisäksi toimeksiantajan ympäristössä on käytössä erittäin suuri määrä VBA-makroja. Jos sähköinen allekirjoitus halutaan ottaa käyttöön, niin nämä kaikki pitää allekirjoittaa. Ongelma on, että kaikista makroista ei ole pidetty kirjaa. Jos toimeksiantaja haluaisi ottaa käyttöön sähköisen allekirjoituksen, niin kaikkien VBA-makrojen etsiminen kestäisi vuosia.

1.1 Tutkimusongelma

Toimeksiantaja tutki monia ratkaisuja sähköisen allekirjoituksen käyttöönottoon ja lopulta päätti tutkia tarkemmin ohjelmaa nimeltä SignServer. SignServer on ruotsalaisen yhtiön PrimeKeyn (PrimeKey 2016) tuottama ohjelma, joka voi suorittaa sähköisen allekirjoituksen

automaattisesti. Tiedosto pitää vain syöttää SignServerille, ja se suorittaa lopun allekirjoitusprosessista. Tämän lisäksi allekirjoitusoikeus voidaan jakaa SignServerin avulla valittujen työntekijöiden kesken, jotta toimeksiantajan työntekijät voivat itse allekirjoittaa tarvittavat VBA-makrot. SignServer siis korjaa edellisessä luvussa kuvatut ongelmat sähköisen allekirjoituksen käyttöönotossa.

SignServerissä on kuitenkin ongelma. SignServerille pitää luoda eri allekirjoitusprosessi erilaisille tiedostotyypeille. PDF-allekirjoittaja ei voi sähköisesti allekirjoittaa XML-tiedostoja ja toisinpäin. Jokainen allekirjoitusprosessi pitää kehittää erikseen, ja SignServerissä ei ole VBA-allekirjoittajaa tällä hetkellä. PrimeKey voi kehittää VBA-allekirjoittajan, mutta siitä syntyy toimeksiantajalle kustannuksia. Toimeksiantaja haluaa siis selvityksen siitä, soveltuuko SignServer heidän ympäristössä ja soveltuuko se VBA-makrojen allekirjoittamiseen.

1.2 Työn tavoite

Tämän opinnäytetyön tavoite on saada selville, soveltuuko SignServer VBA-makrojen sähköiseen allekirjoitukseen. Laveasti sanottuna tämä tarkoittaa, että toimeksiantaja haluaa selvityksen siitä, ovatko SignServerin tarjoamat palvelut, muun muassa arkisto ja loki, yhteensopivia toimeksiantajan ympäristön kanssa, ja voiko se suorittaa vaativia sähköisiä allekirjoituksia. Tämän lisäksi toimeksiantaja haluaa saada selville, onko SignServerissä muita toimintoja tai ominaisuuksia, joista voisi olla toimeksiantajalle hyötyä.

SignServerille pitää suunnitella toimiva palvelin. Palvelimen suunnitelmapohja tulee olemaan LAMP-malli, joka on palvelimien suunnittelussa käytetty viitekehys (Semanticscholar 2017). SignServer pitää myös mallintaa toimeksiantajan ympäristöön. Mallinnuksen teoreettinen viitekehys on Microsoftin Code Signing Best Practices -dokumentti (Microsoft 2007).

Tämän jälkeen SignServerille suoritetaan eri testejä. Niiden tuloksia verrataan toimeksiantajan vaatimuksiin SignServerille. Jos SignServer suorittaa sille asetetut vaatimukset hyväksyttävästi, niin toimeksiantaja ottaa SignServerin käyttöön, sijoittaa resursseja omaan sovellukseen, joka toimii käyttöliittymänä, ja ostaa VBA-makrojen allekirjoittajan PrimeKeylta.

Toimeksiantaja haluaa varmistuksen siitä, että SignServer voi sähköisesti allekirjoittaa Microsoft Portable Executable (PE) tiedostoja. Lisäksi toimeksiantaja haluaa SignServerin lokin, arkiston ja tiedostojen aikaleimauksen toimivan hyväksyttävästi. Toimeksiantaja haluaa myös, että SignServerin palvelimelle suoritettaisiin kovennusta testauksen jälkeen.

2 Yrityskumppani

Tämän työn aikana käsiteltiin herkkäluonteisia yrityssalaisuuksia. Etenkin huomioitavaa ovat tiettyjen laitteiden nimet, ja ohjelmien julkaisunumerot. Jos näistä tiedoista tulisi julkisia,

tietoturvahyökkäysten suunnitteleminen toimeksiantajaa vastaan olisi huomattavasti helpompaa. Täten siis monet toimeksiantajan ympäristöön liittyvät nimet on jätetty mainitsematta. Toimeksiantajan ympäristöstä kuvaillaan joitakin osia, mutta suurin osa kriittisestä infrastruktuurista on salattu, tai jätetty mainitsematta.

Toimeksiantajalla oli töissä kaksi henkilöä, jotka olivat selvittämässä SignServerin soveltuvuutta toimeksiantajan ympäristöön, kun tämän opinnäytetyön kirjoittaminen alkoi. Tämä opinnäytetyö tehtiin yhteistyössä heidän kanssaan, ja he toimivat ohjaajina opinnäytetyölle.

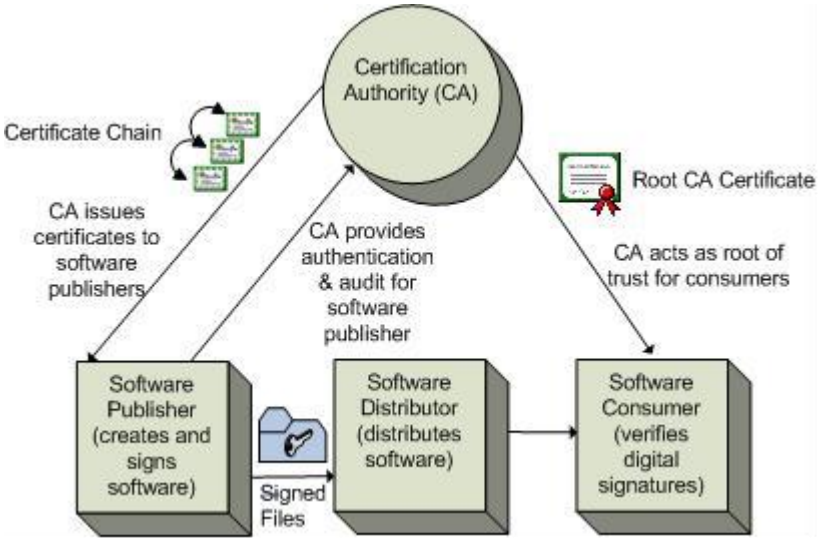
3 Tietoturva teoriaa

3.1 Sähköinen allekirjoitus

Sähköinen allekirjoitus on yksi PKI:n käytöistä. Sähköinen allekirjoitus on tiedostoon liitetty data, joka sitoo tiedoston sen allekirjoittajaan ja varmistaa, että tiedostoa ei ole muutettu allekirjoituksen jälkeen. Allekirjoituksen suorittamiseen tarvitaan yksityinen ja julkinen avain, joista julkinen avain on varmenne. Tiedostosta lasketaan hash-arvo, joka allekirjoitetaan yksityisellä avaimella ja liitetään tiedostoon. Sähköinen allekirjoitus on nyt suoritettu. Allekirjoituksen voi todentaa sen allekirjoittajan julkisella avaimella. Julkinen avain laskee tiedoston hash-arvon, ja vertaa sitä tiedostoon liitettyyn hash-arvoon. Jos hash-arvo on sama, tiedosto on todennettu. (Microsoft 2007, 5 – 6.)

3.2 Sähköisen allekirjoituksen ekosysteemi

Sähköisen allekirjoituksen ekosysteemiin kuuluu kaikki roolit, jotka tarvitaan sähköisen allekirjoituksen käyttöönottoon. Kuviossa 1 esiintyy kaikki roolit.



Kuvio 1: Sähköisen allekirjoituksen ekosysteemi (Microsoft 2007)

Varmenteiden haltija (CA) myöntää varmenteita muille rooleille ekosysteemissä ja omistaa juurivarmenteen, johon kaikki muut varmenteet luottavat. Ohjelman julkaisija luo ohjelmia ja sähköisesti allekirjoittaa tiedostoja CA:n myöntämällä varmenteella. Ohjelman jakelija jakaa tai myy ohjelmia ohjelmien kuluttajille. Ohjelman kuluttajat käyttävät ohjelmia ja tilaavat varmenteiden haltijalta julkisen varmenteen, jolla kuluttaja voi todentaa ohjelman sähköisen allekirjoituksen. (Microsoft 2007.)

Tämä ekosysteemi on PKI:n toteutus. PKI on lyhenne sanasta Public Key Infrastructure ja on suomennettuna julkisten avainten infrastruktuuri. PKI-asiantuntija Martin Furuhed (2018) selittää PKI:n oleelliset osat yksinkertaisesti. PKI koostuu käytännöistä, standardeista, laitteista ja ohjelmista, jotka hallitsevat varmenteiden luomista, jakelua, peruuttamista ja hallintaa. PKI:n tarkoitus on sitoa varmenne ja toinen taho toisiinsa. Varmenne on tiedosto, joka pitää sisällään tunnistustietoja, sarjanumeron ja vanhentumispäivän. Varmenteita käytetään moniin tarkoituksiin, muun muassa henkilöllisyyden todentamiseen, tiedon salaukseen ja sähköisten allekirjoitusten suorittamiseen.

3.3 Aikaleimaus

Sähköisen allekirjoituksen yhteydessä tiedoston allekirjoituksen voi aikaleimata. Allekirjoitusvarmenteiden ongelma on, että jos varmenne vanhenee, niin varmenteeseen ei enää luoteta. Tällöin ei myöskään varmenteella allekirjoitettuihin tiedostoihin ei enää luoteta. Tiedostot pitää allekirjoittaa uudestaan. Aikaleimaus korjaa tämän. Aikaleimaus osoittaa, milloin allekirjoitus on tehty. Täten voidaan todentaa, että allekirjoitusvarmenne on ollut voimassa allekirjoitushetkellä ja allekirjoitukseen voi luottaa varmenteen vanhenemisen jälkeen. (Lynch 2018.)

4 Menetelmät ja tutkimusprosessi

4.1 Ketterät menetelmät

Ketterät menetelmät, toisin tunnettuna Agile, ovat joukko ohjelmistokehityksessä ja IT-projektien läpiviemisessä käytettyjä menetelmiä. Ketterien menetelmien julistus (Agile Manifesto 2001) kuvailee ketterien menetelmien ihanteet. Ne ovat: toimiva ohjelmisto on tärkeämpi kuin kattava dokumentaatio, yksilöt ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut, yhteistyö asiakkaan kanssa yli sopimusneuvottelujen ja muutokseen vastaaminen on tärkeämpää kuin pitäytyminen suunnitelmaan. Ketteriä menetelmiä on suuri määrä.

Tämän työn viikoittainen työrutiini esiintyy kuviossa 2:

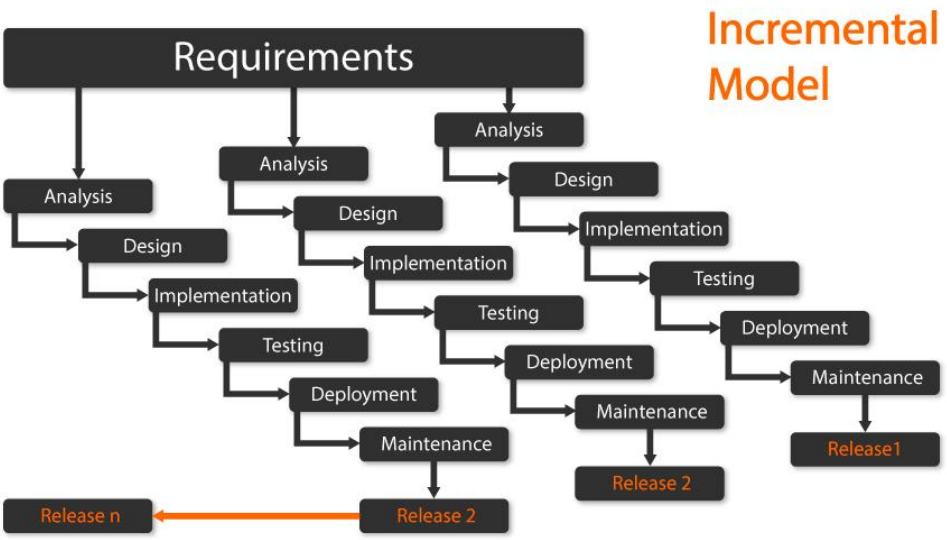
Kuvio 2: Viikoittainen rutiini

Viikon alussa pidettiin tapaaminen ohjaajien kanssa. Heille esiteltiin edellisen viikon löydöt, selvitykset ja ongelmat, ja mitä seuraavan viikon aikana pitäisi saada tehtyä. Työt saatiin usein tehtyä ennen viikonloppua, ja ylijäänyt aika käytettiin dokumentaatioon ja opinnäytetyöhön.

4.2 Kehittämismenetelmä

Tämän opinnäytetyön kehittämismenetelmä on inkrementaalinen malli. Se on ketterä menetelmä, joka painottaa koko järjestelmän osittaista toteutusta ja toiminnallisuuden nopeaa, syklittäistä kehittämistä. Inkrementaalinen malli kiinnittää ensisijaisesti huomiota järjestelmän vaatimuksiin ja vasta sen jälkeen järjestelmän toteutukseen. Jos järjestelmän toteutuksen aikana ilmenee virheitä, tai järjestelmää halutaan kehittää, niin inkrementaalisen mallin avulla voidaan palata ripeästi vaatimuksien määrittelyyn. Tällöin inkrementaalinen malli vähentää kustannuksia, jotka voisivat esiintyä epäonnistuneen toteutuksen aikana, mutta silti reagoi ripeästi muutoksiin. (New Line Technologies 2018.)

Kuvio 3 on esimerkki inkrementaalisen mallin rungosta.



Kuvio 3: Inkrementaalinen toteutusmalli (New Line Technologies 2018)

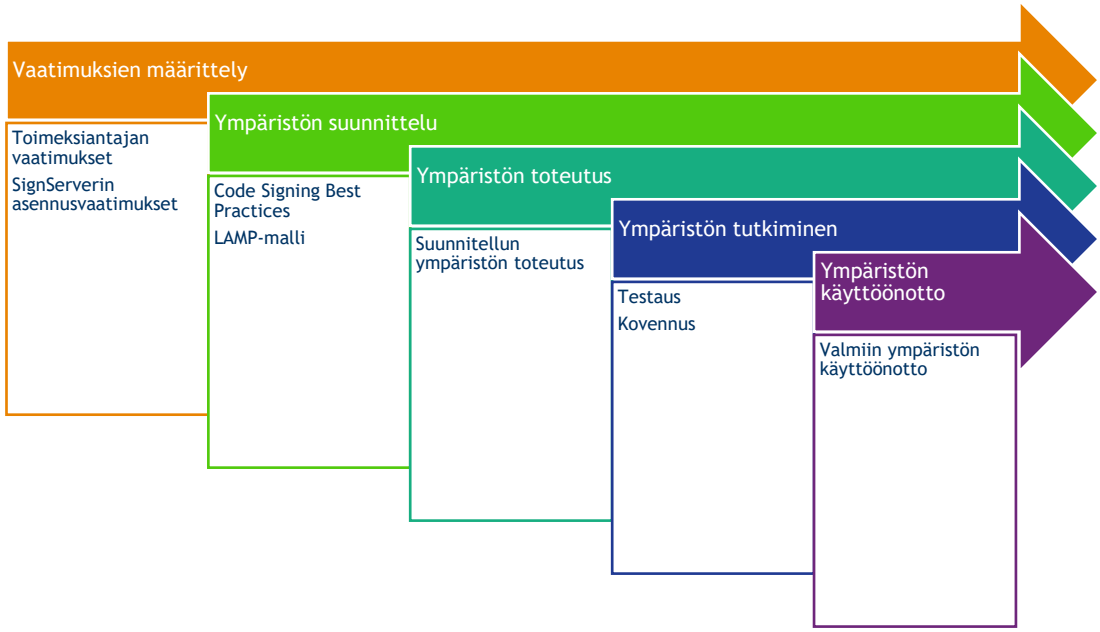
Tämä opinnäytetyö ei vaadi kaikkia inkrementaalisen mallin vaiheita, jotta se voidaan toteuttaa. Kehitysvaiheet tälle opinnäytetyölle ovat vaatimuksien määrittely, ympäristön suunnittelu, ympäristön käyttöönotto ja testaus.

Inkrementaalinen malli sopii tälle työlle hyvin. SignServerin palvelin pitää suunnitella, rakentaa ja testata. Nämä vaiheet ovat luonnostaan osa inkrementaalista mallia. Inkrementaalinen malli mahdollistaa ripeän reaktion mahdollisiin virheisiin, jotka voivat ilmetä toteutuksen aikana. Tämän lisäksi toimeksiantajalle syntyvä riski on pieni, sillä he eivät ole sijoittaneet resursseja toimintoihin, joista he ovat epävarmoja. (New Line Technologies 2018.)

Tämä opinnäytetyö ei ole toimeksiantajan ensimmäinen eikä viimeinen kehityssykli SignServerille. Tätä opinnäytetyötä edellytti Proof of Concept -testi. Toimeksiantaja halusi tietää, onko SignServer yhteensopiva heidän avainhallintamenetelmänsä kanssa. Tämän lisäksi jos SignServer todetaan soveltuvan toimeksiantajalle, niin toimeksiantajan pitää kehittää käyttöliittymä SignServerille ja valmistella SignServerille palvelin, joka toimii tuotantoympäristössä. Kumpikin näistä vaatii oman kehityssyklin.

4.3 Tutkimusprosessi

Kuviossa 4 ilmenee tämän opinnäytetyön tutkimusprosessi.

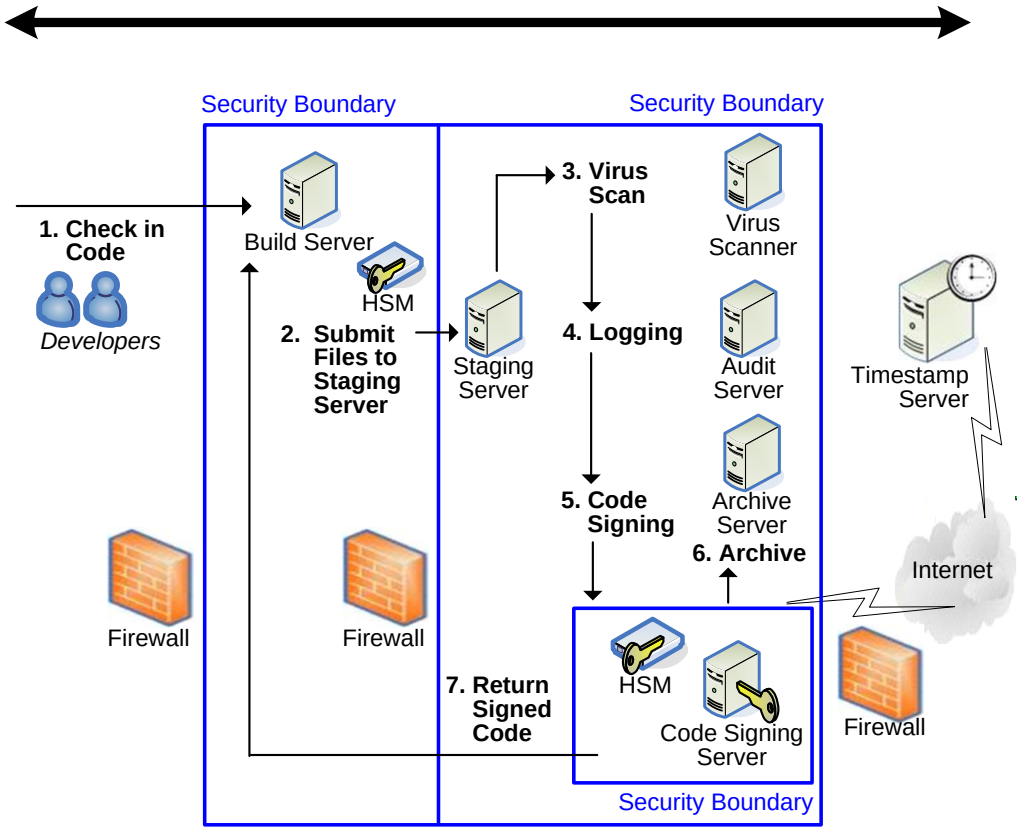


Kuvio 4: Tutkimuksen tutkimusprosessi

Tämä rakenne mahdollistaa SignServerin palvelimen suunnittelun, rakentamisen ja testaamisen. SignServeriä ei ole vielä otettu käyttöön tuotantoympäristössä, käyttöönotto tapahtuu tulevissa kehityssykleissä.

5 SignServer palvelimen arkkitehtuurin mallinnus

Toimeksiantaja valitsi palvelimen arkkitehtuurin teoreettiseksi malliksi Microsoftin Code Signing Best Practices -dokumentin (Microsoft 2007). Se on yksi viitatuimmista saatavilla olevista koodin allekirjoittamisen viitekehyksistä. Dokumentti on yli 10 vuotta vanha, mutta siihen vieläkin viittaavat muun muassa organisaatiot CA Security Council (CA Security Council 2016), Digicert (Digicert) ja asiantuntijat kuten Larry Seltzer (Seltzer 2013). Se ei pelkästään käsittele koodin allekirjoituksen teoriaa, vaan myös esittelee eri malleja ja käyttötapauksia koodin allekirjoituksesta. Se on myös yksi harvoista dokumenteista, joka käsittelee koodin automaattista allekirjoitusta. Kuviossa 5 on Microsoftin malli (Microsoft 2007, 48) automaattisen allekirjoituksen topologialle.



Kuvio 5: Microsoftin malli automaattiselle sähköiselle allekirjoittamiselle (Microsoft 2007, 48)

Malli on yksinkertainen, mutta käsittelee kaikki osiot arkkitehtuurista. SignServer-palvelimen arkkitehtuuri muistuttaa kuvaa melko tarkasti, mutta se poikkeaa mallista tietyin tavoin.

Malli (Microsoft 2007, 48) olettaa, että kaikki allekirjoittajat ovat ohjelmistokehittäjiä. Se olettaa myös, että allekirjoitusprosessissa käytetään käännösautomaatiota. Tämä malli soveltuu siis parhaiten koodikehittäjille. Tämä ei kuitenkaan kelpaa toimeksiantajan suunnitelmiin täysin. Allekirjoitettavia VBA-makroja on jo olemassa suuri määrä, ja niistä ei ole pidetty tarkkaa lukua. Toimeksiantajan pitää siis kehittää omalle mallilleen menetelmä, jolla VBA-makrot saadaan sähköisesti allekirjoitettua.

Todennuksesta malli ei mainitse mitään. Malli vaikuttaa olettavan kaikilla, joilla on oikeus syöttää tiedostoja käännösautomaatioon, olevan myös oikeudet allekirjoittaa ohjelmia. Malli ei myöskään käsittele oikeuksia ympäristöön. Samalla kehittäjällä on oikeus allekirjoittaa Microsoft PE -tiedostoja ja Java-ohjelmia. Tämä voi aiheuttaa ongelmia. Toimeksiantajan piti siis kehittää joko oma menetelmä todennukseen tai jäsentää jo käytettyjä menetelmiä SignServerin todennukseen.

Tietoturvallisuusmenetelmistä mainitaan paljon. Malli ehdottaa (Microsoft 2007, 48 - 50) selviä rajoja allekirjoitusverkkoon, ja palomuurien käyttöä rajoilla. Allekirjoitettava tiedosto

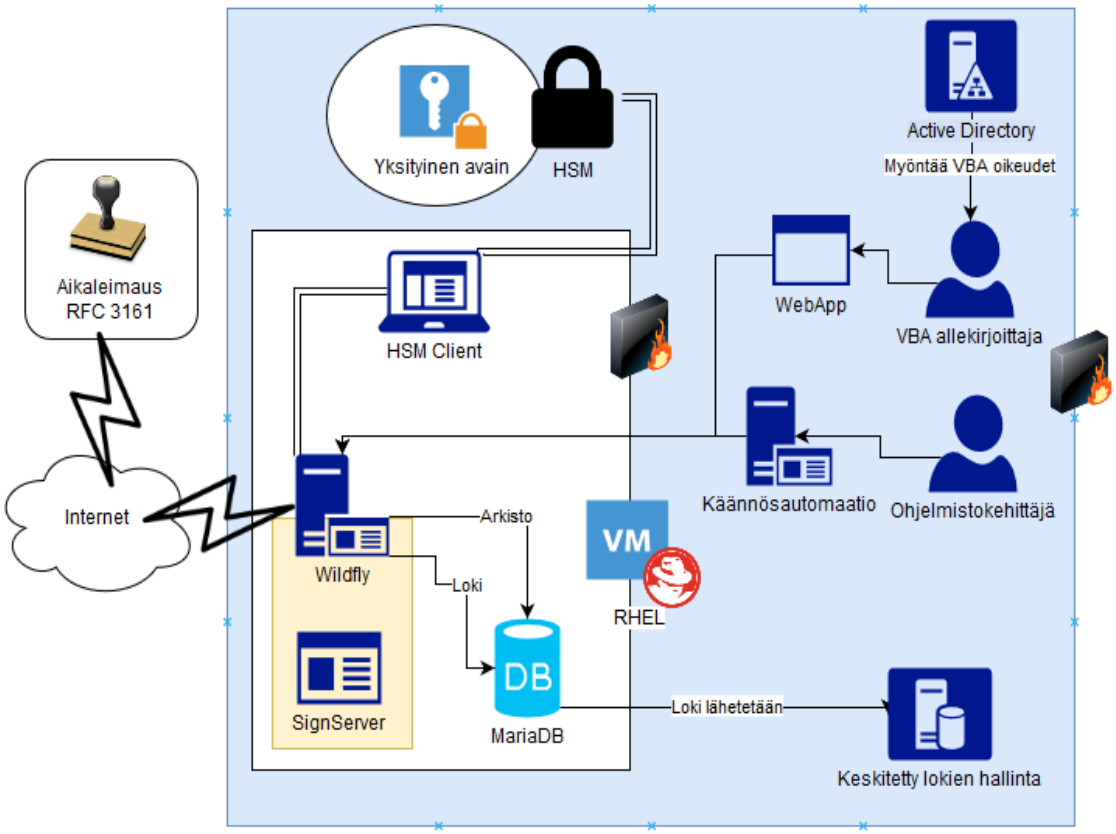
kokee virustarkistuksen ainakin kahdelta virustorjuntaohjelmalta, tapahtuma kirjataan lokiin, tiedosto arkistoidaan ja aikaleimauspalvelu on toisessa verkossa. Kaiken kaikkiaan malli käsittelee tietoturvallisuutta hyvin. Ainoa merkittävä puute mallissa on allekirjoittajan todennuksen puute.

Microsoftin mallissa Hardware Security Module (HSM) ja allekirjoittava palvelin ovat saman rajan sisällä, tai ainakin loogisesti lähellä toisiaan. Tämä ei ole välttämättä parhain päätös. HSM on erittäin turvallinen laite, mutta on aina hyvä idea pitää HSM omassa alueessaan. Toimeksiantaja on myös ostanut HSM-palvelun käyttöpalvelutoimittajalta, joten HSM ja SignServer eivät voi sijaita samassa alueessa.

Malli ei myöskään ehdota mitään ohjelmia, jotka voisivat täyttää nämä vaatimukset. Microsoft ei tarjoa omia työkaluja, jotka voisivat suorittaa automaattisen allekirjoituksen, eikä Microsoft halua mainostaa toisten yhtiöiden tuotteita. Toisaalta tämä on ymmärrettävää, sillä suositellut ohjelmat voivat muuttua tietotekniikan maailmassa nopeasti. Suositukset, jotka tehdään vuona 2007, eivät välttämättä ole hyödyllisiä vuonna 2018.

5.1 Toteutus

Toimeksiantajan malli sähköiselle allekirjoitukselle on jaettu kahteen kuvaan, SignServerin topologiaan ja allekirjoituksen malliin. Kuviossa 6 on toimeksiantajan malli automaattiselle sähköiselle allekirjoittamiselle:



Kuvio 6: Toimeksiantajan malli automaattiselle sähköiselle allekirjoittamiselle

Microsoftin malli oletti, että kaikki allekirjoittajat ovat ohjelmistokehittäjiä. Toimeksiantaja on soveltanut tämän omaan malliinsa. On oletettavaa, että toimeksiantaja käyttää SignServeriä allekirjoittamaan kehitteillä olevat ohjelmat tulevaisuudessa. Tämä menetelmä ei kuitenkaan sovellu VBA-makrojen allekirjoittamiseen.

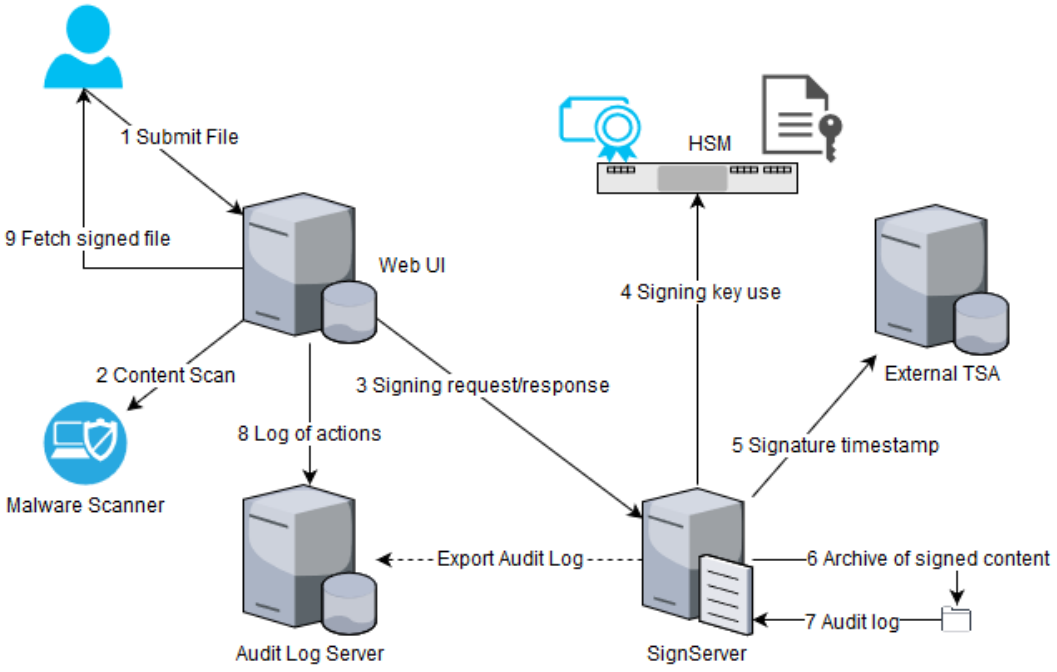
Toimeksiantajan pitää siis kehittää oma sovellus, jolla voi lähettää VBA-makroja SignServerille sähköisesti allekirjoitettavaksi. VBA-makrojen suuri määrä on toinen ongelma, jolle löydettiin ratkaisu. Toimeksiantaja aikoo jakaa käyttöoikeuksia sovellukseen valikoiduille työntekijöille. Kuka tahansa toimeksiantajan työntekijä voi hakea oikeudet allekirjoittaa VBA-makroja. Käyttöoikeudet pitää hakea erikseen toimeksiantajan käyttöoikeuspalvelusta, ja esimiehen pitää hyväksyä pyyntö. Tällöin käyttöoikeudet allekirjoittamiseen myönnetään vain heille, joilla on tarve allekirjoittaa VBA-makroja.

Microsoftin malli ei maininnut mitään todennuksesta. Toimeksiantaja päätti käyttää jo olemassa olevaa välityspalvelinta valvomaan SignServerin todennusta. Välityspalvelin toimii tarkastamalla toimeksiantajan Active Directorysta, että käyttäjällä on oikeudet sähköisesti allekirjoittaa tiedostoja. Jos käyttäjällä on oikeudet, ohjelma jatkaa allekirjoitusprosessin seuraavaan vaiheeseen. Muulloin pyyntö hylätään.

HSM ja allekirjoittava palvelin olivat saman loogisesti rajatun alueen sisällä Microsoftin mallissa. Toimeksiantajan mallissa HSM ei ole fyysisesti tai loogisesti yhteydessä SignServeriin, vaan yhteys HSM:ään hoidetaan paikallisella HSM client -yhteydellä. Clientilla on oikeus selailla HSM:n avaimia, mutta se ei voi käyttää niitä ilman todennusta. SignServerin palvelimelta ei voi muokata HSM:n tiedostoja millään tavalla. Palvelimelta voi esimerkiksi luoda avaimia, mutta kaikki toiminnot tarvitsevat todennuksen, ja että fyysisessä HSM:ssä on turvallisuusmerkki kiinni.

Microsoftin malli ei mainitse mitään pilvipalveluista. Syynä tälle on, että parhaat käytännöt -dokumentti kirjoitettiin vuonna 2007. Pilvipalvelut olivat silloin olemassa, mutta ne eivät olleet kokeneet niiden viimeaikaista räjähdysmäistä nousua. Virtuaalipalvelin, jossa SignServer sijaitsee, on käyttöpalvelutoimittajan palvelin tiimin hallussa. Samoin myös HSM on käyttöpalvelutoimittajan HSM-tiimin hallussa. Tämä on hyödyllistä toimeksiantajalle, sillä toimeksiantajan ei tarvitse ylläpitää omaa HSM:ää. Käyttöpalvelutoimittaja vastaa fyysisestä turvallisuudesta ja operoinnista. Toimeksiantaja vastaa hallinnasta ja valvoo, että käyttöpalvelutoimittaja toimii sovittujen toimintatapojen mukaan.

Kuviossa 7 ilmenee VBA-makrojen allekirjoitusprosessi:



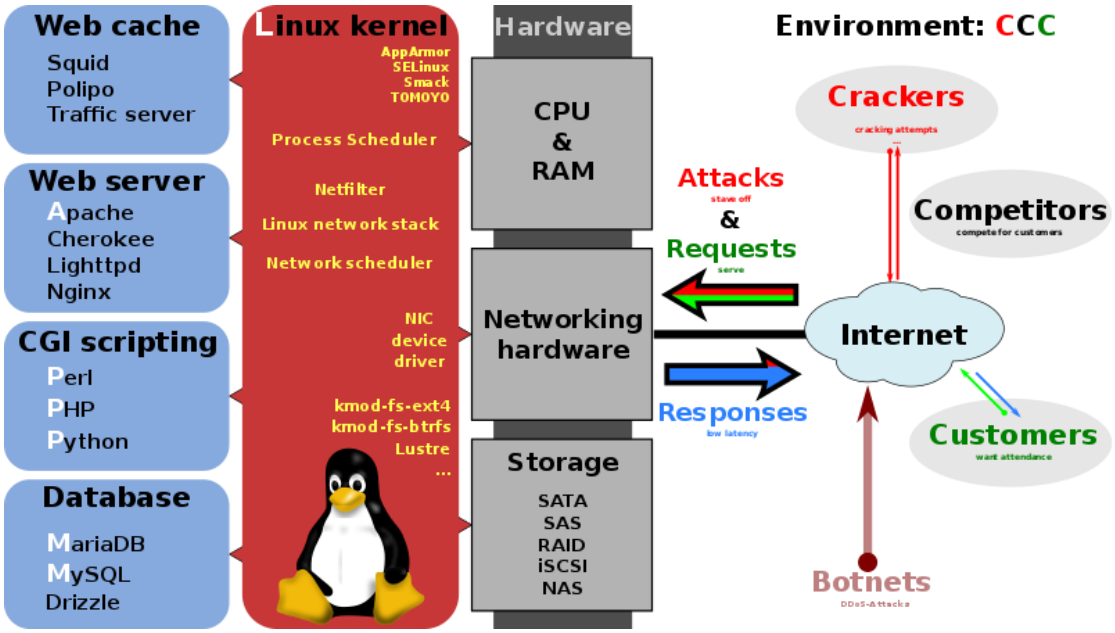
Kuvio 7: Toimeksiantajan malli allekirjoituksen prosessille

Kaikki tiedoston lähetyksen jälkeen tulee toimia automaattisesti.

6 Palvelimen kuvaus

Tämä osio kuvailee ohjelmia, joita SignServer tarvitsee toimiakseen. Toimeksiantaja haluaa pitää salassa ohjelmien tarkat versionumerot tietoturvallisista syistä, joten niitä ei ole mainittu.

Ympäristö luotiin seuraten LAMP-mallia. LAMP on kokoelma ohjelmia, jotka muodostavat yhdessä palvelimen, joka soveltuu dynaamisten Internet-sivujen ja web-ohjelmien rakentamiseen. LAMP-mallin komponentit ovat käyttöjärjestelmä, Internet-palvelin, tietokanta ja ohjelmointikieli. LAMP lyhennys tulee ohjelmista Linux, Apache HTTP Server, MySQL ja PHP, mutta kaikki nämä komponentit voidaan vaihtaa vapaasti. Ainoa yhteys eri osien välillä on, että kaikkien lähdekoodi on avoin. (Semanticscholar 2017.) LAMP valittiin viitekehykseksi ympäristölle, koska SignServerin vaatimukset ympäristölle ovat hyvin yhteensopivia LAMP-mallin kanssa. Kuviossa 8 näkyy LAMP-ympäristön malli. Kaikki punaisen Linux kernel osion oikealla puolella sijaitsevat osiot eivät kuulu tähän opinnäytetyöhön.



Kuvio 8: LAMP-malli (Wikimedia)

Tästä osiosta puuttuu kuvailu virtuaalikoneen ominaisuuksista. Virtuaalikone oli tilattu käyttöpalvelutoimittajalta, joten tarkkaa tietoa koneen kaikista ominaisuuksista ei ollut saatavilla. Tämän lisäksi käyttöpalvelutoimittaja ei halua, että heidän virtuaalikoneiden tietoja jaetaan ilman käyttöpalvelutoimittajan lupaa.

6.1 Linux

Linux on tällä hetkellä yleisin käyttöjärjestelmä maailmassa. Syy tähän on, että Linux on Android-puhelimen käyttöjärjestelmän ydin ja Linux on usein IoT-laitteiden käyttöjärjestelmä

(Computer Hope 2018). Linuxin käyttöjärjestelmäydin on myös ilmainen, jonka takia monet yhtiöt suosivat sitä. Tämän lisäksi Linuxin käyttölisenssi, GNU GPL 2, takaa oikeuden käyttää ja muokata Linuxia vapaasti (Free Software Foundation 1991). Tällöin Linuxin muokkaaminen käyttäjille tai yritykselle soveltuvaksi on hyvin helppoa.

Linux on useasti virtuaalikoneen tai palvelimen käyttöjärjestelmä. Linux on kevyt käyttöjärjestelmä. Linuxin perusasennukselle on asennettu vain välttämättömät ohjelmat. Linux ympäristöä voi siis muokata ilman ongelmia ohjelmien yhteensopivuuden kanssa. Toiseksi Linuxia voi helposti hallita terminaaliyhteydellä, sillä käytännössä kaikki Linuxille luodut ohjelmat on luotu toimimaan komentorivillä. Tämä on hyödyllistä, sillä jos palvelua voi hallita helposti komentorivillä, se ei tarvitse graafista näyttöä. Tällöin se voidaan poistaa käytöstä, ja palvelin ei tuhlaa resursseja näytön piirtämiseen. Linuxia on myös hyvin kestävä käyttöjärjestelmä ja se voi toimia monia kuukausia ilman huoltoa tai ongelmia. Tämä tekee Linuxin käytöstä ja huollosta helppoa. Linuxia pidetään myös tietoturvallisesti vahvana verrattuna muihin käyttöjärjestelmiin. (Hopping & Shepherd 2018.)

6.2 Red Hat Enterprise Linux

Red Hat Enterprise Linux, epävirallisesti lyhennettynä RHEL, on Linux-jakelupaketti, joka on tarkoitettu erityisesti yrityskäyttöön. Alun perin Red Hat -yhtiö tarjosi vain Red Hat Linux -jakelupaketin, mutta Red Hat -yhtiö päätti lakkauttaa sen vuonna 2003. Vuodesta 2003 alkaen, Red Hat -yhtiön jakelupaketit on jaettu kahteen eri jakeluun. (Red Hat 2012.) Fedora on yhteisökehitetty ilmaisjakelu, jota Red Hat -yhtiö sponsoroi, ja Red Hat Enterprise Linux on maksullinen jakelu. Suuri hyöty RHEL:ssä on, että sille on tarjolla tukipalveluja. Jos RHEL:ssä ilmenee ongelmia, niin toimeksiantajan ei tarvitse käyttää tunteja virheiden syiden etsimiseen, vaan he voivat pyytää Red Hatilta apua. Tämän lisäksi RHEL:in ja Fedoran välillä on erityinen suhde. Fedora-yhteisö kehittää Fedora-jakelulle koko ajan ajantasaisia toimintoja, tietoturvaratkaisuja ja palveluita. Nämä valuvat ajan myötä RHEL-jakelulle. (Fedora Project 2018.) Tällöin RHEL pysyy aina ajan tasalla.

RHEL on Linux-jakelu, jota toimeksiantajan käyttöpalvelutoimittaja käyttää virtuaalikoneissa, joten toimeksiantaja valitsi sen. RHEL on yleisesti tunnettu tehokkaana yrityskäyttöön tarkoitettuna jakelupakettina, joten sen käyttöönotto ei ollut ongelma.

6.3 Hardware Security Module

Hardware Security Module (HSM) on tärkeä laite PKI-toteutuksessa. HSM on luotu erityisesti suorittamaan kryptografisia toimintoja. Se voi luoda, hallinnoida, ja säilyttää sisällään yksityisiä avaimia. Avaintenhallinta perustuu ennalta määriteltyihin prosesseihin ja toimintatapoihin, ja se vaatii aina kaksi henkilöä. Koska yksityisten avainten turvallisuus on kriittinen osa PKI:ta, HSM:t ovat erittäin tärkeitä laitteita. (Smirnoff 2017.)

HSM-laitteiden hallinnoinnista ja operoinnista vastaa toimeksiantajan käyttöpalvelutoimittaja. Tämä on pieni riski, sillä fyysistä HSM-laitetta ei säilytetä toimeksiantajan tiloissa. HSM-palvelun ostamisessa on kuitenkin monia hyötyjä. Palvelun ostaminen käyttöpalvelutoimittajalta on halvempaa. Tämän lisäksi toimeksiantajan pitäisi kouluttaa tai palkata omaa henkilöstöä hallinnoimaan HSM:ää, jos he käyttäisivät omaa HSM:ää.

Oleellisin hyöty on tehtävien erottaminen (Gregg, Nam, Northcutt & Pokladnik) henkilöstön kesken. Jos vain yksi henkilö hallinnoisi koko PKI-toteutusta, se mahdollistaisi väärinkäytön ja aiheuttaisi tietoturvauhkan. Nyt kuitenkin on henkilö, joka vastaa PKI:n hallinnasta, toinen HSM:n turvallisuudesta, kolmas pääsystä HSM:n tiloihin ja niin edelleen. Tällöin toiminta on hidasta, mutta turvallista.

SignServerin palvelimelle ei asennettu omaa HSM:ää, vaan sille asennettiin HSM client -yhteys toimeksiantajan käyttämään HSM:ään. HSM-yhteys tarvitaan palvelimelle, koska muuten SignServerin käyttämät yksityiset avaimet pitäisi säilyttää turvattomassa ympäristössä.

6.4 Java

Java on erittäin laajasti käytetty ohjelmointikieli. Sen kehitti Sun Microsystems -yhtiö, mutta Oracle osti Sunin vuonna 2010, joten Oracle omistaa tällä hetkellä Javan. Javaa käytetään erityisesti kehittämään ohjelmia, jotka toimivat kämmenlaitteiden kanssa, ja Javaa käytetään myös web-sovelluksien kehityksessä. (Christensson 2012.)

Java valittiin ohjelmistokieleksi, koska SignServer ja Wildfly ovat Java-pohjaisia ohjelmia. SignServer ja Wildfly eivät toimi, jos Javaa ei ole asennettu palvelimelle.

6.5 Wildfly

Wildfly on Java-pohjainen sovelluspalvelin, joka toimii ohjelmistoalustana SignServerille. Wildfly tunnettiin ennen nimellä JBoss AS, mutta se uudelleen nimettiin Wildfly:ksi, jotta se erottuisi sen maksullisesta serkusta JBoss EAP:sta. (Wildfly 2013.) Red Hat -yhtiö omistaa Wildfly:n ja on vastuussa sen kehityksestä, mutta sen käyttö on ilmainen.

Sovelluspalvelimet ovat hyödyllisiä, sillä ne toimivat yhtenä jalustana monille eri ohjelmille. Wildfly:n käytössä on monia hyötyjä ohjelmille. Wildfly tukee kuormituksen tasaamista, failoveria, yhtäaikaista latausta, ja muita palveluja. (Wildfly 2018.) Tällöin Wildfly:n alla toimivien ohjelmien ei tarvitse kehittää näitä toimintoja erikseen, ja ohjelmien kehittäjät voivat keskittyä parantamaan ohjelmien palvelun laatua. Sovelluspalvelimien käytössä on kuitenkin omat ongelmansa. Asetusten määrittäminen voi jopa kaksinkertaistua, sillä asetukset pitää määrittää ohjelmalle ja Wildfly:lle. Esimerkiksi, jos ohjelma on asetettu hyväksymään 100 MB kokoisia tiedostoja, mutta Wildfly on asetettu hyväksymään vain 10 MB kokoisia tiedostoja, ohjelma voi valittaa asetuserheistä.

Wildfly valittiin SignServerin sovelluspalvelimeksi lisenssipolitiikan takia. Toimeksiantajan ympäristössä on käytössä monia eri sovelluspalvelimia, mutta tämä tekee lisenssien hallinnasta vaikeaa. Täten he ovat vaihtamassa monia sovelluspalvelimiaan Wildfly:ksi. Tuotantoverkossa toimivan SignServerin palvelinta ei ole vielä päätetty. Tämän lisäksi Wildfly toimii todennäköisesti hyvin RHEL-ympäristössä, sillä Red Hat -yhtiö omistaa Wildfly:n.

6.6 MariaDB

MariaDB on yhteisökehitetty MySQL-pohjainen relaatiotietokanta. MariaDB:n kehitys alkoi vuonna 2009. MariaDB on ilmainen, ja sen lähdekoodi on avoin. (MariaDB 2018.)

SignServer tarvitsee tietokannan, sillä sinne säilötään SignServerin lokitiedot. Tietokantaan arkistoidaan myös allekirjoitetut tiedostot. MariaDB on siis oleellinen osa SignServerin toimintaa.

Red Hat Enterprise Linux käyttää MariaDB:ta oletustietokantana (Red Hat 2014), jonka takia MariaDB valittiin. On aina parhainta käyttää kaikkein tuettua ohjelmaa, sillä siitä on saatavilla eniten tietoa ja sille on tarjolla eniten tukea. Tällöin tietokannassa ilmenevät virheet voidaan korjata ripeästi.

6.7 SignServer

SignServer on (PrimeKey 2016) ohjelma, joka automatisoi ohjelmien allekirjoitusprosessin. Tiedosto lähetetään SignServeriin, ja SignServer allekirjoittaa, aikaleimaa, arkistoi tiedoston ja kirjaa kaikki tapahtumat lokiin. SignServerin allekirjoittajat toimivat erillisinä allekirjoitusprosesseina. (engl. worker) Jokainen allekirjoittajaprosessi pitää luoda ja määritellä erikseen. Tällöin SignServeriä pitää määritellä paljon, mutta SignServer on erittäin mukautuva ohjelma, joka soveltuu moniin ympäristöihin.

SignServer automatisoi allekirjoitusprosessin raskaimmat osiot. Sähköisen allekirjoituksen suorittaminen ei ole hankala prosessi, mutta muut osiot allekirjoituksessa vievät paljon aikaa. Jos allekirjoittaja haluaa allekirjoittaa ohjelman, hänen täytyy ensin saada käsiinsä varmenne ja sen yksityinen avain, joilla voi allekirjoittaa tiedostoja.

Allekirjoitusvarmenteiden hallinta on tarkkaa puuhaa. Nokian piti maksaa miljoonaluokan lunnaita kiristäjälle, joka oli saanut Nokian Symbian-käyttöjärjestelmän allekirjoitusavaimen käsiinsä (Lehtinen 2015). Allekirjoittajan pitää siis olla hyvin luotettu tai valvottu henkilö. Allekirjoittajan pitää aikaleimata allekirjoitus, joten hänen pitää ottaa yhteyttä aikaleimauspalveluun. Allekirjoituksen jälkeen allekirjoittajan pitää kirjoittaa tapahtumasta lokiin merkinnän, joten hänen täytyy ottaa yhteyttä keskitetyn lokin hallintaan. Hänen pitää myös arkistoida allekirjoitettu tiedosto. SignServer suorittaa tämän prosessin automaattisesti. Allekirjoitus on nopeaa, tehokasta ja aina määrämuotoista.

SignServer keskittää kaikki palvelut yhteen pisteeseen. Äsken mainitussa esimerkinomaisessa allekirjoitustapauksessa, loki, aikaleimaus ja arkisto ovat kaikki eri palvelimilla. Tämän lisäksi hallintaoikeudet on jaettu eri työntekijöiden kesken. Yhteyttä otetaan eri tahoihin sähköpostitse, ja sähköpostivastauksen saaminen voi kestää kokonaisen työpäivän. SignServer hoitaa kaiken mainitun itse, samalla palvelimella. Esimerkiksi SignServerin loki ottaa helposti talteen kaikki tapahtumat, ja ne voidaan lähettää keskitettyyn lokien hallintaan automaattisesti.

SignServer tarjoaa myös ratkaisun sähköisesti allekirjoitettavien tiedostojen määrälle. Yksi ongelma sähköisten allekirjoitusten käyttöönotossa on, että toimeksiantajan ympäristössä on jo olemassa paljon VBA-makroja. Jos toimeksiantaja yrittäisi ottaa käyttöön VBA-makrojen sähköisen allekirjoituksen ilman SignServeriä, niiden etsimiseen ja allekirjoittamiseen kestäisi vuosia. Ratkaisu tähän ongelmaan on SignServerin mukautuvuus. Toimeksiantaja aikoo jakaa VBA-makrojen allekirjoitusoikeuden valittujen työntekijöiden kesken. Tällöin työntekijät, jotka tuottavat VBA-makroja, voivat hankkia oikeuden allekirjoitukseen ja allekirjoittaa omat VBA-makronsa. Makrot tallentuvat myös SignServerin arkistoon, joten niistä voidaan pitää kirjaa.

SignServer parantaa myös toimeksiantajan jo olemassa olevia toimintoja. DevOPS on IT-organisaation toimintamalli, joka tehostaa ohjelmistokehitystä. Se on saanut vaikutteita ketteristä menetelmistä, etenkin Lean-menetelmästä. DevOPS korostaa ohjelmien ylläpitäjien ja kehittäjien yhteistyötä koko sovelluksen elinkaaren ajan, nopeita kehitysjaksoja ja työn sujuvuuden parantamista automatisoinnilla. Tällöin muutosten teko nopeutuu, kehittäjät saavat jatkuvaa palautetta ja tuote saadaan markkinoille nopeammin. (Solinor.)

Toimeksiantajan DevOPS-tiimi luo Microsoft PE -pohjaisia asennuspaketteja ja ohjelmia, jotka sähköisesti allekirjoitetaan. SignServer parantaa heidän allekirjoitusprosessiaan automatisoinnilla. Riski kuitenkin on, että suuri joukko työntekijöitä saa oikeuden allekirjoittaa PE-tiedostoja, mikäli SignServer allekirjoittaa sekä VBA-makroja että PE-tiedostoja automaattisesti. Jokaiselle allekirjoitusprosessille voidaan kuitenkin määrittää omat käyttäjät. Tällöin suurellekin joukolle työntekijöitä voidaan antaa oikeus allekirjoittaa VBA-makroja ja vain DevOPS-tiimille oikeus allekirjoittaa Microsoft PE -tiedostoja.

7 Testaus

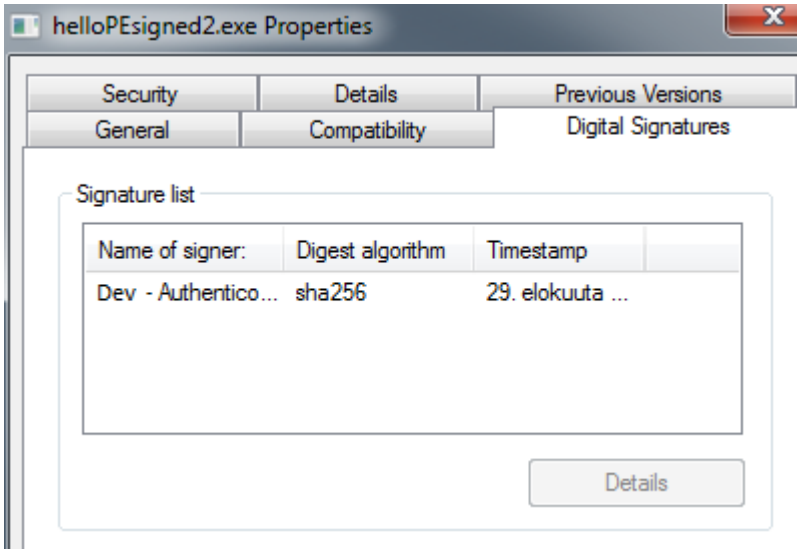
7.1 Microsoft Authenticode -allekirjoitus

Tämä testi suoritettiin kahdessa vaiheessa. Ensiksi suoritettiin viisi allekirjoitusta eri Microsoft PE -tiedostoilla. SignServein ohessa oli asentunut joukko tiedostoja, jotka oli tarkoitettu testaamaan sähköisiä allekirjoituksia. Näiden seassa oli joukko allekirjoittamattomia PE-tiedostoja, joita käytettiin tähän testiin. Allekirjoitusten jälkeen,

tiedostojen allekirjoitukset tarkastettiin. Viitekehyksenä oikealle Authenticode-allekirjoitukselle on Microsoftin oma dokumentti Authenticode-tiedostoista (Microsoft 2008).

Tämä osio ei onnistunut täysin, mutta toimeksiantajalle oleellinen osio onnistui. Toimeksiantaja halusi tietää, pystyykö SignServer sähköisesti allekirjoittamaan Microsoft PE -tiedostoja, esimerkiksi exe- tai msi -tiedostoja, onnistuneesti. Tämä on toimeksiantajalle tärkeä tieto, sillä he käyttävät monia asennuspaketteja ja ohjelmatiedostoja, jotka he ovat kehittäneet itse. Jos niiden sähköinen allekirjoitus toimii onnistuneesti, toimeksiantaja saisi palvelimen, joka hoitaisi PE-tiedostojen automaattisen sähköisen allekirjoituksen. Tämä osio on myös tärkeä toimeksiantajalle, sillä jos Authenticode-allekirjoitus onnistuu, niin toimeksiantajan mielestä myös VBA-makrojen sähköinen allekirjoitus voi onnistua.

Exe-tiedosto sähköisesti allekirjoitettiin onnistuneesti, mutta msi-tiedoston allekirjoitus epäonnistui. Tämän todettiin johtuvan SignServerin sisäisestä toteutusvirheestä juuri msi-allekirjoitusprosessin kohdalla. Tämän hetkinen tieto on, että tämä virhe korjataan SignServerin seuraavassa versiossa. Tärkeintä kuitenkin on, että PE-tiedostojen allekirjoitus toimii. Allekirjoitus täyttää Microsoftin Authenticode PE -dokumentin (Microsoft 2008) vaatimukset. Kuviossa 9 on todiste onnistuneesta sähköisestä allekirjoituksesta, tarkempia yksityiskohtia ei valitettavasti voi ilmoittaa.

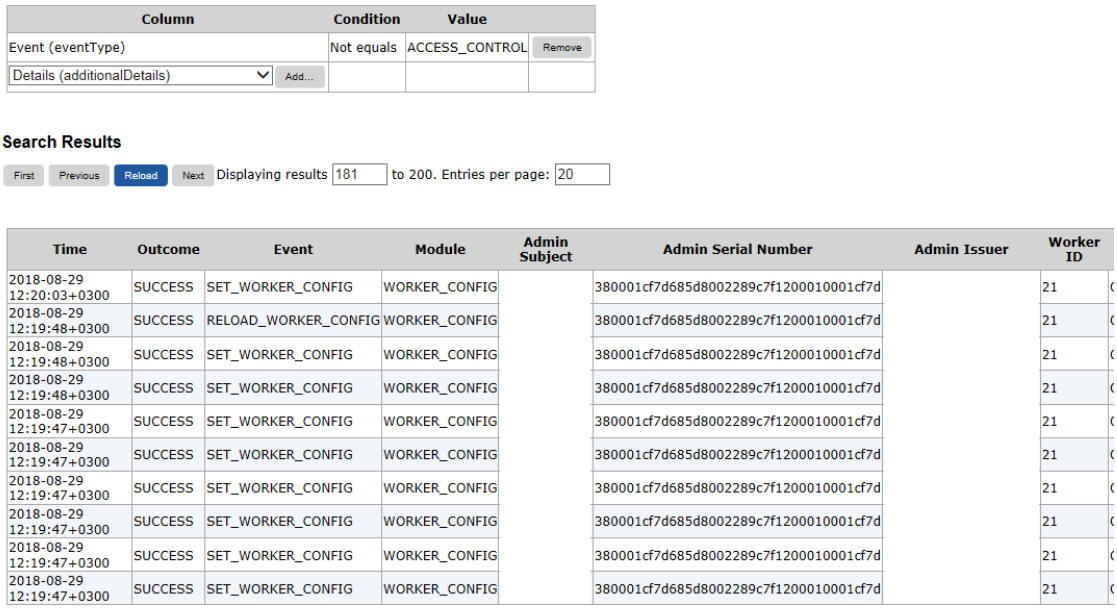


Kuvio 9: Todiste onnistuneesta sähköisestä allekirjoituksesta

7.2 Loki

Tämä testi suoritettiin kahdessa vaiheessa. Loki täytettiin ensin tapahtumilla ja tämän jälkeen lokista tarkastettiin, että mitä tietoa se oli ottanut kirjaan. Tapahtumat olivat muun muassa allekirjoituksia ja asetusmuutoksia eri allekirjoitusprosesseihin.

SignServerin Lokin todettiin toimivan hyvin, paria huomiota lukuun ottamatta. SignServerin loki toimii allekirjoitusprosessikohtaisena. Jokaiselle allekirjoitusprosessille SignServerissä voidaan säätää omat säännöt siitä, mitä kirjataan lokiin ja ei. Tämä tekee lokista erittäin mukautuvan. Lokiin voidaan kirjata allekirjoittajan tunnusnumero, milloin tapahtuma tapahtui, oliko tapahtuma onnistunut ja niin edelleen. Kuviossa 10 on pelkistynyt näkymä SignServerin lokista. Jotkin herkkäluonteiset tiedot on salattu.



Column	Condition	Value	
Event (eventType)	Not equals	ACCESS_CONTROL	Remove
Details (additionalDetails) Add...			

Search Results

Displaying results 181 to 200. Entries per page: 20

Time	Outcome	Event	Module	Admin Subject	Admin Serial Number	Admin Issuer	Worker ID
2018-08-29 12:20:03+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:48+0300	SUCCESS	RELOAD_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:48+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:48+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:47+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:47+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:47+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:47+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:47+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21
2018-08-29 12:19:47+0300	SUCCESS	SET_WORKER_CONFIG	WORKER_CONFIG		380001cf7d685d8002289c7f1200010001cf7d		21

Kuvio 10: Näkymä SignServerin lokista

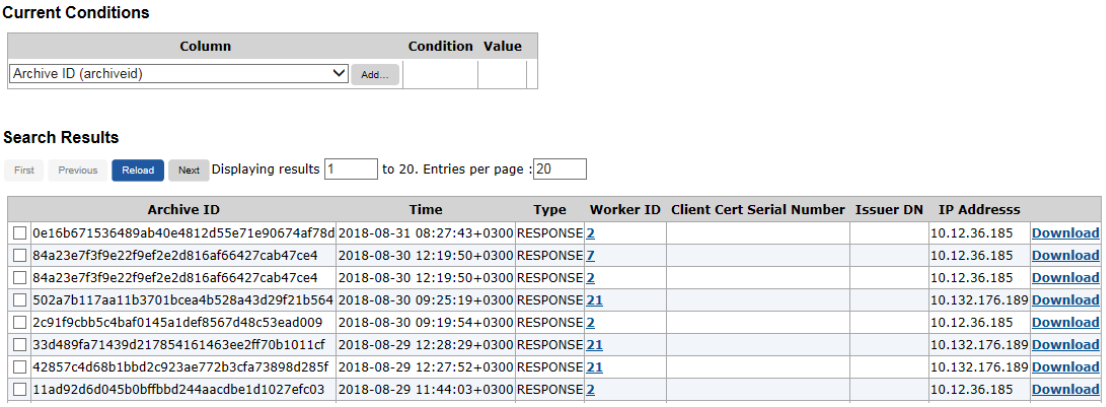
Lokissa on kuitenkin yksi ongelma. Tällä hetkellä se ei tallenna allekirjoituspyyntöä tekevän henkilön varmenteen Common Name:a (CN), vaan pelkästään henkilön varmenteen sarjanumeron. Varmenteen CN on tunniste, jolla on helppo eritellä ja tunnistaa eri varmenteita. Tämän voi tehdä varmenteen sarjanumerolla, mutta se on ihmiselle vaikeaa, sillä sarjanumero on pitkä rivi numeroita. Sarjanumero myös vaihtuu, kun varmenne uusitaan, joten vanhoista sarjanumeroista on pakko pitää kirjaa. On aina parempi käyttää Common Namea. Toimeksiantajan suunnitelma korjata tämä ongelma on pyytää PrimeKeyta lisäämään lokiin toiminnon, jolla voi se ottaa kirjaan Common Namet.

7.3 Arkisto

Arkiston testi suoritettiin samalla tavalla, kuin lokin testi. Eri tiedostoille suoritettiin sähköisiä allekirjoituksia. Tämän jälkeen arkistosta tarkastettiin, että oliko arkisto ottanut tiedoston talteen.

Arkisto soveltui toimeksiantajalle täydellisesti. SignServer säilyttää kopion allekirjoitetusta tiedostosta, niin kuin haluttiin. Arkistoidut tiedostot säilytetään MariaDB:ssa. Arkistossa oli sama ongelma kuin lokissa, se ei pidä kirjaa allekirjoittajien Common Nameista. Tämä ei

kuitenkaan ole yhtä tärkeää arkistossa kuin lokissa. Kuviossa 11 on näkymä SignServerin arkistosta.



Current Conditions

Column	Condition	Value
Archive ID (archiveid) Add...		

Search Results

First Previous Reload Next Displaying results 1 to 20. Entries per page : 20

	Archive ID	Time	Type	Worker ID	Client Cert Serial Number	Issuer DN	IP Addresss	
☐	0e16b671536489ab40e4812d55e71e90674af78d	2018-08-31 08:27:43+0300	RESPONSE	2			10.12.36.185	Download
☐	84a23e7f3f9e22f9ef2e2d816af66427cab47ce4	2018-08-30 12:19:50+0300	RESPONSE	7			10.12.36.185	Download
☐	84a23e7f3f9e22f9ef2e2d816af66427cab47ce4	2018-08-30 12:19:50+0300	RESPONSE	2			10.12.36.185	Download
☐	502a7b117aa11b3701bcea4b528a43d29f21b564	2018-08-30 09:25:19+0300	RESPONSE	21			10.132.176.189	Download
☐	2c91f9cbb5c4baf0145a1def8567d48c53ead009	2018-08-30 09:19:54+0300	RESPONSE	2			10.12.36.185	Download
☐	33d489fa71439d217854161463ee2ff70b1011cf	2018-08-29 12:28:29+0300	RESPONSE	21			10.132.176.189	Download
☐	42857c4d68b1bbd2c923ae772b3cfa73898d285f	2018-08-29 12:27:52+0300	RESPONSE	21			10.132.176.189	Download
☐	11ad92d6d045b0bffbbd244aacdbe1d1027efc03	2018-08-29 11:44:03+0300	RESPONSE	2			10.12.36.185	Download

Kuvio 11: Näkymä SignServerin arkistosta

7.4 Aikaleimaus

Tämä testi olisi suoritettu kahdessa vaiheessa. Ensiksi olisi suoritettu joukko allekirjoituksia, jotka olisi aikaleimattu. Sitten tiedostoista olisi tarkastettu, että oliko ne aikaleimattu onnistuneesti.

Aikaleimausta ei valitettavasti voitu kokeilla. Syy tähän oli hiekkalaatikkoympäristö. Toimeksiantajan hiekkalaatikkoympäristöstä ei pääse millään tavalla ulkoverkkoon, ja toimeksiantaja ei halua käyttää paikallista aikaleimauspalvelua. Toimeksiantajan käyttämä aikaleimauspalvelu on herkkäluonteista tietoa, joten sitä ei tässä kuvata.

Tähän voi lisätä, että toimeksiantaja on varma, että aikaleimaus toiminto toimii. Toimeksiantaja on käyttänyt aikaleimauspalvelua pitkään, ja he uskovat sen toimivan SignServerin kanssa. Jos SignServer ei voisi käyttää ulkoista palvelua aikaleimauksien tekoon, niin SignServer on käytännössä hyödytön toimeksiantajalle. Aikaleimauksen kokeilun puute oli harmillista, mutta testaus ei seisahtanut, vaikka aikaleimausta ei voitu suorittaa.

7.5 Hylätyt toiminnallisuudet

SignServerissä oli monia mielenkiintoisia toiminnallisuuksia, joita toimeksiantaja ei tällä hetkellä näe tarvitsevansa. Ne eivät joko soveltuneet toimeksiantajan ympäristöön, tai niistä ei ollut toimeksiantajalle hyötyä. Tähän osioon on kerätty lista mielenkiintoisista toiminnallisuuksista, joiden käyttöönotto hylättiin. Tämä ei ole täysi lista hylätyistä toiminnallisuuksista.

7.5.1 Asiakkaan puoleinen hajautus

Asiakkaan puoleinen hajautus, englanniksi clientsidehashing, on tekniikka, joka säästää kaistanleveyttä. Normaalisti SignServerille lähetetään allekirjoitettavaksi kokonainen tiedosto, joka voi olla kooltaan melko suuri. Tämä tiedosto myös arkistoidaan MariaDB:hen, joka sijaitsee SignServerin palvelimella. Ajan myötä arkisto täyttyy, jolloin SignServer ei voi enää toimia. Clientsidehashing on menetelmä, joka ei lähetä koko tiedostoa SignServeriin, vaan tiedostosta lasketaan etukäteen hash-arvo, joka lähetetään SignServeriin. Tämä hash-arvo allekirjoitetaan, ja sen jälkeen liitetään alkuperäiseen tiedostoon. Tämä säästää kaistanleveyttä ja palvelimen kovalevyn tilaa. Valitettavasti tämä tekniikka piti hylätä, sillä jos vain hash-arvo lähetetään SignServeriin, niin vain hash-arvo arkistoidaan. Toimeksiantaja haluaa arkistoon kokonaisen allekirjoitetun tiedoston, joten tämä tekniikka hylättiin.

7.5.2 Allekirjoitusprosessit

SignServeriin kuuluu melkein 20 eri allekirjoitusprosessia. Suuri osa niistä esitettiin toimeksiantajan edustajille, ja niistä vain Jarchive Signer kiinnosti toimeksiantajaa tällä hetkellä. Suurin osa allekirjoitusprosesseista ei päässyt tarkastuksen läpi. Esimerkiksi PDF-Signer on allekirjoitusprosessi, joka nimensä mukaan allekirjoittaa PDF-tiedostoja. Tämä on hyödyllistä, sillä allekirjoituksella voi osoittaa, että PDF on toimeksiantajan julkaisema, eikä väärennös. Toimeksiantajan edustajien mielestä siitä ei kuitenkaan ollut tällä hetkellä toimeksiantajalle hyötyä. Toinen esimerkki allekirjoitusprosessista, josta ei ollut toimeksiantajalle hyötyä, on CMS-Signer. Tämä allekirjoitusprosessi allekirjoittaa CMS-viestejä, eli (IETF 2009) kryptografisesti salattuja viestejä. Toimeksiantaja ei käytä CMS-tekniikkaa, mutta SignServerin sisällä CMS-viestejä käytetään erityisesti jo mainitun Clientsidehashing tekniikan kanssa. Koska se hylättiin, CMS-allekirjoittajat hylättiin myös.

7.5.3 Sisäinen aikaleimaaja

SignServeriin on mahdollista säätää sisäinen aikaleimaaja. Sisäinen aikaleimaaja voi suorittaa aikaleimauksen sähköiselle allekirjoitukselle myös hiekkalaatikkoympäristössä. Sisäinen aikaleimaus on kuitenkin tietoturvallisesti ongelmallista. Jos toimeksiantaja aikaleimaisi omat allekirjoituksensa, niin toimeksiantaja voisi mahdollisesti muokata aikaleimauksia myöhemmin. Tämän takia aikaleimauksen suorittavat ulkoiset ja luotetut tahot. Toimeksiantaja hylkäsi tämän toiminnon.

7.5.4 Varmenteiden validoija

Validoija voi tarkastaa, että käytössä olevat varmenteet ovat ajan tasalla. Monta eri validoijaa oli tarjolla, mutta ne kaikki toimivat manuaalisesti. Tästä ei ole hyötyä toimeksiantajalle, sillä varmenteet voidaan tarkastaa itse. SignServer ei myöskään allekirjoita tiedostoa, jos allekirjoittamiseen käytettävä varmenne, on vanhentunut. Tällöin ei ole riskiä siitä, että ohjelma allekirjoitetaan vanhentuneella varmenteella. Validoijan pystyttäminen

vaatisi myös ylimääräisen sulkulistan, eli CRL-palvelun. Tämä palvelu todettiin olevan tarpeeton.

7.5.5 Varmenteiden uusija

Tämä on automaattinen palvelu, joka ajoittain tarkastaa, että allekirjoitusvarmenteet eivät ole vanhentuneet. Se yrittää myös uusia varmenteen, jos se vanhentuisi määritettävissä olevan aikarajan sisällä. Tämä siis estäisi varmenteiden vanhenemisen. Jos varmenne vanhenee, on tilattava ja asennettava uusi varmenne. Tämä olisi hyödyllinen palvelu, sillä se automatisoisi varmenteiden uusimisen, mikä voi olla hankala prosessi.

Valitettavasti tämä toiminto ei sovi toimeksiantajalle, sillä se toimii vain avainvarasto EJBCA:n kanssa. EJBCA (PrimeKey 2018) on varmenteiden haltija -sovellus, jonka PrimeKey on kehittänyt. Koska toimeksiantaja käyttää toista avainvarastoa, tämä toiminto pitää hylätä. Tämä palvelu toimii vain EJBCA-varaston kanssa, koska PrimeKey omistaa EJBCA:n. He voivat siis luoda SignServerille erityisiä toimintoja, jotka toimivat vain heidän tuotteiden kanssa. PrimeKeylla ei ole oikeuksia tehdä muutoksia muihin HSM-laitteisiin.

7.6 Kovennus

Kovennuksella tarkoitetaan tietoturvayhteydessä järjestelmän mahdollisten hyökkäysväylien poistamista, jotta järjestelmän tietoturvataso nousisi. Kovennus on monitasoinen prosessi, johon kuuluu monta eri vaihetta. Järjestelmään asennetaan vain ohjelmia, jota se tarvitsee toimiakseen palveluna. Kaikki ylimääräiset ohjelmat ovat palvelulle hyödyttömiä, ja ne hidastavat palvelun uudelleenkäynnistystä. Ylimääräiset ohjelmat voivat myös avata haavoittuvaisuuksia, joten on parhainta, että niitä ei asenneta ollenkaan. Kaikki käyttämättä olevat portit suljetaan, jotta pääsy järjestelmään vaikeutuu. (OIT UoC 2017.)

Kovennuksen osio tästä työstä oli pienempi kuin alun perin oletettiin. SignServerin palvelin on ostettu käyttöpalvelutoimittajalta. Palvelin on kovennettu käyttöpalvelutoimittajan omien standardien mukaan, ja muiden kovennusohjeiden suorittaminen on vaikeampaa.

Tästä huolimatta kovennusta yritettiin suorittaa. CIS:in (2017) RHEL -kovennusohje on yksi ohjeista, joka valittiin kovennukseen. SignServerin palvelin suoritti kovennusohjeet hyvin. Valitettavasti en voi liittää tarkkaa listausta kovennuksesta, sillä se saattaisi paljastaa mahdollisia hyökkäysväyliä palvelimeen.

Muita kovennusohjeita yritettiin suorittaa, mutta ne eivät onnistuneet täysin. Wildfly-sovelluspalvelinta yritettiin koventaa, mutta toimeksiantajalle sopivia kovennusohjeita ei ollut saatavilla. Tiettyjä yksinkertaisia kovennuksia yritettiin, mutta ne saivat SignServerin kaatumaan, joten ne piti perua. Kovennusohjeiden puute Wildfly:lle ei kuitenkaan ollut suuri ongelma, sillä SignServerin asennusohjeissa (PrimeKey 2016) oli osio, jossa Wildfly:lle tehtiin

asetusmuutoksia. Suurin osa näistä asetuksista kovensivat Wildfly:ta, joten muiden kovennusohjeiden etsiminen ei ollut elintärkeää.

MariaDB:ta yritettiin myös koventaa. MariaDB:n tietopohjasta (MariaDB Knowledge Base) löytyy paljon ohjeita, jotka parantavat sen turvallisuutta. Näistä suoritettiin osa, mutta ei kaikkea. Esimerkiksi Secure Sockets Layer (SSL) yhteyksien asentaminen MariaDB:lle voisi taata, että MariaDB:n välistä tietoliikennettä ei voisi vakoilla. Suurin osa MariaDB:n tietoliikenteestä on kuitenkin Wildfly:n ja MariaDB:n välillä, eli tietokoneen sisäistä keskustelua. Jos hyökkääjä voisi kuunnella tätä tietoliikennettä, hän voisi tehdä paljon enemmän tuhoa kuin vakoilla tietokantaa. MariaDB:n kovennus todettiin tarpeettomaksi.

8 Johtopäätökset ja kehittämisehdotukset

SignServerin todettiin soveltuvan toimeksiantajalle. Ainoa puute täydellisestä testauksesta on aikaleimaus. Sitä ei valitettavasti saatu toimimaan, koska palvelimesta ei päässyt ulkoverkkoon sisäinen politiikan takia. Tämä on mielestäni aihe, joka voi olla toimeksiantajalle ongelma. Jos SignServer ei voi ottaa yhteyttä toimeksiantajan käyttämään aikaleimauspalveluun, niin SignServer ei kelpaa heille. Toimeksiantajan pitää tarkistaa aikaleimauksen toimivuus ripeästi.

Toinen asia joka voi aiheuttaa ongelmia toimeksiantajalle, on SignServerin integraatio sen välityspalvelimeen. Tätä ei voitu kehittää hiekkalaatikkoympäristössä. Tämän tarkistus on tärkeää, sillä toimeksiantaja haluaa käyttää välityspalvelinta SignServerin käytön todennuksessa. Sen tarkistus on kuitenkin ongelmallista, sillä kaikki ohjelmat, joiden halutaan olevan yhteydessä välityspalvelimeen, pitää liittää siihen erikseen. Liitos voi siis olla helppo tai vaikea, ja se voi kestää vähän tai paljon aikaa. On tärkeää saada liitoksen kehitys aluille niin pian kuin mahdollista.

Toimeksiantajan pitää myös kehittää sovellus, jotta toimeksiantaja voi toteuttaa oman mallinsa SignServerille. Tämä ei kuitenkaan ole ongelma. SignServer on mukautuva ohjelma, ja se voi ottaa komentoja, jotka lähetetään verkkoympäristön läpi. Sovelluksen kehityksen oletetaan menevän hyvin.

SignServeristä löytyi yksi allekirjoitusprosessi, joka oli toimeksiantajan mielestä hyödyllinen heille. Jarchive Signer on allekirjoitusprosessi, joka sähköisesti allekirjoittaa Java-asennuspaketteja ja eri arkistoja. Tämä on hyödyllinen allekirjoitusprosessi, sillä ohjelmistoasennuspakettien turvallisuus on tärkeää.

Toinen SignServerin sisäinen prosessi, joka on toimeksiantajalle hyödyllinen, on HSMKeepAliveService. Tämä on automaattisesti toimiva palvelu, joka testaa yhteyttä HSM:ssä sijaitsevaan yksityiseen avaimeen. Tämä on hyödyllistä, sillä on riski, että yhteys kryptoavaimeen voi katketa, jos sitä ei käytetä pitkällä aikavälillä. Tämä palvelu yrittää estää

yhteyden katkeamista testailemalla yhteyttä kryptoavaimeen ajoittain. Tämä toiminto saatiin testattua, ja sen todettiin toimivan hyväksyttävästi.

Toimivaa SignServer palvelinta ei saatu tuotantoympäristöön. Tämä ei ollut välttämätöntä opinnäytetyölle, mutta olisin halunnut saada toimivan palvelimen toimintaan. Yksi syy tähän on, koska projekti hidastui noin puolitoista kuukautta. SignServerin toimituksessa oli hidasteita, yhden vakavan virheen korjaaminen kesti pari viikkoa ja projekti suoritettiin kesän aikana, jolloin työohjaajat olivat kesälomilla. Toisaalta ohjaajien mukaan tällaisen palvelun käyttöönottaminen tuotantoon vie paljon aikaa, joten olisi epätodennäköistä, että tuotantoon saataisiin toimiva palvelin parissa kuukaudessa.

Kovennus oli pienempi osio tästä työstä kuin alun perin oletettiin. Palvelimen osto käyttöpalvelutoimittajalta ja salassa pidettävien tietojen kuvaamatta jättäminen ovat suurimmat syyt tähän. Koska palvelin oli hankittu käyttöpalvelutoimittajalta, niin palvelimen ympäristön muokkaaminen olisi ollut huono idea. Minulla ei myöskään ollut riittäviä hallintaoikeuksia, että olisin voinut muokata ympäristön tärkeitä muuttujia. Kovennuksen suoritus oli siis mahdotonta.

Tämän työn aikana korostui ketterien menetelmien parhain osio, joka on suoran viestinnän tärkeys. Minulla oli viikoittainen tapaaminen ohjaajieni kanssa, mutta tämän lisäksi yksi ohjaajistani istui vierelläni. Täten sain palautetta työstäni milloin tahansa. Ketterien menetelmien puolesta puhujat väittävät (Itewiki) suoran, kasvokkain viestinnän olevan parhain tapa viestiä, ja olen samaa mieltä.

Yksi ketterien menetelmien heikkous ilmeni tämän projektin aikana. Mitä tapahtuu, kun viestintä puuttuu? Tämä opinnäytetyö tehtiin kesän aikana, joten projektin ohjaajat olivat kesälomalla mittavan ajan. Tämä aiheutti lopulta elokuun alussa tukalan tilanteen, sillä tarvitsin kuittauksen ohjaajalta. Koska viestintää tai tarkkaa projektidokumentaatiota ei ollut, puolitoista työviikkoa menetettiin. Sain kehitettyä muita osioita tämän ajanjakson aikana, mutta viestinnän puute hidasti projektia.

Tämän opinnäytetyön kirjaamisen aikana ilmeni salassapidon tärkeys. Alun perin oli ilmoitettu, että toimeksiantajan nimi voitaisiin mainita valmiissa opinnäytetyössä. Tämä ei kuitenkaan toteutunut. Tämän lisäksi oletin, että kaikkien ympäristössä käytettyjen ohjelmien versionumerot voitaisiin ilmoittaa valmiissa opinnäytetyössä. Tällöin tutkimuksellinen periaate tutkimuksien toistamisesta voitaisiin suorittaa helposti. Toimeksiantaja ei kuitenkaan suostunut tähän. Toimeksiantaja ei halua kuvailla heidän ympäristöstä hyökkäysväyliä, jotka versionumerot paljastavat. Toistoperiaatteen suorittaminen on nyt vaikeampaa, mutta ymmärrän toimeksiantajan näkemyksen salaisuudesta.

Salassapitovelvoite siis vaikeuttaa opinnäytetyön toistuvuutta ja käyttökelpoisuutta. Ideoin opinnäytetyön alkuvaiheissa, että tämä opinnäytetyö voisi opastaa SignServerin käyttöönottoa muiden yritysten ympäristöissä. Tämä ei kuitenkaan toteutunut salassapitovelvoitteen takia. Suurin ongelma on, että HSM on täysin salattu. Koska siitä ei ole ilmoitettu mitään tietoa, tämä opinnäytetyö ei voi opastaa SignServerin asennuksessa.

Näitä tuloksia ei voi siis käyttää apuna, jos haluaa asentaa SignServerin. Tähän voi lisätä maininnan, että tämän opinnäytetyön ohella luotiin tarkat asennusohjeet SignServerille. Niitä seuraamalla voi asentaa SignServerin tässä työssä käytettyyn hiekkalaatikkoympäristöön. Ongelma on, että asennusohjeet kytkeytyvät käyttöpalvelutoimittajan ympäristöön, ja ohjeet ovat toimeksiantajan omaisuutta. Niitä ei voi liittää tähän opinnäytetyöhön.

Tämä opinnäytetyö toimii enemmänkin vakuutuksena, että SignServerillä voi ottaa käyttöön sähköisen allekirjoituksen. Liki kaikki yhtiöt, jotka ovat keskittyneet ohjelmistokehitykseen, tarvitsevat sähköistä allekirjoitusta. Sähköinen allekirjoitus takaa, että tiedostoa ei ole muutettu, ja ilmoittaa tiedoston allekirjoittajan. Ei voi ikinä olla varma, onko allekirjoittamaton ohjelma turvallinen. Hyökkääjä voi helposti naamioida viruksen toiseksi ohjelmaksi, jos ohjelmaa ei ole sähköisesti allekirjoitettu. Yhtiöllä on suuri maineriski, jos se ei sähköisesti allekirjoita ohjelmiaan, joten on aina parhainta ottaa se käyttöön.

Lähteet

Painetut

Vacca, J. 2014. Managing Information Security. 2. painos. Waltham: Syngress.

Sähköiset

Agile Manifesto. 2001. Manifesto for Agile Software Development. Viitattu 7.11.2018
http://agilemanifesto.org/

CA Security Council. 2016. Code Signing Whitepaper. Viitattu 15.11.2018
https://casecurity.org/wp-content/uploads/2016/12/CASC-Code-Signing.pdf

Center for Internet Security. 2017. CIS Red Hat Enterprise Linux Benchmark. Viitattu 19.10.2018
https://www.cisecurity.org/cis-benchmarks/

Christensson, P. 2012. Java Definition. Viitattu 8.11.2018
https://techterms.com/definition/java

Computer Hope. 2018. What is the most popular operating system? Viitattu 24.10.2018
https://www.computerhope.com/issues/ch001777.htm

Digicert. Protecting Code Signing Certificate Private Keys. Viitattu 15.11.2018
https://www.digicert.com/code-signing/best-practice-guide.htm

Fedora Project. 2018. Fedora and Red Hat Enterprise Linux. Viitattu 8.11.2018
https://docs.fedoraproject.org/en-US/quick-docs/fedora-and-red-hat-enterprise-linux/index.html

Free Software Foundation. 1991. GNU General Public License, version 2. Viitattu 15.11.2018
https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

Furuhed, M. 2018. Public key infrastructure (PKI) explained in 4 minutes. Viitattu 8.11.2018
https://www.nexusgroup.com/blog/crash-course-pki/

Gregg, J., Nam, M., Northcutt, S. & Pokladnik, M. Separation of Duties in Information Technology. Viitattu. 17.10.2018
https://www.sans.edu/cyber-research/security-laboratory/article/it-separation-duties

Hopping, C. & Shepherd, A. 2018. The Benefits of Linux Servers. Viitattu 15.11.2018
https://www.itpro.co.uk/linux/28951/the-benefits-of-linux-servers

IETF. 2009. RFC 5652- Cryptographic Message Syntax (CMS). Viitattu 25.10.2018
https://tools.ietf.org/html/rfc5652

Itewiki. Ketterät menetelmät, agile, LEAN ja scrum. Viitattu 18.10.2018
https://www.itewiki.fi/opas/ketterat-menetelmat-agile-lean-ja-scrum/

Lehtinen, P. 2015. Täydellinen rikos? Nokia maksoi kiristäjälle miljoonien lunnaat – poliisi täysin ymmällään. Viitattu 8.11.2018
https://www.mtvuutiset.fi/artikkeli/taydellinen-rikos-nokia-maksoi-kiristajalle-miljoonien-lunnaat-poliisi-taysin-ymmallaan/5233636#gs.XDdkoq8

Lynch, V. 2018. Best Practices for Timestamping. Viitattu 8.11.2018
https://www.digicert.com/blog/best-practices-timestamping/

MariaDB. 2018. About Us. Viitattu 18.10.2018
https://mariadb.com/about-us/

MariaDB Knowledge Base. Securing MariaDB. Viitattu 19.10.2018
https://mariadb.com/kb/en/library/securing-mariadb/

Microsoft. 2007. Code Signing Best Practices. Viitattu 19.10.2018.
http://download.microsoft.com/download/a/f/7/af7777e5-7dcd-4800-8a0a-b18336565f5b/best_practices.doc

Microsoft. 2008. Windows Authenticode Portable Executable Signature Format. Viitattu 24.10.2018
http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/authenticode_pe.docx

Microsoft. 2018a. Office VBA Reference. Viitattu 5.11.2018.
https://docs.microsoft.com/en-us/office/vba/api/overview/

Microsoft. 2018b. Ransomware. Viitattu 15.11.2018.
https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/ransomware-malware

New Line Technologies Blog. 2018. Incremental Model of Software Development Life Cycle. Viitattu 7.11.2018
http://newline.tech/blog/incremental-model-of-software-development-life-cycle/

OIT University of Colorado. 2017. Security Awareness- Hardening Your Computer. Viitattu 18.10.2018
https://oit.colorado.edu/it-security/security-awareness/hardening-your-computer

PrimeKey. 2016. SignServer - Manual. Viitattu 24.10.2018
https://www.signserver.org/doc/current/manual.html

PrimeKey. 2018. EJBCA – The Open Source CA. Viitattu 16.10.2018
https://www.ejbca.org/

Red Hat. 2012. 10 years of Red Hat Enterprise Linux. Viitattu 24.10.2018
https://www.redhat.com/en/10yearsofrhel

Red Hat. 2014. Chapter 17. Web Servers And Services. Viitattu 18.10.2018
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/7.0_release_notes/chap-red_hat_enterprise_linux-7.0_release_notes-web_servers_and_services

Seltzer, L. 2013. Securing Your Private Keys As Best Practice for Code Signing Certificates. Viitattu 15.11.2018
https://www.symantec.com/content/en/us/enterprise/white_papers/b-securing-your-private-keys-csc-wp.pdf

Semanticscholar. 2017. LAMP (Software Bundle). Viitattu 28.10.2018
https://www.semanticscholar.org/topic/LAMP-(software-bundle)/166909

Smirnoff, Peter. 2017. Understanding Hardware Security Modules (HSMs). Viitattu 17.10.2018
https://www.cryptomathic.com/news-events/blog/understanding-hardware-security-modules-hsms

Solinor. DevOPS-Tehostettua ohjelmistokehitystä. Viitattu 15.11.2018

https://solinor.fi/devops-tehostettua-ohjelmistokehitysta/

Wildfly. 2013. Why Rename Jboss AS? Viitattu 17.10.2018
https://web.archive.org/web/20130928204119/http://www.wildfly.org/faq/

Wildfly. 2018. What is Wildfly? Viitattu 17.10.2018.
http://wildfly.org/about/

Kuviot