Tampereen ammattikorkeakoulu

Kertakäyttöviestipalvelu turvalliseen salaisten tietojen vaihtamiseen

Mirjami Laiho

OPINNÄYTETYÖ
Marraskuu 2024

Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

LAIHO, MIRJAMI:
Kertakäyttöviestipalvelu turvalliseen salaisten tietojen vaihtamiseen

Opinnäytetyö 60 sivua, joista liitteitä 0 sivua
Marraskuu 2024

Yritysten sisäiseen ja ulkoiseen viestintään on olemassa paljon erilaisia palveluja ja alustoja. Usein riskinä on kuitenkin tietojen pitkäaikainen tallentuminen palvelimille ja rajattu näkyvyys tietojen käsittelystä kolmannen osapuolen toimesta. Opinnäytetyön taustalla oli toimeksiantajayrityksen tarve yhtenäistää ja helpottaa turvallista salaisten tietojen vaihtamista sekä yrityksen sisällä että sen sidosryhmien kanssa.

Tässä opinnäytetyössä yhtenäistettiin toimeksiantajayrityksen käytäntöjä salaisten tietojen lähettämisessä ja vastaanottamisessa yrityksen sisällä, sekä parannettiin tiedonvaihdon turvallisuutta. Työn osana kehitettiin kertakäyttöviestipalvelu yrityksen sisäiseen käyttöön sekä muodostettiin dokumentaatio palvelun kehittämisestä, sen taustoista sekä jatkokehityskohteista.

Opinnäytetyön toiminnallisessa osuudessa suunniteltiin ja toteutettiin kertakäyttöviestipalvelu, jota voidaan käyttää yrityksen sisäiseen salaisten tietojen vaihtoon. Palvelun toteutuksen myötä syntyi dokumentaatio, joka sisältää palvelun kehityksen ja sen taustojen kuvailun, tietoa palvelun rakenteesta ja ominaisuuksista sekä mahdolliset jatkokehityskohteet. Työssä kuvattiin myös salaisuuksien turvallisen hallinnan periaatteita ja kertakäyttöviestien käyttöä tietoturvatavoitteiden tukena. Näiden periaatteiden tunteminen vahvistaa käyttäjien tietoturvatietoisuutta ja tukee turvallisia toimintatapoja salaisten tietojen käsittelyssä. Opinnäytetyössä tarkasteltiin myös palvelun toteutuksen pohjana käytettyä HashiCorp Vaultia. Vaultin esittely tuo esiin sen tarjoamat mahdollisuudet salaisuuksien turvalliseen käsittelyyn ja hallintaan. Tämä tukee sekä palvelun jatkokehitystä että uusien ratkaisujen suunnittelua ja toteutusta.

Palvelun jatkokehityksessä pyritään parantamaan olemassa olevia ominaisuuksia ja käytettävyyttä sekä lisäämään uusia ominaisuuksia. Palvelua on tarkoitus testata yrityksen sisällä ja testihuomioiden perusteella kehittää palvelua vielä turvallisemmaksi ja käyttäjiä paremmin palvelevaksi. Pitkän aikavälin tavoitteena on laajentaa kertakäyttöviestipalvelun käyttöä myös yrityksen ja sidosryhmien väliseen turvalliseen tiedonjakamiseen.

Asiasanat: kertakäyttöviesti, salaisuudet, hashicorp vault, salaisuuksien hallinta

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Development

LAIHO, MIRJAMI:
One-time Message Service for Secure Exchange of Confidential Data

Bachelor's thesis 60 pages, appendices 0 pages
November 2024

This thesis addresses the need for the commissioning company to standardize and secure the exchange of sensitive information internally and with stakeholders. Existing communication platforms often contain the risk of long-term data storage on servers and lack transparency in data handling by third parties.

The objective of this thesis was to unify the company's practices for securely exchanging sensitive information internally. The purpose of this thesis was to develop a one-time message service for internal use and to create documentation outlining the development process, background, and potential future enhancements of the service.

This thesis developed a one-time message service for internal use within the company. Key principles of secure information management were also analyzed, including the benefits and risks of one-time messages, best practices in secrets management, and the use of HashiCorp Vault for secure data handling. Vault's capabilities and principles of one-time messaging provide a strong foundation for secure practices and future service improvements, promoting awareness and better handling of sensitive data.

The service will undergo internal testing to gather feedback, refine usability, and improve security. Long-term development aims to extend the service's use for secure communication with external stakeholders, ensuring a user-friendly and robust solution for sensitive information exchange.

Key words: one-time message, secrets, hashicorp vault, secrets management

SISÄLLYS

LYHENTEET JA TERMIT

API	Ohjelmointirajapinta (Application Programming Interface)
CLI	Komentorivi (Command Line Interface)
OAuth-token	Käyttöoikeustunnus, joka myönnetään käyttäjälle tai sovellukselle OAuth-protokollan avulla
OIDC	OpenID Connect -tunnistautumisprotokolla, joka tarjoaa todennuksen käyttäjän identiteetistä
RBAC	Roolipohjainen käyttöoikeuksien hallinta (Role-Based Access Control), menetelmä, jossa käyttäjille myönnetään käyttöoikeudet roolien mukaan
TTL	Voimassaoloaika (Time to Live), ajanjakso, jonka ajan tieto, yhteys tai resurssi on voimassa ennen vanhentumistaan tai poistumistaan
UI	Käyttöliittymä (User Interface)

1 JOHDANTO

Kun yritykset siirtävät yhä enemmän palveluitaan pilveen, myös tarve jakaa salaisia tietoja, kuten salasanoja, varmenteita ja avaimia, kasvaa. Näiden tietojen turvallinen jakaminen yrityksen sisällä sekä ulkopuolisten tahojen kanssa voi tuoda mukanaan haasteita. Vaikka markkinoilla on useita palveluja tietojen turvalliseen vaihtamiseen, monissa niistä puuttuu riittävä näkyvyys siitä, mihin kaikkialle salaiset tiedot tallentuvat ja miten niitä tosiasiassa käytetään. Jos yrityksellä ei ole yhtenäisiä ja turvallisiksi todettuja prosesseja salaisten tietojen jakamiseen, salaisuuksien hallinta ja seuranta on vaikeaa, mikä lisää tietojen väärinkäytön riskiä.

Opinnäytetyön toimeksiantajana toimii Haltu Oy. Työn tavoitteena on yhtenäistää yrityksen käytäntöjä salaisten tietojen lähettämisessä ja vastaanottamisessa yrityksen sisällä, ja parantaa salaisten tietojen vaihtamisen turvallisuutta. Tavoitteena on myös, että kehitettyä käytäntöä voidaan myöhemmin hyödyntää turvallisen tiedonjakamisen jatkokehittämisessä koskemaan myös ulkopuolisten sidosryhmien kanssa tapahtuvaa tiedonvaihtoa. Työn tarkoituksena on kehittää kertakäyttöviestipalvelu Haltu Oy:n sisäiseen käyttöön sekä muodostaa dokumentaatio palvelun kehittämisestä, sen taustoista sekä jatkokehityskohteista.

Työssä käsitellään ensin yleisesti salaisuuksien hallinnan ja kertakäyttöviestien perusperiaatteita. Tämän jälkeen syvennytään HashiCorp Vaultiin, salaisuuksien ja salauksen hallintajärjestelmään, jota voidaan hyödyntää turvallisessa salaisuuksien hallinnassa, ja joka toimii perustana kehitettävälle kertakäyttöviestipalvelulle. Lopuksi kuvataan opinnäytetyön tuotoksena muodostuneen kertakäyttöviestipalvelun toiminnallisuudet, rakenne ja kehitysprosessi sekä hahmotellaan palvelun tulevia kehitystarpeita.

2 SALAISUUDET JA SALAISUUKSIEN HALLINTA

2.1 Salaisuuksien määritelmä ja tyypit

Salaisuudella (secret) tyypillisesti tarkoitetaan arkaluonteista tietoa, jonka avulla voidaan tunnistautua suojattuun järjestelmään. Salaisuus voi olla myös salausavain tai -avainpari, jota käyttämällä salausta vaativia tietoja voidaan salata tai purkaa. (Lundberg 2019b; IBM Cloud 2024.) Esimerkkejä salaisuuksista ovat käyttäjätunnukset ja salasanat, API-avaimet, tokenit, SSH-avaimet, salausavaimet ja varmenteet (Cloudflare 2024; CyberArk 2024b).

Salaisuudet voivat olla luonteeltaan staattisia tai dynaamisia (IBM Cloud 2024). Staattiset salaisuudet ovat pitkäikäisiä ja säilyvät muuttumattomina pitkiä aikoja. Niitä käytetään perinteisesti käyttäjätileihin tunnistautumiseen, palveluiden väliseen todennukseen tai sovellusten ja tietokantojen välisiin yhteyksiin. (Alvas 2024.) Staattiset salaisuudet eivät muutu, ellei joku muuta niitä manuaalisesti. Tämän vuoksi ne muuttuvat yleensä harvoin. (HashiCorp n.d.-b.) Dynaamiset salaisuudet taas ovat lyhytikäisiä ja ne luodaan pyynnöstä. Näin voidaan muodostaa tilapäinen pääsy resursseihin. Käytännössä dynaamiset salaisuudet luodaan esimerkiksi keskitetyn salaisuuksien hallintajärjestelmän kautta. Palvelun tai sovelluksen pyytäessä pääsyä, järjestelmä kommunikoi kohderesurssin kanssa luodakseen väliaikaiset tunnistetiedot. Kohderesurssi voi tässä yhteydessä olla esimerkiksi tietokanta tai pilvipalvelu. Luodut tunnistetiedot peruutetaan automaattisesti, kun niille määritelty voimassaoloaika (TTL, Time to Live) tulee täyteen. (Alvas 2024.)

Staattisten salaisuuksien etuna on niiden helppokäyttöisyys ja yksinkertaisuus. Niiden käyttöönotto ja integrointi sovelluksiin on nopeaa, mikä lyhentää kehitykseen kuluvaa aikaa. Yksinkertaisuudella on kuitenkin kääntöpuolensa, sillä staattisiin salaisuuksiin liittyy paljon turvallisuusriskejä. Koska staattiset salaisuudet pysyvät pitkiä aikoja muuttumattomina, ne ovat ajan kuluessa aina vain alttiimpia paljastumiselle. Jos hyökkääjä saa salaisuuden hallintaansa, sen avulla on mahdollista saada pitkäaikainen pääsy arkaluonteisiin tietoihin ja järjestelmiin. (Alvas 2024.)

Dynaamisten salaisuuksien etuna on pienempi mahdollisuus salaisuuden paljastumiselle. Kun salaisuudella on lyhyt voimassaoloaika, pienentyy myös aikaikkuna salaisuuden joutumiselle vääriin käsiin. Ja koska dynaaminen salaisuus poistuu käytöstä sen voimassaoloajan päätyttyä, sen avulla ei voi saada pitkäaikaista pääsyä järjestelmiin tai tietoihin. Lisäksi salaisuuden elinkaaren hallinnan ollessa automatisoitu, poistuu inhimillisten virheiden mahdollisuus ja hallinnollisen työn määrä vähenee. (Alvas 2024.)

Dynaamisten salaisuuksien käyttöönotto ei ole kuitenkaan aina yksinkertaista. Vaikka dynaamiset salaisuudet ovat skaalautuvia ja voidaan helposti ottaa käyttöön useissa pilvimpäristöissä ja palveluissa, käyttöönotto vaatii huolellista integrointia olemassa oleviin järjestelemiin ja voi tuoda niihin monimutkaisuutta. Lisäksi dynaamisia salaisuuksia luovan ja käyttävän järjestelmän tulisi olla hyvin saatavilla, jotta kriittiset automaattiset tehtävät eivät kaadu käyttökatkoksiin. Dynaamisten salaisuuksien tarkastaminen tai niiden käytön seuraaminen on myös haastavampaa kuin staattisten niiden lyhytikäisyyden vuoksi. Järjestelmän toimintoihin voi myös aiheutua viivettä lyhytkestoisten salaisuuksien luomisen, jakelun ja validoinnin tapahtuessa reaaliajassa. Viive on useimmiten minimaalinen, mutta tulee ottaa huomioon etenkin aikasidonnaisten tai paljon käytettyjen sovellusten tai järjestelmien kohdalla. (Alvas 2024.)

2.2 Salaisuuksien hallinta

Salaisuuksien hallinnalla yleisesti tarkoitetaan arkaluonteisten tietojen turvallista tallentamista (Cloudflare 2024). Salaisuuksien hallinta ei ole yksittäinen asia, vaan koostuu pikemminkin useasta eri turvallisuuteen liittyvästä osa-alueesta, jotka yhdessä varmistavat salaisuuksien asianmukaisen hallinnan ilman vaaraa salaisten tietojen joutumisesta vääriin käsiin. Näitä osa-alueita ovat muun muassa roolipohjainen käyttöoikeuksien hallinta (RBAC, role-based access control), todennettu pääsynhallinta ja -seuranta sekä periaatteet oikeuksien myöntämisestä. (Red Hat 2024.)

Salaisuuksien hallintaan on olemassa monenlaisia työkaluja ja valmiita ratkaisuja. Salaisuudenhallintaratkaisut on suunniteltu avuksi turvalliseen salaisuuksien tallentamiseen, hallintaan ja jakamiseen. Hallintaan kuuluu muun muassa turvallisuuskäytäntöjen ylläpitäminen ja pääsynseuranta lokien avulla. (Witts 2024.) Valmiita ratkaisuja salaisuuksien hallintaan ovat esimerkiksi AWS Secrets Manager, Google Cloud Secrets Manager ja HashiCorp Vault. Vaikka perusajatus eri salaisuuksien hallinnan työkaluissa on sama, on niissä tarjolla useita erilaisia ominaisuuksia, ja ratkaisuista kannattaakin valita omaan käyttötarkoitukseen soveltuvin. (Chuvakin & Viswanathan 2023; Witts 2024.)

AWS Secrets Manager on Amazon Web Servicen salaisuuksien hallintatyökalu, jonka avulla voi hallita ja kierrättää salaisuuksia. Työkalulla on API, jota hyödyntämällä sovellukset voivat hakea turvallisesti tallennettuja salaisuuksia. Pääsyä salaisuuksiin voidaan hallinnoida määrittelemällä käyttöoikeuksia ja käytäntöjä ja alusta tarjoaa myös kattavat toiminnot valvontaan auditointityökalujen avulla. Myös Google Cloud Secrets Manager tarjoaa samankaltaisia ominaisuuksia. Sopivaa alustaa valitessa on hyvä pitää mielessä, että käyttöönotossa voi olla eduksi jos organisaatio käyttää jo valmiksi saman palveluntarjoajan muita palveluita. Esimerkiksi AWS Secrets Manager toimii hyvin yhteen Amazonin muiden tuotteiden kanssa ja Google Cloud Secrets Manageriin voi liittää roolipohjaiset käyttöoikeudet Google Cloudin identiteetinhallinta-alustan avulla. (Witts 2024.)

2.2.1 Salaisuuksien hallinnan haasteet ja riskit

Organisaatioiden siirtäessä toimintaansa jatkuvasti enenevissä määrin pilveen, myös tietoturvariskien määrät kasvavat (Johnson 2024). Organisaatioiden salaisuuksien turvallisuus on yksi suurimmista riskeistä, joita tulisi hallita ympäristöjä ja sovellusinfrastruktuuria suojatessa. Salaisuuksien hallintaan liittyen nousee esiin ainakin neljä riskitekijää: salaisuuksien hallitsematon leviäminen (secrets sprawl), inhimillinen erehdys (human error), erilaisten vaatimusten noudattamista koskevat rikkomukset sekä käyttökatkot (downtime). (Chuvakin & Viswanathan 2023.)

Salaisuuksien hallitsematon leviäminen (secrets sprawl) tarkoittaa säilytettävien salaisuuksien leviämistä useaan eri paikkaan niin, että niiden seuraaminen ja hallinta vaikeutuu merkittävästi. Yhä useampi organisaatioiden työtehtävistä on siirretty pilvipalveluihin, mikä kasvattaa räjähdysmäisesti niihin liittyvien salaisuuksien määrää. (Chuvakin & Viswanathan 2023.) On yhä tavallista, että salaisuuksia säilytetään selkotekstinä monissa eri paikoissa, kuten versionhallintajärjestelmissä, konfiguraationhallintajärjestelmissä, tiedostoissa tai CI/CD-putkissa. Organisaatioiden ja työntekijöiden osaamisen kehittyessä on siirrytty parempiin tallennustekniikoihin, kuten salattuihin asemiin tai tiedostojen salaamisen ja purkamiseen. Turvallisemmista tallennustekniikoista huolimatta salaisuudet silti monesti leviävät useisiin paikkoihin, jollei niitä hallita oikein. Tällöin on vaikeaa hallita, missä kaikkialla salaisuuksia on, ketkä kaikki pääsevät niihin käsiksi, koska ne on lisätty tai poistettu tai ovatko ne muuttuneet. (Lundberg 2019b.)

Inhimillisten virheiden mahdollisuus sekä erilaiset vaatimukset liittyen salaisten tietojen käsittelyyn tulee myös ottaa huomioon salaisuuksien hallintaan liittyviä riskejä kartoittaessa. Ihmiset ovat erehtyväisiä, ja voivat esimerkiksi jakaa salaisuuden vahingossa väärälle henkilölle tai konfiguroida salaisuudenhallintatyökalun virheellisesti. Näillä virheillä voi olla vakavia seurauksia. Lisäksi arkaluonteisia tietoja käsitteleviin organisaatioihin saatetaan soveltaa erilaisia säännösten noudattamista koskevia määräyksiä, mukaan lukien GDPR (General Data Protection Regulation). Salaisten tietojen käsittely ilman asianmukaista hallintaa voi johtaa näiden säännösten rikkomuksiin. Salaisuuksien epäpätevä hallinnointi ja kierrättäminen voi myös johtaa neljäntenä lueteltuun riskiin, eli käyttökatkoihin palveluissa tai sovelluksissa. (Chuvakin & Viswanathan 2023.)

Salaisuuksien hallinta kehitysympäristöissä voi olla haasteellista. Kehittäjien näkökulmasta voi olla käytännöllistä tallettaa salaisuuksia ja salasanoja sisältäviä konfiguraatiotiedostoja kehitetyn koodin mukana. Syynä tähän on monesti tehokkuus ja helpompi ja nopeampi yhteistyö muiden kehittäjien kanssa. (Hospelhorn 2023.) Jos kehittäjällä ei ole käytössään salaisuuksien hallintaratkaisua, voi kehittäjä myös tästä syystä joutua jakamaan salaisuuksia kehitystiimin kesken manuaalisia prosesseja käyttäen. Tämä johtaa tyypillisesti

puutteelliseen dokumentointiin ja salaisuuksien viemiseen suojaamattomiin ympäristöihin. Toisena haasteena voi olla salaisuuksien hallintajärjestelmän keskitetyn hallinnoinnin puute. Mikäli kehitystiimejä on useita, ja jokainen tiimi dokumentoi, hallitsee ja säilyttää salaisuuksia eri tavoin, yhteinen tekeminen ja turvallinen salaisuuksien jakaminen voi olla haastavaa ja salaisuuksia löytyy useasta eri hallintajärjestelmästä tai muusta sijainnista. (Cloudflare 2024.)

Yhden haasteen salaisuuksien hallintaan tuovat kolmannen osapuolen tilit ja etäyhteysratkaisut. Organisaation ulkopuoliset tahot saattavat tarvita pääsyn sisäisiin tietokantoihin tai sovelluksiin. Salaisuuksien turvallisen hallinnan toteuttaminen voi olla haastavaa käyttäjien ollessa organisaation ulkopuolella. (Cloudflare 2024.)

2.2.2 Salaisuuksien hallinnan perusperiaatteet

Ensimmäinen askel organisaation salaisuuksien turvalliseen hallintaan on koota ja tallentaa kaikki salaisuudet keskitettyyn salaisuuksien hallintajärjestelmään (Johnson 2024). Tällöin salaisuudet ovat turvallisessa ympäristössä erillään sovellusten koodista ja kehittäjät välttävät todennäköisemmin salaisuuksien tallentamisen selkotekstinä eri lokaatioihin (Cloudflare 2024). Keskittämällä salaisuudet vältetään myös salaisuuksien hallitsematonta leviämistä (secrets sprawl) (Lundberg 2019b). Salaisuuksien hallintaratkaisun keskittäminen ei kuitenkaan aina tarkoita sitä, että organisaatiolla on vain yksi järjestelmä salaisuuksien hallintaan. Keskittäminen voi tarkoittaa myös useiden salaisuudenhallintaratkaisujen käyttämistä, mikäli se on perusteltua käyttötapausten kannalta. Tällöin tulee pitää huolta, että tiimit pitävät yllä vuorovaikutusta eri järjestelmien välillä. Vaikka organisaatio keskittäisi kaikki salaisuudet yhteen järjestelmään, ongelmaksi muodostuu usein kyseisen järjestelmän pää- tai hallintatunnukset, jotka on myös talletettava johonkin turvalliseen paikkaan. (OWASP 2024.)

Keskittämisen lisäksi on tärkeää, että organisaation sisällä on selvää, mihin salaisuutta käytetään ja mistä se löytyy. Käytännössä tämä varmistetaan vakiinnuttamalla ja yhtenäistämällä salaisuuksien hallintaan liittyvät toimintatavat

ja prosessit. Yhtenäisten toimintatapojen tulisi sisältää asioita liittyen salaisuuksien elinkaaren hallintaan, todennukseen, valtuutukseen ja kirjanpitoon. (OWASP 2024.)

Salaisuuksien hallintajärjestelmän on tärkeää olla tarpeeksi suorituskykyinen ja hyvin saatavilla. Mikäli järjestelmä ei tue korkeaa saatavuutta (high availability), kärsii sekä tiimien toimintakyky että salaisuuksista riippuvaisten sovellusten ja palveluiden toiminta. Pitkät odotusajat salaisuuksien hakemiselle voivat pidentää sovellusten käynnistymisaikoja ja hidastaa häiriötilanteissa palveluiden toimintaan palauttamista tiimien odotellessa pää- tai hallintatilien tunnuksia. (OWASP 2024.)

Auditointi on olennainen osa salaisuuksien hallintaa (OWASP 2024). Kaikkia käyttöoikeuksia tulee seurata ja pitää yllä kattavaa tarkastusprosessia (CyberArk 2024b). Vähintään tulisi tarkastella ja lokittaa kuviossa 1 luetellut tapahtumat. Auditoinnin kannalta on tärkeää, että kaikkiin tapahtumiin kirjataan lisäksi aikaleimat. Siksi salaisuudenhallintaratkaisussa tulee olla asianmukaiset aikasynkronointiprotokollat, jotka varmistavat aikaleiman oikeellisuuden lokeissa. (OWASP 2024.)

KUVIO 1. Salaisuuksien hallintaratkaisussa auditoitavat tapahtumat (OWASP 2024).

Tärkeä periaate salaisuudenhallintajärjestelmän valtuutuksia konfiguroitaessa on vähimmäisten tarvittavien oikeuksien periaate (principle of least privilege, PoLP). Tässä konseptissa käyttäjälle annetaan vain työtehtäviensä suorittamiseen tarvittavat vähimmäisoikeudet. Samaa mallia voidaan soveltaa ihmisten lisäksi myös sovelluksiin, laitteisiin ja järjestelmiin. (CyberArk 2024a.) Käyttäjien pääsyoikeuksien ja valtuutuksien hallintaan voi käyttää apuna myös roolipohjaista käyttöoikeuksin hallintaa (role-based access control, RBAC), jolloin käyttäjille voidaan myöntää oikeuksia ennalta määriteltyjen roolien perusteella (CyberArk 2024b).

Jotta inhimillisten virheiden mahdollisuuden riski voidaan laskea mahdollisimman pieneksi, tulisi salaisuuksien hallintaan liittyviä prosesseja automatisoida. Automatisointimahdollisuuksista ovat esimerkiksi dynaamisten salaisuuksien käyttöönotto ja salaisuuksien automatisoitu kierrättäminen ja luominen. Staattisten salaisuuksien säännöllinen kierrättäminen kannattaa, jotta mikään tunnistetieto ei ole kauaa voimassa ja mahdollista pitkäaikaista pääsyä arkaluonteisiin tietoihin. Kun kierrättäminen on automatisoitu, se tulee tehtyä säännöllisesti ja virheiden mahdollisuus pienenee. (OWASP 2024.)

3 KERTAKÄYTTÖVIESTIT

3.1 Kertakäyttöviestin konsepti ja periaatteet

Kertakäyttöviesti on viesti tai tieto, joka on tarkoitettu avattavaksi ja tarkasteltavaksi vain kerran. Yleisiä esimerkkejä ovat kertakäyttöiset salasanat tai linkit. (Chopra 2024.) Lisäksi on runsaasti tarjolla erilaisia pikaviestisovelluksia, joiden avulla on mahdollista lähettää tietyn aikarajan sisällä katoavia tai avaamisen jälkeen tuhoutuvia viestejä (Skyda 2023a). Näistä monille tuttuja esimerkkejä ovat Snapchat, WhatsApp ja Telegram (Nield 2022). Kaikille kertakäyttöisille tiedoille yhteistä on periaate siitä, että tiedon näkee vain kerran ja sen jälkeen se tuhoutuu lopullisesti. Tyypillisesti tiedolle voi lisäksi asettaa aikarajan, jonka kuluessa se tuhoutuu, vaikka sitä ei olisi ehditty vielä avata. (Chopra 2024; Hasseltine 2024.)

Yleisimmät käyttötapaukset kertakäyttöisille viesteille ovat kirjautumistietojen jakaminen, arkaluontoisten henkilökohtaisten tietojen välittäminen tai luottamuksellisten tietojen lähettäminen organisaation sisällä (Onetime Secret n.d.; Chopra 2024). Kertakäyttöviestin elinkaari alkaa tyypillisesti viestin luomisella ja salaamisella. Jos kyseessä on kertakäyttöinen linkki, seuraavaksi luodaan linkki johon viesti upotetaan. Tämän jälkeen viesti tai linkki jaetaan vastaanottajalle, joissakin tapauksissa salauksen purkamiseen tarvittavan avaimen kanssa. Vastaanottaja klikkaa linkkiä tai avaa viestin ja tarvittaessa syöttää salausavaimen. Kun vastaanottaja lopulta sulkee viestin, sitä ei voi enää avata uudelleen. (Chopra 2024.)

Kertakäyttöviesteissä on omat etunsa, muttta myös riskinsä. Sen jälkeen kun viesti on lähetetty, lähettäjällä ei ole kontrollia siihen, kuka viestin lopulta näkee, kuinka viestin sisältöä käytetään ja miten se tallennetaan. Jos vastaanottaja kopioi ja tallentaa salaisuuden paikkaan, joka ei ole tietoturvallinen, kertakäyttöviestin käyttämisen tarkoitus menee hukkaan. Vaikka lähettäjä olisi määritellyt viestille aikarajan, jonka jälkeen se tuhoutuu, lähettäjä ei voi vaikuttaa siihen, kauanko vastaanottaja käyttää ja säilyttää salaisuutta jossain muussa paikassa. (Chopra 2024.) Nämä asiat tulisi ottaa huomioon kertakäyttöviestejä

käytettäessä, kertakäyttöviestipalveluja kehitettäessä sekä soveltuvia käyttötapauksia harkittaessa (Chopra 2024; Hasseltine 2024).

Tärkeä osa kertakäyttöviestien turvallisuudesta muodostuu loppukäyttäjän kouluttamisesta. Aiemmin luetellut riskit viestin vastaanottajan käyttäytymisestä sekä salaisuuksien vaarantumisesta viestin avaamisen ja kopioimisen jälkeen voidaan saada hallintaan selkeiden ohjeiden ja käytäntöjen laatimisella. Organisaatiossa tulisi olla yhtenäiset ohjeet arkaluontoisten tietojen käsittelyä varten ja tapaan, millä kertakäyttöviestejä käytetään. Ohjeiden ja käytäntöjen lisäksi käyttäjiä tulisi myös kouluttaa tietoturvalliseen työskentelytapaan ja lisätä tietoisuutta erilaisista riskeistä salaisuuksien käsittelyssä. (Lark Editorial Team 2024.)

3.2 Kertakäyttöisyyden tietoturvahyödyt

Kertakäyttöisyys tuo mukanaan monia tietoturvahyötyjä. Yksi isoimmista hyödyistä on omien tietojen ja digitaalisen jalanjäljen kontrolloiminen. Kun käyttäjä lähettää viestin palveluntarjoajan kautta tai tallentaa tietojaan johonkin järjestelmään, ei voi olla täysin varma, että kukaan muu ei lue tietoja. Lähetetty viesti jää palveluntarjoajan palvelimille, ja esimerkiksi tietomurron sattuessa viestit voivat päätyä vääriin käsiin. Toisekseen, palveluntarjoaja saattaa käyttää tietoja voiton tavoittelemiseen tai analyysitarkoituksiin. Kertakäyttöisyyden etuna on, että viestit eivät jää palvelimille vaan ne tuhoutuvat heti avaamisen jälkeen, ja niitä ei tallenneta sen pidemmäksi aikaa kuin mitä viestin lähettäjä on määritellyt. (Skyda 2023a.)

Digitaalinen jalanjälki tarkoittaa jälkeä kaikista tiedoista, jotka jäävät käyttäjän jokaisesta toiminnasta internetissä. Kuvassa 1 on havainnollistettu mistä kaikista tiedoista digitaalinen jalanjälki voi koostua. Käyttäjän toiminta esimerkiksi sosiaalisessa mediassa, verkkokaupoissa tai hakukoneissa tallentuu digitaalisten järjestelmien verkostoon ja muodostaa tietojäljen, käyttäjän digitaalisen jalanjäljen. (Skyda 2023b.) Eri palvelimille tallennetut viestit ovat suuri ja arkaluontoinen osa digitaalista jalanjälkeä. Kertakäyttöviestejä hyödyntämällä käyttäjä voi pienentää omaa digitaalista jalanjälkeään, kun viestit

eivät tallennu eri palveluntarjoajien palvelimille. Etenkin jos kertakäyttöviestipalvelun avulla voi lisäksi estää vastaanottajia tallentamasta tai lähettämästä tietoja eteenpäin, arkaluonteisten tietojen hallinta ja valvonta on helpompaa. (Skyda 2023a.)

KUVA 1. Digitaalinen jalanjälki (Skyda 2023b).

Viestien kertakäyttöisyys yhdistettynä viestien salaukseen vähentää myös toistohyökkäysten (replay attacks) riskiä (Wickr 2020; Chopra 2024). Toistohyökkäyksissä hyökkääjä sieppaa ja lähettää uudelleen tietoja, jotka on aiemmin vaihdettu kahden osapuolen välillä. Esimerkiksi käyttäjän tehdessä palvelimelle pyynnön, hyökkääjä kuuntelee ja varastaa nämä tiedot. Myöhemmin hyökkääjä lähettää samat tiedot uudelleen palvelimelle toistaakseen aiemman tapahtuman ja päästäkseen näin esimerkiksi kirjautumaan palvelimelle. (Okta 2024.) Salatun kertakäyttöviestin tapauksessa hyökkääjä ei saa selville salaisuutta salauksen ansiosta ja kertakäyttöisyyden vuoksi ei voi toistaa pyyntöä (Chopra 2024; Okta 2024).

Väliintulohyökkäyksessä (man-in-the-middle attack) hyökkääjä sieppaa ja mahdollisesti muuttaa kahden osapuolen välistä viestintää osapuolten tietämättä.

Hyökkääjän tavoitteena on salakuunnella, muuttaa viestiä tai syöttää mukaan haitallista sisältöä. (James 2023.) Salauksen ansiosta väliintulohyökkäysten vaikutukset voivat jäädä pieniksi, sillä hyökkääjä ei pääse muokkaamaan tai lukemaan salaista viestiä (Chopra 2024). Kertakäyttöviestin lyhyt olemassaoloaika myös pienentää osaltaan hyökkääjän mahdollisuuksia hyödyntää kaappaamaansa viestiä (Wickr 2020).

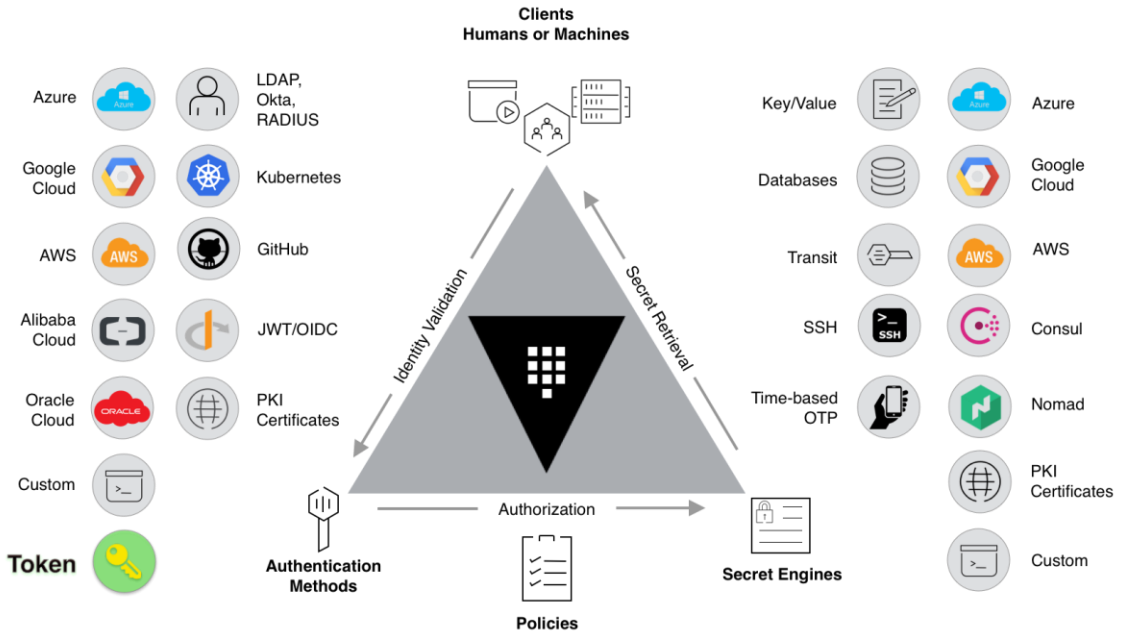
4 HASHICORP VAULT APUNA SALAISUUKSIEN HALLINNASSA

HashiCorp Vault toimii isossa roolissa tämän työn yhteydessä rakennettavassa kertakäyttöviestipalvelussa. Tässä kappaleessa tutustutaan ensin siihen, mikä HashiCorp Vault on, mitkä ovat sen päätoiminnallisuudet, ja mistä eri komponenteista se rakentuu. Tämän jälkeen käydään läpi Vaultin Response Wrapping -ominaisuutta, joka toimii pohjana kertakäyttöviestipalvelulle.

4.1 Mikä on HashiCorp Vault?

HashiCorp Vault on identiteettipohjainen salaisuuksien ja salauksen hallintajärjestelmä (HashiCorp n.d.-b). Vaultin avulla voidaan suojata, tallentaa ja valvoa pääsyä salaisiin tietoihin, kuten salasanoihin, varmenteisiin tai tunnisteisiin. Vault-alusta muodostaa dynaamisen infrastruktuurin, joka on laajennettavissa erilaisilla lisäosilla. (Krausen 2019.) Salaisuuksia ja käyttöoikeuksia voi hallita hyödyntämällä ohjelmointirajapintoja (API) tai käyttäjäystävällistä käyttöliittymää (UI) (Sabharwal, Pandey & Pandey 2021). Näiden kahden lisäksi Vaultia on mahdollista käyttää myös komentorivin (CLI) kautta (McTeer & Krausen 2020, 14).

Kuvassa 2 on esillä Vault Triangle, joka esittelee Vaultin tärkeimmät toiminnallisuudet yhdessä tilannekuvassa. Kuvan vasemmalla puolella on luettelo erilaisia tunnistautumismenetelmiä, jotka on mahdollista ottaa käyttöön Vaultissa ja oikealla puolella taas vastaava lista Secrets Engineistä. Näistä Vaultin eri komponenteista kerrotaan tarkemmin tulevissa alaluvuissa. Kuvassa on myös visualisoitu Vaultin tyypillinen tietovirta liittyen tunnistatutumiseen, valtuutukseen ja salaisuuksien noutamiseen. (Lundberg 2019a.)



KUVA 2. Vault Triangle: Vaultin toiminnallisuudet ja tietovirta (HashiCorp n.d.-a).

Vaultista on saatavilla erilaisia versioita. Itse hallinnoitavia ja ylläpidettäviä versioita ovat avoimen lähdekoodin (Open Source) versio sekä yritysversio (Enterprise). Avoimen lähdekoodin versio on käytettävissä ilmaiseksi, yritysversio maksusta. Yritysversion lisäominaisuuksia ovat esimerkiksi katastrofivalmius (disaster recovery), pääsy kaikkiin todennusmenetelmiin, nimialueet (namespaces), replikointimahdollisuus (replication) muihin Vault-klustereihin (cluster) eli järjestelmiin eri datakeskuksissa tai pilvipalveluissa sekä monivaiheinen tunnistautuminen (multi-factor authentication, MFA). Mikäli hallinnoinnin ja ylläpidon haluaa siirtää HashiCorpille, on mahdollista tilata HashiCorp Cloud (HCP) -versio Vaultista. Tällöin maksu Vaultin käytöstä määräytyy tunneittain ja myös kaikki yritysversion ominaisuudet ovat käytettävissä. (Krausen 2019; HashiCorp n.d.-c.)

4.2 Vaultin tärkeimmät ominaisuudet

Vaultin tärkeimpiin ominaisuuksiin lukeutuvat salauspalvelut (Encryption as a Service), pyynnöstä luotavat salaisuudet (on-demand secrets) sekä salaisuuksien ja käyttöoikeuksien hallinta, kuten uusiminen (renewal) ja peruuttaminen (revocation) (Krausen 2019; Sabharwal ym. 2021). Vaultin avulla

on mahdollista salata ja purkaa tietoja niitä tallentamatta. Datan voi säilöä esimerkiksi SQL-tietokannan kaltaisessa paikassa ja salausparametrit on mahdollista määritellä itse. Salauspalvelu mahdollisttaa kehittäjille datan salaamisen ilman omien salausmenetelmien rakentamista. (HashiCorp n.d.-b.)

Salaisuuksia on mahdollista luoda pyynnöstä moniin palvelinpuolen järjestelmiin, kuten tietokantoihin tai pilvipalveluihin (Chapman 2024). Vault pystyy käsittelemään dynaamisia salaisuuksia, jotka luodaan tarpeen mukaan. Dynaamiset salaisuudet ovat paljon turvallisempia kuin staattiset, ennalta määritellyt ja jaetut salaisuudet. (Sabharwal ym. 2021.) Dynaamisten salaisuuksien etuna on niiden rajattu olemassaoloaika. Lyhentämällä salaisuuksien olemassaolon aikaa pienennetään riskiä niiden vaarantumiselle. (Chapman 2024.) Pyynnöstä luotu, dynaaminen salaisuus peruutetaan (revoke) automaattisesti kun sen sopimus (lease) päättyy. Sopimuksen keston määrittelee ennalta asetettu voimassaoloaika eli TTL, Time to Live. (Sabharwal ym. 2021.) Yksittäisten salaisuuksien lisäksi myös useiden salaisuuksien peruuttaminen kerralla on mahdollista. Sopimus voidaan myös uusia (renew) ennen sen päättymistä. (HashiCorp n.d.-b.)

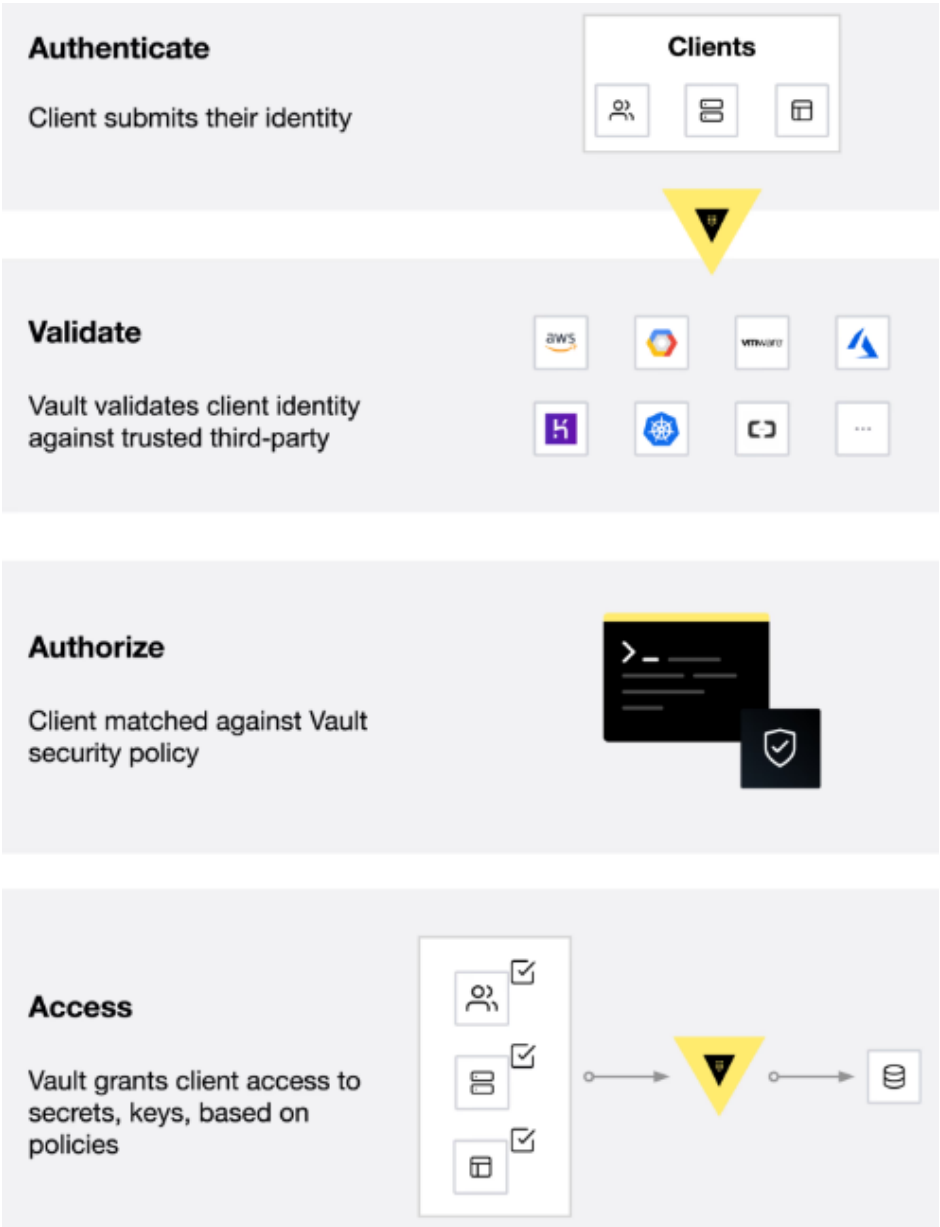
Salaisuuksien hallinta on yksi Vaultin tärkeimmistä ominaisuuksista. Vaultiin voi tallentaa mitä vain salaista dataa, kuten salasanoja tai avaimia. (Sabharwal ym. 2021.) Vault salaa kaiken datan ennen sen tallentamista, jolloin tallennetun datan käsiin saaminen ei vielä paljasta tallennettuja salaisuuksia. Dataa on mahdollista tallentaa useampaan paikkaan, esimerkkeinä tallentaminen levylle tai HashiCorp Consuliin. (HashiCorp n.d.-b.)

4.3 Vaultin toimintaprosessi

Vaultin toiminta kulkee kuvan 3 mukaisen prosessin mukaan. Ensin käyttäjä tunnistautuu tarjoamalla käyttäjätietoja. Sen jälkeen Vault validoi käyttäjän kolmannen osapuolen, kuten GitHubin tai Google Cloudin avulla. Mikäli tunnistautumisprosessi onnistuu, käyttäjälle luodaan token, eräänlainen tunnisteväline, johon voidaan myös liittää tietoja käyttäjän oikeuksista Vaultissa.

(HashiCorp n.d.-b.) Tunnistautumisprosessia ja eri todennusmenetelmiä kuvataan kattavammin myöhemmin.

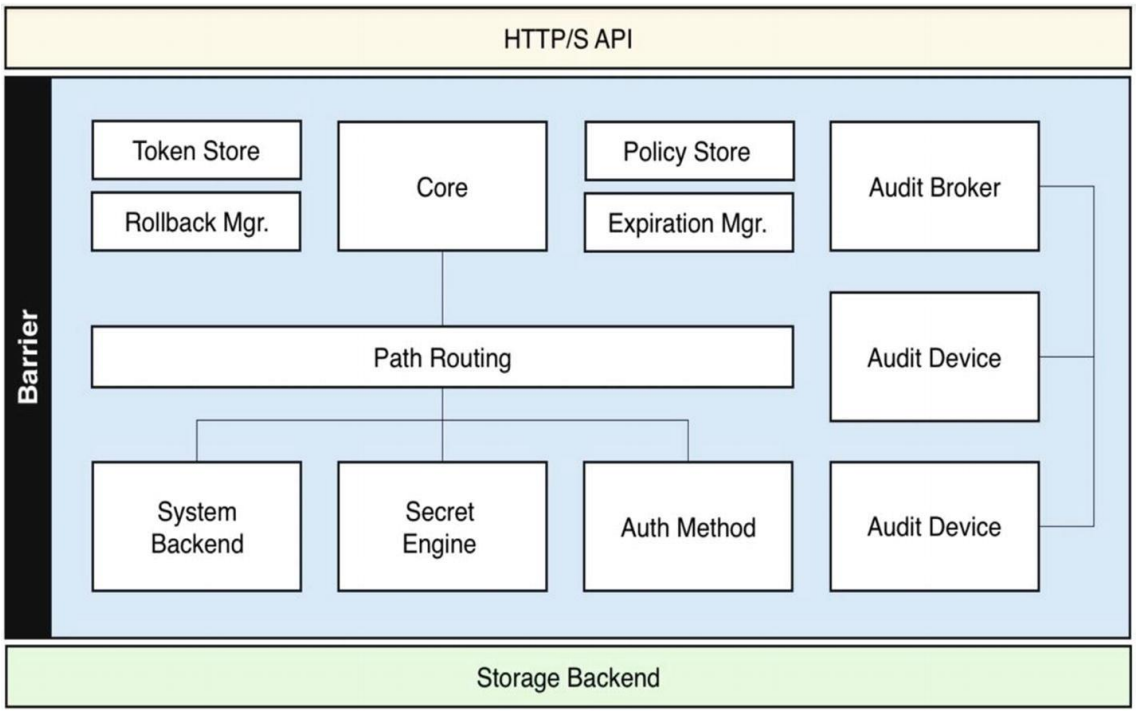
Kun käyttäjä on tunnistettu ja tunnistus vahvistettu kolmannen osapuolen lähdettä käyttäen, käyttäjän tokenille lisätään omat turvallisuuskäytännöt eli policyt. Turvallisuuskäytäntö sisältää joukon sääntöjä, jotka määrittelevät, mihin kaikkiin API-päätepisteisiin käyttäjän on mahdollista päästä. (HashiCorp n.d.-b.) Vault on rakennettu turvallisuuden vuoksi siten, että oletuksena käyttäjällä on minimaaliset oikeudet. Kaikki mihin ei ole erikseen annettu oikeuksia, on oletuksena kielletty. (Myers 2023, 19.) Mikäli käyttäjältä löytyy pyyntöä vastaava oikeus, käyttäjä saa pääsyn kohteeseen (HashiCorp n.d.-b).



KUVA 3. Vault työnkulkukaavio (HashiCorp n.d.-b).

4.4 Vaultin arkkitehtuuri

Vaultin looginen arkkitehtuuri muodostuu kuvan 4 mukaisesti (Sabharwal ym. 2021). Kuvassa ylhäällä on API, jonka kautta käyttäjä tai sovellus on yhteydessä Vaultiin. Alhaalla on tallennusjärjestelmä (Storage Backend), johon data tallennetaan. Keskellä ovat Vaultin komponentit, joita ympäröi kryptografinen este (barrier). Suojaavan esteen läpi pääsee vain autentikoitu käyttäjä, jolla on tarvittavat oikeudet. (Krausen 2019.)



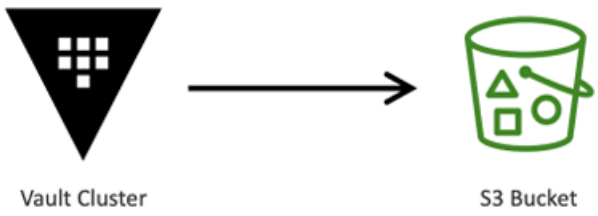
KUVA 4. Vaultin arkkitehtuuri (Sabharwal ym. 2021).

4.4.1 Tallennusjärjestelmät

Kryptatun datan tallennuspaikkana toimii palvelinpuolen tallennusjärjestelmä, Storage Backend (Young 2023, 27). Tallennusjärjestelmä määritellään halutuilla parametreilla Vaultin konfiguraatiotiedostossa (Krausen 2019). Tällä hetkellä Vault on yhteensopiva yli kahdenkymmenen eri tallennusjärjestelmän kanssa. Osa järjestelmistä on HashiCorpin tukemia ja ylläpitämiä, kuten Consul, Filesystem, In-Memory ja Integrated Storage. Näistä vain Consul ja Integrated

Storage ovat sopivia käytettäväksi tuotannossa, Filesystem ja In-Memory sopivat vain paikalliseen testaamiseen. Jos esimerkiksi In-Memory-järjestelmää käytettäisiin tallennusjärjestelmänä tuotannossa pyörivässä Vaultissa, kaikki data menetettäisiin päivityksen tai uudelleenkäynnistyksen yhteydessä. (Young 2023, 27–28.) Tällä hetkellä HashiCorp suosittelee Integrated Storagen käyttöä tallennusjärjestelmänä useimmille käyttötapauksille. Se on käytettävissä Vault 1.4-versiosta alkaen. Vanhemmille Vault-versioille HashiCorp suosittelee Consulin käyttöä. (HashiCorp n.d.-b.) Tallennusjärjestelmänä on mahdollista käyttää myös muiden organisaatioiden ylläpitämiä ratkaisuja, kuten Amazon Web Servicen (AWS) ylläpitämää DynamoDB:tä (Young 2023, 27). Muut tallennusjärjestelmävaihtoehdot vaihtelevat relaatiotietokannoista ja NoSQL-tietokannoista pilvipohjaisiin ratkaisuihin (Soto Bueno & Block 2023).

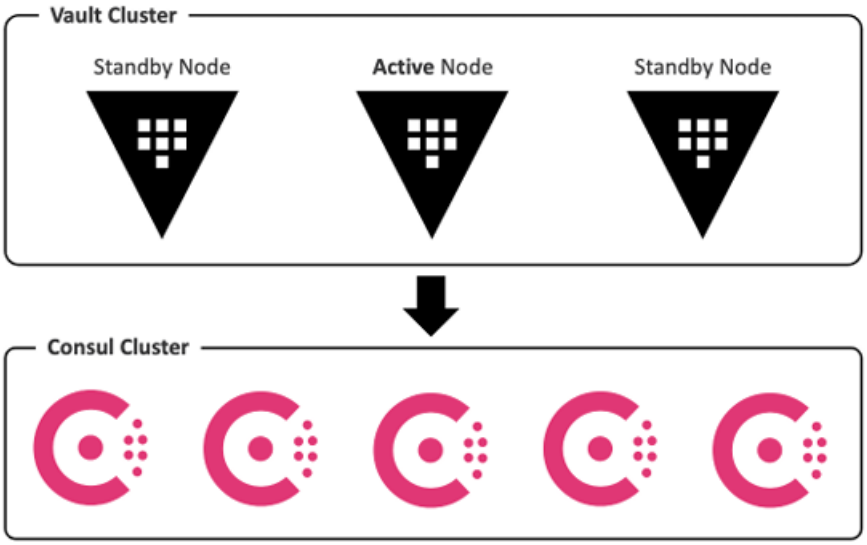
Sopivaa tallennusjärjestelmää valitessa on hyvä ottaa huomioon järjestelmän ylläpitäjän lisäksi myös sen saatavuus (Young 2023, 27). Osa tallennusjärjestelmistä tukee korkeaa saatavuutta (high availability, HA) ja osalla taas saattaa olla paremmat työkalut tietosuojaa ja hallinnointia varten (Krausen 2019). Mikäli järjestelmä ei tue korkeaa saatavuutta, vain yksi Vault-tietosolu (node) voi kirjoittaa, lukea ja käyttää dataa tästä järjestelmästä. Tämä tilanne on esitetty kuvassa 5, jossa Vault-klusteri eli tietosolujen muodostama järjestelmä sisältää vain yhden kyseiseen operaatioon käytettävän Vault-tietosolun. Tällöin esimerkiksi päivityksistä aiheutuu palvelukatkosia, ja järjestelmä voidaan palauttaa käyttöön vasta kun tämä yksittäinen tietosolu on jälleen käytössä. (Young 2023, 28.)



KUVA 5. Yksittäinen Vault-tietosolu (Young 2023, 28).

Kuva 6 demonstroi korkean saatavuuden järjestelmän toimintaa. Esimerkkinä on käytetty HashiCorp Consulia. Korkean saatavuuden tuki tarkoittaa, että useat

tietosolut voivat käyttää järjestelmää, jolloin käyttökatkosten vaikutus minimoidaan. Kuvan esimerkissä on käytössä kolme Vault-tietosolua joista yksi on aktiivinen ja kaksi toimii varalla. (Young 2023, 29.)



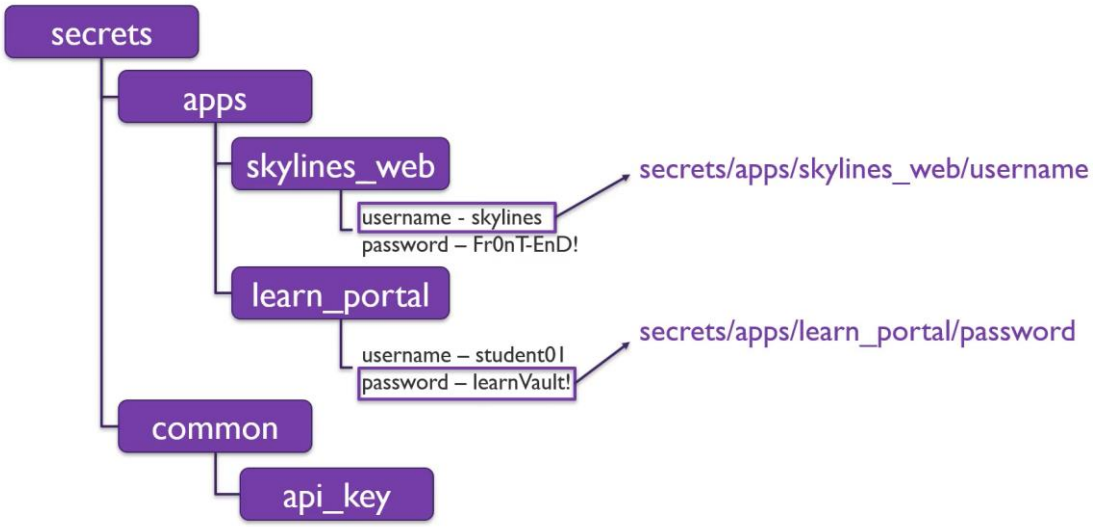
KUVA 6. Esimerkki korkean saatavuuden järjestelmästä (Young 2023, 29).

4.4.2 Polut

Polut (paths) ovat Vaultin rakennuspalikoita. Kaikki toiminta Vaultissa perustuu polkuihin ja jokainen päätepiste on saavutettavissa polkujen kautta. Esimerkkejä polkujen kautta saavutettavista kohteista ovat Secrets Enginet, tunnistautumismenetelmät, staattiset salaisuudet sekä turvallisuuskäytännöt eli policyt. Kun jokin komponentti on otettu käyttöön Vaultissa, sen kanssa ollaan yhteydessä polkujen kautta. (McTeer & Krausen 2020, 15.) Secrets Enginet ja tunnistautumismenetelmät otetaan käyttöön niille ennaltamääritellyn polun avulla. Käyttöönoton yhteydessä joidenkin polkujen nimiä on myös mahdollista muuttaa mieleisekseen. Osalla komponenteista on valmiina oletuspolkuja myös käyttöönottopolun alapuolella. Esimerkiksi Database Secrets Enginellä on valmiiksi käytössä polut config, roles ja creds. (Krausen 2019.)

Kuvassa 7 esitellään polkujen muodostumista Vaultissa. Esimerkkinä käytetään Vaultin Key/Value (KV) Secrets Engineä, jolle on valittu nimeksi secrets. Polun etuliite (prefix) kertoo, mille Vault-komponentille pyyntö tulee ohjata. Kuvan

esimerkin tapauksessa pyyntö ohjataan KV Secrets Enginelle, eli polun etuliitteelle secrets. Ylimmän tason alta löytyy polku apps, jonka alle voidaan lisätä eri sovelluksia, joiden salaisuuksia halutaan tallentaa. Kuvan esimerkissä sovelluksen skylines_web alle on talletettu käyttäjätunnus ja salasana. Mikäli siis haluttaisiin päästä lukemaan skylines_webin käyttäjätunnus, tulisi käyttää polkua secrets/apps/skylines_web/username. Ylimmän tason alle on määritelty myös polku common, jonka alle voidaan tallentaa yleisiä, tiettyyn sovellukseen liittymättömiä salaisuuksia, kuten API-avaimia. (Krausen 2019.)



KUVA 7. Esimerkki Vaultin poluista (Krausen 2019).

4.4.3 Secrets Enginet

Secrets Enginet ovat Vaultin komponentteja, joiden avulla luodaan, tallennetaan ja salataan dataa (HashiCorp n.d.-a). Ilman Secrets Enginejä Vaultin käyttöönotto ei olisi järkevää, sillä ne tarjoavat Vaultin ydintoiminnot. Kaikkia muita Vaultin komponentteja voidaan pitää Secrets Enginejä tukevina komponentteina. Joka Secrets Enginellä on omanlaisensa erityisominaisuudet. Osa luo lyhytikäisiä, dynaamisia salaisuuksia, toiset taas keskittyvät staattisen, pitkäikäisen datan tallentamiseen. Osa Secrets Engineistä voi myös salata selkotekstidataa kuljetuksen aikana. (McTeer & Krausen 2020, 15.)

Secrets Enginen kanssa kommunikointi sekä käyttöönotto tapahtuu polkujen (paths) avulla. Kommunikointi tarvitsee onnistuakseen käyttäjältä tarvittavat

oikeudet, jotka tarkistetaan pyynnön mukana kulkevan tokenin avulla. (McTeer & Krausen 2020, 140.) Oikeuksiin ja tokeneihin liittyvää tietoa käydään lävitse tarkemmin seuraavassa alaluvussa.

Koska Secrets Enginet muodostavat Vaultin päätoiminnot, niitä on käyttövalmiina laaja valikoima. Tyypillisesti organisaatiot käyttävät montaa erilaista Secrets Engineä ottaessaan Vaultin käyttöönsä. Samaa Secrets Engineä voi myös käyttää eri tavoin eri tilanteissa, käyttötapauksen mukaisesti. (McTeer & Krausen 2020, 141.) Secrets Enginejä voi ajatella myös lisäosina (plugin), joista voi ottaa käyttöön ne, jotka sopivat parhaiten käyttäjän tarkoituksiin ja tarpeisiin (HashiCorp n.d.-a). Vaikka uusia Secrets Enginejä lisätään Vaultiin ajan mittaan, niiden käyttötapa säilyy melko samanlaisena. Jo aiemmin esitellyn kuvan 2 Vault Trianglessa oli nähtävillä luettelo kuvan laatimishetken Secrets Engineistä. (Lundberg 2019a.) Vaultilla on useita sisäänrakennettuja Secrets Enginejä, joiden lisäksi on mahdollista ottaa käyttöön myös ulkopuolisten palveluntarjoajien Secrets Enginejä (Prowse 2024). Osa näistä on pilvipalvelupohjaisia (Cloud Provider) Secrets Enginejä, kuten AWS, Google Cloud tai Azure (McTeer & Krausen 2020, 165).

Suosituimpia Secrets Enginejä ovat Key/Value, Transit, PKI, Database ja pilvipalvelupohjaiset Secrets Enginet. Key/Value (KV) Secrets Enginen avulla voi nimensä mukaisesti tallentaa salaisuuksia yleisessä avain-arvo-muodossa. (McTeer & Krausen 2020, 141.) KV Secrets Enginestä on olemassa kaksi versiota, versio 1 ja versio 2. Versio 1 on yksinkertaisempi ja versio 2 sisältää useampia lisäominaisuuksia, esimerkkinä versiointi ja tietojen palauttaminen. Kun tallennetun avain-arvo-parin dataa muutetaan, versioinnin avulla on mahdollista päästä käsiksi aiempiin versioihin datasta. Ilman versiointia vain viimeisin avaimeen kirjoitettu arvo tallennetaan. Molemmille versioille KV Secrets Enginestä on omat käyttötapauksena. Monet organisaatiot ottavat käyttöönä versio 2:n sen laajempien ominaisuuksien vuoksi, mutta versio 1 voi olla parempi vaihtoehto, jos lisäominaisuuksille ei ole tarvetta. Versio 1 vaatii vähemmän tallennustilaa ja vähemmän luku- ja kirjoitustapahtumia, sillä tallennettavana ei ole ylimääräistä metadataa, jota versiossa 2 tallennetaan. (HashiCorp n.d.-b; McTeer & Krausen 2020, 141–142.)

Transit Secrets Engine mahdollistaa salauspalveluiden (Encryption as a Service) tarjoamisen Vaultissa. Ottamalla Transit Secrets Enginen käyttöön, sovellukset voivat lähettää selkotekstidataa Vaultiin salattavaksi. Vault ei tallenna Transit Secrets Enginelle lähetettyä dataa, vaan sovellus voi tallentaa salatut tiedot haluamaansa paikkaan. Kun tietoja taas tarvitaan ja sovellus hakee dataa tallennuspaikasta, data voidaan lähettää Vaultiin purettavaksi. Datan salaamiseen ja salauksen purkamiseen käytetään samaa, prosessin alussa luotua salausavainta. Salausavaimia voidaan helposti kierrättää ja salattu data salataan uudella avaimella uudelleen. Salausavainta luotaessa käyttäjä valitsee avaintyypin Vaultin tukemista vaihtoehdoista. Esimerkkejä avaintyypeistä ovat RSA, AES-GCM ja ECDSA. (McTeer & Krausen 2020, 159–161.) Salauspalvelun käyttöönottaminen helpottaa sovelluskehittäjien työtaakkaa vapauttamalla kehittäjät asianmukaisen salauksen ja salauksen purkamisen rakentamiselta (HashiCorp n.d.-b). Toisena etuna Vaultin salauspalvelun käytössä on alustan tai tallennuspaikan erottaminen tietojen salauksesta. Tietojen vaarantumisen mahdollisuus on entistä pienempi, kun salausavainta ja tietoja ei säilytetä samassa kohteessa. (McTeer & Krausen 2020, 160.) Salauspalvelun lisäksi Transit Secrets Engine voi muun muassa varmentaa ja allekirjoittaa tietoja, luoda tiivisteitä (hash) ja toimia lähteenä satunnaisille tavuille (bytes) (HashiCorp n.d.-b).

PKI Secrets Engine luo dynaamisia X.509-varmenteita (HashiCorp n.d.-b). X.509-varmenteet ovat digitaalisia dokumentteja, jotka edustavat tietokonetta, käyttäjää tai palvelua (Microsoft Learn 2023). X.509-varmenne käyttää julkisen avaimen infrastruktuuristandardia (public key infrastructure, PKI) todentaakseen julkisen avaimen kuuluvan varmenteeseen sisältyvälle käyttäjälle, palvelun tunnisteelle tai tietokoneelle. Jokainen X.509-varmenne sisältää allekirjoitusosan sekä dataosan, jossa on tietoa muun muassa varmenteen kohteesta, varmenteen myöntäjästä ja kohteen julkisesta avaimesta. (Jean-Mary 2020; Microsoft Learn 2023.) Käyttämällä Vaultin PKI Secrets Engineä, palvelut voivat hankkia varmenteita käymättä läpi tavanomaista manuaalista prosessia. Vaultin sisäänrakennetut valtuutus- ja todennusmekanismit tarjoavat todentamistoiminnon. (HashiCorp n.d.-b.) Tämä lyhentää huomattavasti varmenteiden luomiseen menevää aikaa. Lisäksi PKI Secrets Engine mahdollistaa varmenteiden automaattisen uusimisen, jolloin varmenteen

vanhentumisesta johtuvia katkoksia ei pääse syntymään. (McTeer & Krausen 2020, 172–173).

Database Secrets Enginen avulla voidaan luoda dynaamisia tietokantatunnuksia määritettyjen roolien perusteella (HashiCorp n.d.-b). Tuettuna on laaja valikoima tietokantoja. Database Secrets Enginen käytön etuna on, että sovellusten ei tarvitse kovakoodata pitkäikäistä palvelutiliä tietokantaan pääsyä varten. Sovellus voi luoda tarpeen mukaan tunnuksia tietokantaan kun pääsyä tarvitaan ja Vault poistaa tunnukset kun niille määritelty TTL on tullut täyteen. Vaultin tulee tietää, mihin tietokantaan tunnukset luodaan sekä mitkä oikeudet käyttäjälle myönnetään. Tämä kartoitus tehdään roolien avulla, joita voi olla yksi tai useampia riippuen käyttäjien tai sovelluksen tarpeista. Jokainen rooli voidaan konfiguroida antamaan hyvin erilaisia oikeuksia tietokantaan, jotta jokaisella tunnuksella olisi vain vähimmäiset oikeudet joita toimintaan tarvitaan. (McTeer & Krausen 2020, 155.)

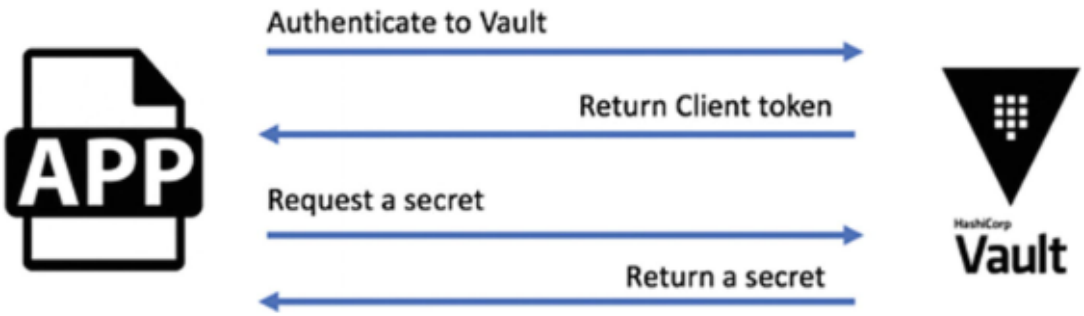
Vault tukee monia suuria pilvialustoja, kuten AWS, AliCloud, Google Cloud ja Azure. Koska organisaatiot ovat siirtäneet monia sovelluksia pilveen, mahdollisuus integroida Vault julkisen pilvipalveluntarjoajan kanssa on tarpeellista. Vaultin avulla voidaan luoda tarpeen mukaan pääsy pilvipalveluun Secrets Enginen tavoin sen sijaan, että julkiseen pilvipalveluun luotaisiin pitkäikäinen tili. (McTeer & Krausen 2020, 165.) Näiden sekä aiemmin lueteltujen lisäksi Vaultilla on tarjolla myös useita muita Secrets Enginejä, ja niitä lisätään tyypillisesti ajan myötä lisää (Lundberg 2019a).

4.4.4 Tunnistautumis- ja valtuutusmenetelmät

Tunnistautumismenetelmät (authentication methods) ovat komponentteja, jotka suorittavat todennuksen käyttäjän tai sovelluksen identiteetistä Vaultille (Krausen 2019). Todennuspyynnöt voivat tulla esimerkiksi sovellukselta, joka tekee API-pyynnön tai käyttäjältä, joka kirjautuu käyttöliittymään. Pyynnön alkuperästä huolimatta, Vaultin tietoihin pääsy ei ole mahdollista ilman asianmukaista todennuspyyntöä tai jo voimassa olevaa tokenia. (McTeer & Krausen 2020, 118.) Token on asiakastunniste, joka käsitteellisesti vastaa verkkosivujen

istuntoevästettä. Kun käyttäjä on todentanut itsensä, Vault palauttaa tokenin, jota käytetään tulevissa pyynnöissä. Tokenin avulla Vault tarkistaa asiakkaan henkilöllisyyden ja tokeniin liitettyjen turvallisuuskäytäntöjen avulla määrittää, mihin kaikkeen käyttäjällä on oikeus päästä käsiksi. (HashiCorp n.d.-b.) Toisin sanoen, Vaultissa tunnistautumismenetelmät tarjoavat todennuksen (authentication) ja tokenit, joihin on liitetty turvallisuuskäytäntöjä (policies), tarjoavat valtuutuksen (authorization) (McTeer & Krausen 2020, 118).

Kuvassa 8 on nähtävillä esimerkki sovelluksen pyynnöstä hakea salaisuus Vaultista. Sovelluksen todennettua itsensä, Vault palauttaa satunnaisesti generoidun tunnisteen eli tokenin. Tämän jälkeen Vault tarkistaa, että tokenilla on oikeudet päästä käsiksi pyydettyyn salaisuuteen ja oikeuksien löytyessä palauttaa sovellukselle pyydetyn datan. (Sabharwal ym. 2021.)



KUVA 8. Esimerkki Vaultiin tunnistautumisesta (Sabharwal ym. 2021).

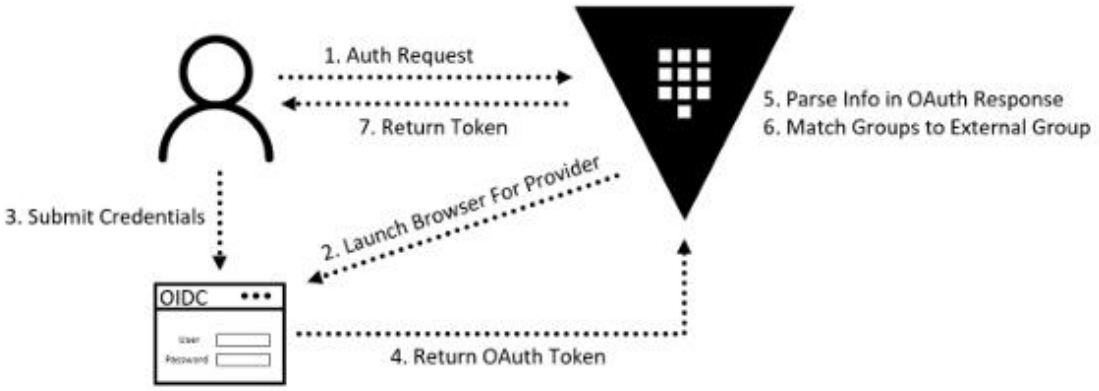
Vaultissa on mahdollista ottaa käyttöön monia erilaisia tunnistautumismenetelmiä (Krausen 2019). Organisaatiot tyypillisesti käyttävät useita eri tunnistautumismenetelmiä käyttötapausten tarpeiden mukaisesti. Käytössä on yleensä vähintään erilaiset tunnistautumismenetelmät käyttäjälle (ihminen) ja sovellukselle tai isäntäkoneelle (laite). (McTeer & Krausen 2020, 118.) Esimerkkejä mahdollisista tunnistautumismenetelmistä ovat lokaali käyttäjänimi ja salasana (User Pass), AppRole, AWS ja GitHub (Prowse 2024). Osa näistä menetelmistä on sisäänrakennettu Vaultiin, kuten AppRole ja User Pass, ja osa taas on ulkopuolisen palveluntarjoajan, kuten GitHub ja AWS (HashiCorp n.d.-b; Chapman 2024).

Käyttäjätunnus ja salasana -yhdistelmät (User Pass) asetetaan suoraan Vaultin tunnistautumismenetelmään users-polun avulla, eikä tunnuksia ja salasanoja voida lukea ulkoisesta lähteestä. AppRole-tunnistautumismenetelmä taas on soveltuvampi automatisoituihin, koneiden tai palvelujen välisiin todennuksiin. (HashiCorp n.d.-b.) AppRolen avulla voidaan käsitellä helposti suurta määrää sovelluksia. Kullekin sovellukselle voidaan luoda oma rooli, jolla on omat määritellyt turvallisuuskäytännöt, ja roolin avulla Vault sallii pääsyn vain sovelluksen tarvitsemiin kohteisiin. (McTeer & Krausen 2020, 119.)

Ulkopuolisten palveluntarjoajien tunnistautumismenetelmien käyttö mahdollistaa sovellusten tunnistautumisen Vaultiin erilaisin menetelmin ilman pitkäikäisiä tunnuksia tai AppRole-konfiguraatiota. Näitä valikoituja pilvipohjaisia identiteetinhallintajärjestelmiä kohdellaan Vaultin toimesta kolmannen osapuolen luotettuina palveluina. (McTeer & Krausen 2020, 127.) Esimerkkejä tarjolla olevista ulkopuolisten palveluntarjoajien tunnistautumismenetelmistä ovat AWS, Okta, LDAP, GitHub, Azure ja AliCloud (Chapman 2024). Kun valittu tunnistautumismenetelmä on otettu käyttöön Vaultissa, pilvipohjaiset yksiköt voidaan liittää turvallisuuskäytäntöön nimettyjen roolien avulla. Eri rooleja voidaan luoda sovellusvaatimusten perusteella ja yksittäisellä roolilla voi olla yksi tai useampia turvallisuuskäytäntöjä. (McTeer & Krausen 2020, 127.)

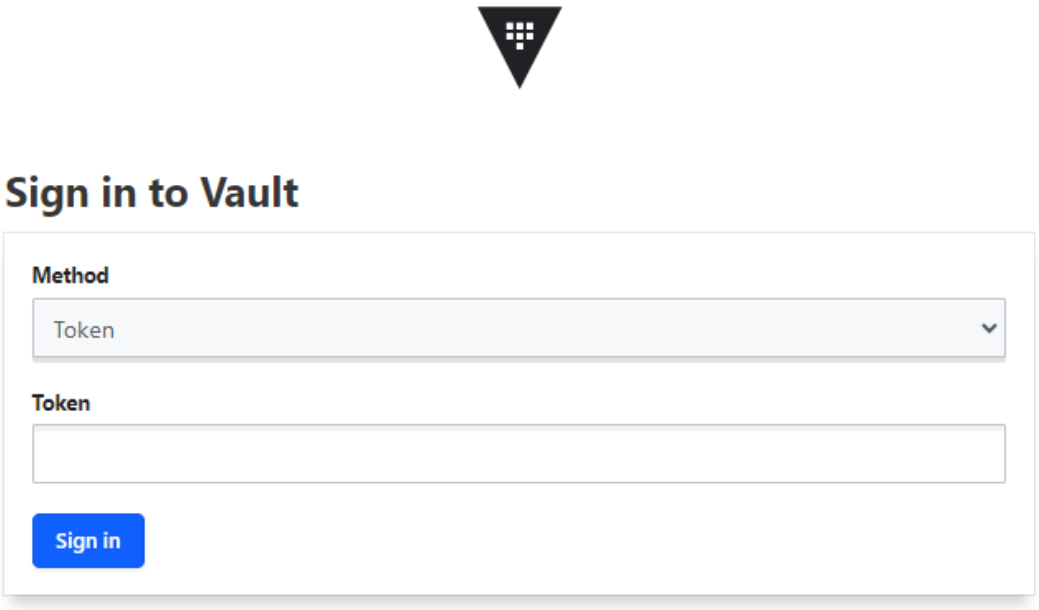
Myös OIDC (OpenID Connect) -tunnistautuminen on mahdollista ottaa käyttöön Vaultissa. Monet organisaatiot käyttävät IT-ympäristöissään todennukseen kertakirjautumisratkaisua (single sign-on). Vaultin OIDC-tunnistautumismenetelmän avulla organisaatiot voivat hyödyntää jo olemassa olevaa käyttäjähakemistoaan Vaultin palveluihin tunnistautumisessa. Kuvassa 9 käydään läpi tunnistautumista Vaultiin OIDC:n avulla. Ensin käyttäjä lähettää tunnistautumispyynnön OIDC-tunnistautumismenetelmän avulla. Seuraavaksi Vault avaa selaimen ja ohjaa käyttäjän konfiguroituun OIDC-palveluntarjoajan osoitteeseen ja käyttäjä täyttää tunnistetietonsa osoitteessa. Tämän jälkeen OIDC-palveluntarjoaja varmentaa tunnistetiedot ja palauttaa OAuth-tokenin Vaultille. Vault jäsentää tokenin tiedot ja vertaa OAuthin tarjoamia ryhmiä, joissa käyttäjä on jäsenenä, ennalta määriteltyihin ulkoisiin Vault-ryhmiin. Jos jokin ryhmistä tai useampi ryhmä täsmää, luodaan token, joka liitetään vastaavan ulkoisen ryhmän turvallisuuskäytäntöihin. Mikäli täsmääviä ryhmiä ei löydy,

luodaan token Vaultin oletusturvallisuuskäytännöllä. (McTeer & Krausen 2020, 132–133.)



KUVA 9. OIDC-tunnistautuminen Vaultiin (McTeer & Krausen 2020, 132).

Tokenit ovat keskeinen osa tunnistautumismenetelmiä. Lähes kaikki pyynnöt (requests) Vaultiin vaativat pyynnön mukaan liitetyn tokenin. (Prowse 2024.) Sen lisäksi, että Vaultiin voidaan tunnistautua erillisellä tunnistautumismenetelmällä, joka luo tokenin dynaamisesti, tokenia voidaan käyttää myös suoraan tunnistautumiseen. (HashiCorp n.d.-b.) Kuvassa 10 on nähtävillä Vaultin kirjautumiskäyttöliittymä (HashiCorp 2024). ”Method”-kentässä on oletuksena valittuna tokenin avulla kirjautuminen. Kun käyttäjä kirjoittaa ”Token”-kenttään voimassa olevan tokenin, käyttäjä tunnistautuu Vaultiin.



KUVA 10. Kirjautuminen Vaultiin tokenin avulla (HashiCorp 2024).

Tokeneita on olemassa kolmea eri tyyppiä. Kuvassa 11 on nähtävillä taulukko eri token-tyypeistä. Service-tokenit ovat käyttäjän näkökulmasta ”normaaleita” tokeneita, joita voi käyttää kaikkiin toimintoihin, kuten uusimiseen (renewal) tai peruuttamiseen (revocation). Vault-versiosta 1.10 lähtien service-tokenit ovat käyttäneet etuliitettä *hvs*. Kuvan 11 alareunassa on nähtävillä esimerkki siitä, miltä service-token voi kokonaisuudessaan näyttää. Etuliitteen jälkeinen merkkijono on vähintään 24 merkkiä pitkä, ja muodostetaan satunnaisesti. Myös Vaultin käyttöönoton yhteydessä luotava root-token on service-token, jonka avulla voi tehdä mitä vain Vaultin sisällä. Root-tokenien kanssa tulisikin noudattaa erityistä varovaisuutta tuotantokäytössä. Vaultin dokumentaatiossa suositellaan root-tokenin käyttämistä vain tarvittaviin alkutoimenpiteisiin, ja sen jälkeen ylläpitäjille voidaan muodostaa rajoitetumpia service-tokeneita. Root-token olisi hyvä peruuttaa alkutoimenpiteiden jälkeen ja luoda se tarvittaessa uudelleen, mikäli vastaan tulee myöhemmin toimenpiteitä joihin sitä tarvitaan. (HashiCorp n.d.-b.)

Token Type	Vault 1.9.x or earlier	Vault 1.10 and later
Service tokens	`s.<random>`	`hvs.<random>`
Batch tokens	`b.<random>`	`hvb.<random>`
Recovery tokens	`r.<random>`	`hvr.<random>`

For example, a service token may look like `hvs.CvmS4c0DPTvHv5eJgXWMJg9r`.

KUVA 11. Vaultin token-tyypit (HashiCorp n.d.-b).

Batch-tokenit ovat service-tokeneita kevyempiä, sillä niiden seuranta ei vaadi tallennusta levylle. Batch-tokenit käyttävät etuliitettä *hvb*. (HashiCorp n.d.-b.) Batch-tokenien käyttötapaukset ovat merkittävästi rajatummat kuin service-tokeneilla. Niitä voidaan käyttää apuna Vaultin sisäisissä toiminnoissa ja klusterien eli järjestelmien replikoinnissa. (Prowse 2024.) Recovery-tokenit taas liittyvät nimensä mukaisesti Vaultin palautustilaan ja -prosessiin. Niiden avulla voidaan tehdä pyyntöjä Vault-tietosolun korjaamiseksi tietosolun ollessa palautustilassa. Recovery-tokenit käyttävät etuliitettä *hvr*. (HashiCorp n.d.-b.)

Tokeneilla on oma hierarkiansa. Tyypillisesti tokenin haltijan luodessa uusia tokeneita, nämä tokenit luodaan alkuperäisen tokenin alaisuuteen, child-tokeneiksi. Kun alkuperäinen token peruutetaan, myös sen alaisuudessa olevat child-tokenit peruuntuvat samalla. Mikäli luotavan tokenin ei haluta muodostuvan child-tokeniksi ja näin ollen peruuntuvan alkuperäisen tokenin peruuntuessa, voidaan uusi token luoda itsenäiseksi. Tällaista tokenia kutsutaan orphan-tokeniksi. Orphan-tokenilla ei ole parent-tokenia ja ne ovat tämän vuoksi oman token-hierarkiansa juuressa. Orphan-tokeneita luodakseen sekä niitä peruuttaakseen käyttäjä tarvitsee siihen soveltuvat oikeudet. (HashiCorp n.d.-b.)

Jokaiselle tokenille, lukuunottamatta root-tokenia, on määritelty voimassaoloaika eli TTL (Time to Live). Tämän ajanjakson jälkeen token peruuntuu automaattisesti. Voimassaoloaika voidaan määritellä tokenia luotaessa, mutta se ei saa ylittää TTL:n enimmäisarvoa, joka perustuu useiden tekijöiden yhdistelmään. Yksi näistä tekijöistä on Vaultin konfiguraatiotiedostossa määritelty TTL:n enimmäisarvo. Jos tokenille ei sitä luotaessa määritellä erillistä voimassaoloaikaa, se määräytyy TTL:n enimmäisajan mukaan. Mikäli token on uusittavissa, Vaultia voidaan myös pyytää pidentämään tokenin voimassaoloaikaa. (HashiCorp n.d.-b.)

Turvallisuuskäytännöt (policies) mahdollistavat Vaultissa roolipohjaisen käyttöoikeuksien hallinnan (RBAC, role-based access control). Tämä on tärkeää, jotta organisaatiot voivat evätä tai myöntää pääsyn eri kohteisiin työntekijöiden tai asiakkaiden roolien tai vaatimusten mukaan. Vaultissa on valmiina kaksi turvallisuuskäytäntöä: root- ja default- eli oletuskäytäntö. Root-käytäntö tarjoaa pääkäyttäjän oikeudet ja oletuskäytäntö perustason vakio-oikeudet. Kumpaakaan turvallisuuskäytäntöä ei voi poistaa, mutta oletuskäytäntöä voi muokata. Root-turvallisuuskäytäntöön liitetty token saa rajoittamattoman pääsyn kaikkeen Vaultissa, joten sen myöntämistä tulisi välttää ja käyttää vain välttämättömissä tilanteissa kuten alkutoimenpiteissä tai palautustilanteissa. Oletusturvallisuuskäytäntö liitetään kaikkiin tokeneihin automaattisesti, jollei tokenia luodessa määritellä toisin. (McTeer & Krausen 2020, 178–179.) Oletusturvallisuuskäytäntö sisältää perustoiminnallisuuksia, kuten tokenin omien tietojen hakemisen (HashiCorp n.d.-b.). Näiden kahden valmiin

turvallisuuskäytännön lisäksi organisaatioiden tulisi luoda käyttäjien ja sovellusten erityisiin tarpeisiin sopivia käytäntöjä. Jokainen käyttötapaus saattaa vaatia erillisen turvallisuuskäytännön. (McTeer & Krausen 2020, 179.)

Vaultin turvallisuuskäytännöt kirjoitetaan joko HCL- tai JSON-syntaksilla (McTeer & Krausen 2020, 178). HCL eli HashiCorp Configuration Language on yhteensopiva JSON-muodon kanssa (Soto Bueno & Block 2023). Jokainen turvallisuuskäytäntö määrittelee sekä toiminnon (capabilities) että polun, johon kyseisen toiminnon käyttö halutaan sallia. Nämä toiminnot vastaavat yleensä HTTP-verbiä. Taulukossa 1 on lueteltu turvallisuuskäytäntöjen toiminnot ja niitä vastaavat HTTP-verbit. (McTeer & Krausen 2020, 178–179.)

TAULUKKO 1. Turvallisuuskäytäntöjen toiminnot (capabilities) ja niitä vastaavat HTTP-verbit.

Toiminto	**HTTP-verbi**
Create	POST/PUT
Read	GET
Update	POST/PUT
Delete	DELETE
List	LIST
Deny	-
Sudo	-

Yksinkertaisimmillaan turvallisuuskäytäntö muodostuu polusta ja sille myönnetyistä toiminnoista, kuten create tai update. Turvallisuuskäytännöistä voidaan myös tehdä monimutkaisempia käyttämällä parametreja tai muuttujien korvaamista. (Krausen 2019.) Parametrien avulla voidaan estää, sallia tai asettaa pakolliseksi tiettyjen avainten tai arvojen käyttö. Tämän ominaisuuden avulla voidaan varmistaa, että vain ennalta määriteltyjä arvoja käytetään. Usein Vaultin turvallisuuskäytännöissä käytetään apuna jokerimerkkiä (*), jonka avulla voidaan myöntää pääsy kaikkiin polkuihin ennen jokerimerkkiä määritellyn polun alapuolella. Jokerimerkkiä voidaan käyttää vain polun lopussa. Kuvassa 12 on esimerkki Vaultin turvallisuuskäytännöstä, jossa käytetään parametreja sekä jokerimerkkiä. Esimerkin käytäntö sallii siihen liitetyn tokenin luoda avain/arvo-

parin polun secret/vault/authors alle. Avaimen on kuitenkin oltava joko ”headshot” tai ”bio”. (McTeer & Krausen 2020, 182–183.)

```
#Permit book publisher to create Vault authors

path "secret/vault/authors/*" {
  capabilities = ["create"]
  allowed_parameters = {
    "headshot" = []
    "bio"      = []
  }
}
```

KUVA 12. Esimerkki Vaultin turvallisuuskäytännöstä (McTeer & Krausen 2020, 183).

4.4.5 Auditointityökalut

Vault käyttää auditointityökaluja (Audit Devices) luodakseen lokeja jokaisesta todennetusta pyynnöstä ja pyyntöön liittyvästä vastauksesta. Lokien avulla ylläpitäjät voivat varmistaa tuotteen asianmukaisen käytön ja tarkastella tapahtumia. Lokeista voi olla myös apua, kun selvitellään ongelmia esimerkiksi käyttäjän sisäänpääsyssä Vaultiin tai toisaalta tutkitaan luvattomia pääsyjä Vaultiin. (McTeer & Krausen 2020, 187.)

Auditointityökalujen muodostamat lokit sisältävät tietoa käyttäjästä ja pyynnöstä, kuten käyttäjän IP-osoitteen, pyynnön ajankohdan ja Vaultin pyyntöön tuottaman datan. Ennen kuin tiedot kirjoitetaan lokeihin, arkaluontoiset tiedot kuten tunnukset ja salaisuudet tiivistetään (hashing), jotta mitään luottamuksellista tietoa ei säilytetä selkotekstinä. Tiivistämisestä huolimatta lokeja tulee käsitellä asianmukaisella tavalla. Mikäli lokeja tallennetaan ja säilytetään paikallisesti, tulisi ne salata ja huolehtia, ettei lokitietoihin pääse käsiksi luvattomasti. (McTeer & Krausen 2020, 187.)

4.4.6 Käyttöliittymät

Vaultia voidaan käyttää kolmen eri käyttöliittymän kautta: UI (User Interface), CLI (Command Line Interface) ja HTTP API (Application Programming Interface) (HashiCorp n.d.-b). Pohjimmiltaan vuorovaikutus Vaultin kanssa tapahtuu kaikissa tapauksissa API:n eli ohjelmointirajapinnan kautta, sillä myös CLI:n tai UI:n kautta Vaultia käytettäessä kaikki pyydetyt toiminnot suoritetaan API:n avulla. CLI ja UI voivat olla loppukäyttäjälle käyttäjäystävällisempiä vaihtoehtoja, mutta ne eivät tue kaikkia toimintoja, mitä suoraan API:n avulla voi tehdä. (McTeer & Krausen 2020, 19.)

Vault on salaisuudenhallintatyökaluna suunniteltu käytettäväksi API:n kautta – automatisoidusti ilman manuaalista puuttumista. Tehtävien automatisointi poistaa inhimillisten virheiden riskin ja parantaa turvallisuutta. Siksi API:n pitäisi olla yleisin tapa käyttää Vaultia. Ennen kuin tarvittavat sovellusintegraatiot on toteutettu API:a käyttäen, on kuitenkin usein tarve konfiguroida Vault CLI:n eli komentorivin kautta. Kun alkutoimenpiteet ja siihen liittyvä konfigurointi on valmis, suurin osa kaikista päivittäisistä toiminnoista voidaan siirtää tehtäväksi ohjelmallisesti API:n kautta. Komentorivin kautta on mahdollista päästä käsiksi lähes kaikkiin API-polkuihin, muutamia poikkeuksia lukuunottamatta. Komentorivin kautta työskentely onkin luonteva tapa monelle käyttäjälle ja ylläpitäjälle tehdä tarvittavia manuaalisia toimenpiteitä esimerkiksi käyttöliittymän kautta klikkailun sijaan. (McTeer & Krausen 2020, 20.)

Vault UI tai Web UI eli web-käyttöliittymä on käytettävissä useimmilla verkkoselaimilla. Käyttöliittymään pääsee käyttämällä Vault-klusterin DNS-nimeä tai verkkoaseman nimeä sekä sille konfiguroitua porttia. (McTeer & Krausen 2020, 20.) Esimerkiksi HashiCorpin palveluverkkoalusta Consulia käytettäessä voisi osoite Vault UI:lle olla ”https://vault.service.consul:8200/ui” (HashiCorp n.d.-b). Vaultia ei ole alunperin suunniteltu käytettäväksi käyttöliittymän kautta, mutta sitä on myöhemmin paranneltu ja uusia ominaisuuksia on lisätty useissa julkaisuissa. Joillekin käyttäjille web-käyttöliittymä on sopivin vaihtoehto Vaultin käyttämiseen ja kaikkein yleisimmät käyttäjän tai ylläpitäjän suorittamat tehtävät voidaan tehdä käyttöliittymän kautta. (McTeer & Krausen 2020, 20.)

4.5 Response Wrapping -ominaisuus

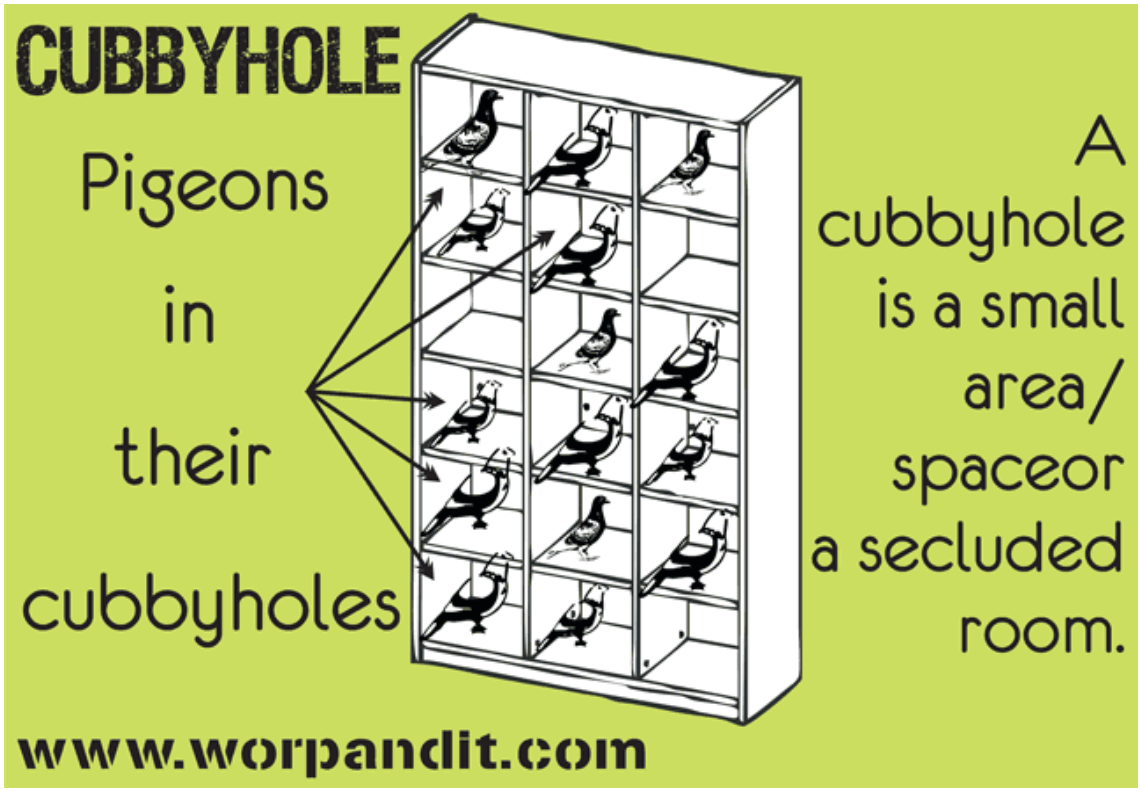
Tämän työn yhteydessä rakennettavan kertakäyttöviestipalvelun pohjana toimii Vaultin Response Wrapping -ominaisuus. Response Wrapping -ominaisuuden avulla salainen tieto voidaan välittää vastaanottajalle Vaultin kautta sen sijaan, että salaisuus lähetettäisiin verkon kautta. Response Wrapping -ominaisuus liittää salaisen tiedon kertakäyttöiseen ja ajallisesti rajoitettuun tokeniin, ja palauttaa käyttäjälle salaisen tiedon sijaan tokenin. Tätä tokenia kutsutaan wrapping-tokeniksi. Wrapping-tokenin voi välittää eteenpäin käyttäjälle, jolle salainen tieto halutaan siirtää. Wrapping-tokenin avulla tieto voidaan purkaa (unwrap) Vaultissa alkuperäiseen muotoonsa. (HashiCorp n.d.-b.)

Response Wrapping -ominaisuuden käyttäminen salaisuuden välittämiseen pienentää salaisuuden paljastumisen riskiä. Kun salaisuus kerran avataan wrapping-tokenin avulla, sitä ei voi avata enää uudestaan. Tämän avulla varmistetaan, että salaisuus on nähty vain kerran, eikä sitä ole esimerkiksi siepattu matkan varrella. Ominaisuutta käyttämällä vältytään myös siltä, että itse salaisuutta kuljetettaisiin osoitteesta toiseen, sen sijaan kuljetetaan vain viittausta siihen. Itse salaisuus ei liiku Vaultista pois missään vaiheessa. Ja koska salaisuus on liitetty ajallisesti rajoitettuun tokeniin, myös salaisuudella on sama ennalta määritelty voimassaoloaika – tyypillisesti lyhyt sellainen. (HashiCorp 2020.)

4.5.1 Cubbyhole Secrets Engine

Cubbyhole Secrets Engine mahdollistaa Response Wrapping -ominaisuuden toiminnan. Jokaisella tokenilla on oma Cubbyhole, tallennustila salaisuuksille. Cubbyhole Secrets Enginessä polut on määritelty token-kohtaisesti eikä yksikään token voi käyttää toisen tokenin Cubbyholea. Kun tokenin voimassaoloaika päättyy, myös sen Cubbyhole tietoineen tuhoutuu. Siksi Cubbyholeen tallennetulle datalle ei ole olemassa erillistä TTL-arvoa, vaan sen voimassaoloaika on aina linkitetty tokeniin, jonka avulla data on kirjoitettu. (HashiCorp n.d.-b.)

Cubbyhole voi terminä kuulostaa poikkeavalta Secrets Enginelle. Se tarkoittaa pientä tilaa jollekin tai jollekulle. Cubbyholea voidaan ajatella esimerkiksi lokerikon yhtenä lokerona, kuten kuvassa 13 havainnollistetaan. Kuvassa olevat lokerot kuvaavat hyvin myös Vaultin Cubbyholea. Jokaisella tokenilla on oma lokeronsa, kuten kuvassa olevilla puluilla. Vain lokeron omistava token voi käyttää omaa lokeroaan – kuten on myös kuvan pulujen laita. Edes pääkäyttäjätunnuksilla (root) ei voi päästä käsiksi tokenin dataan Cubbyholessa, jollei kyseisiä tietoja ole kirjoitettu root-tokenia käyttäen. (Irez 2022.) Cubbyhole Secrets Engine on aina oletuksena käytössä eikä sitä voi poistaa käytöstä tai siirtää toiseen polkuun (HashiCorp n.d.-b).

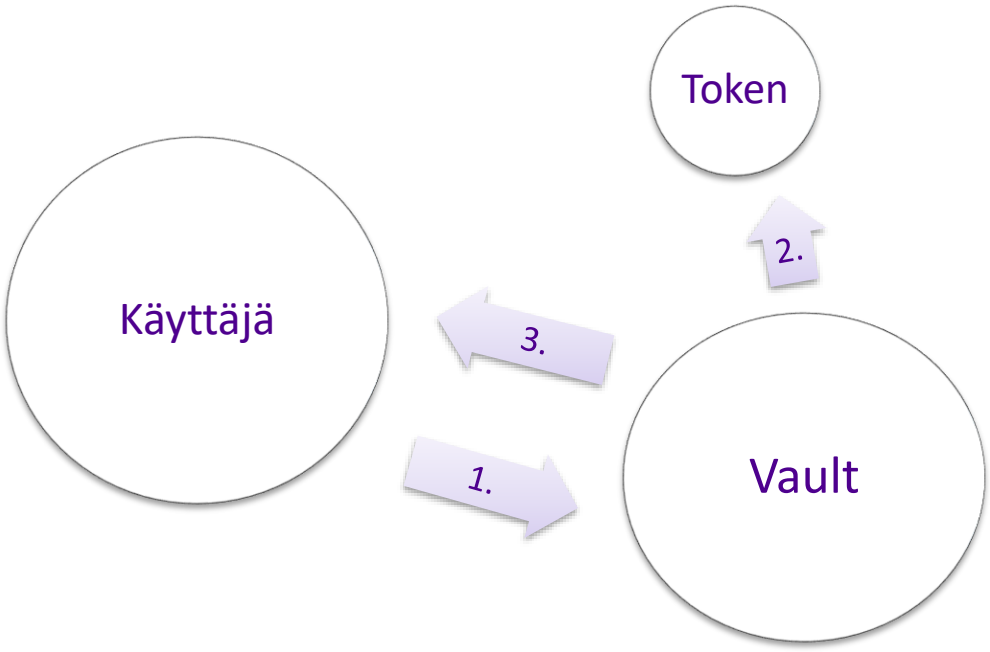


KUVA 13. Havainnollistava kuva Cubbyholen merkityksestä (Wordpandit n.d.).

4.5.2 Salaisen tiedon paketoiminen

Response Wrapping -ominaisuuden avulla salainen tieto kuvaannollisesti katsoen paketoidaan (wrapping) tokenin sisään, mistä juontuu myös ominaisuuden nimi ”Response wrapping”, vastauksen paketointi. Todellisuudessa Response Wrapping -ominaisuuden salaisuuden paketointi

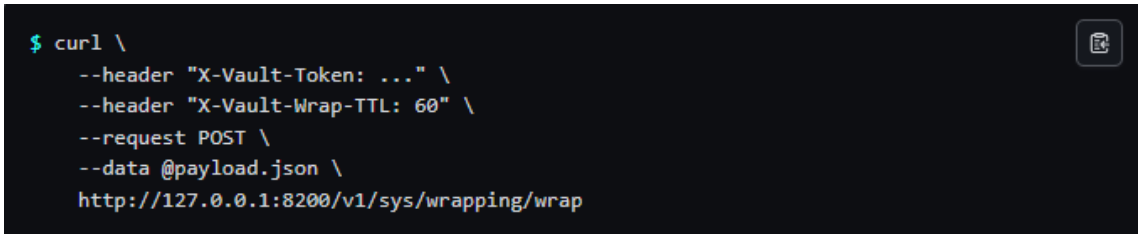
etenee kuvion 2 mukaisesti. Nuoli 1 kuvaa tapahtumaa, jossa käyttäjä lähettää Vaultille pyynnön salaisuuden paketoinnista. Tässä kohtaa käyttäjä määrittelee salaisuudelle TTL-ajan, jonka kuluessa loppuun salaisuus tuhoutuu. Pyynnön vastaanotettuaan Vault luo uuden tokenin, jolle liitetään käyttäjän antama TTL-arvo. Tätä tapahtumaa kuvaa nuoli 2. Uusi token on wrapping-token, joka on TTL-arvon luoman ajallisen rajoituksen lisäksi myös kertakäyttöinen. Vault tallentaa käyttäjän lähettämän pyynnön sisältämän salaisuuden luodun wrapping-tokenin Cubbyholeen. Kohdassa 3 Vault palauttaa wrapping-tokenin käyttäjälle. (HashiCorp n.d.-b.)



KUVIO 2. Response Wrapping -ominaisuus ja salaisuuden paketoiminen (HashiCorp n.d.-b).

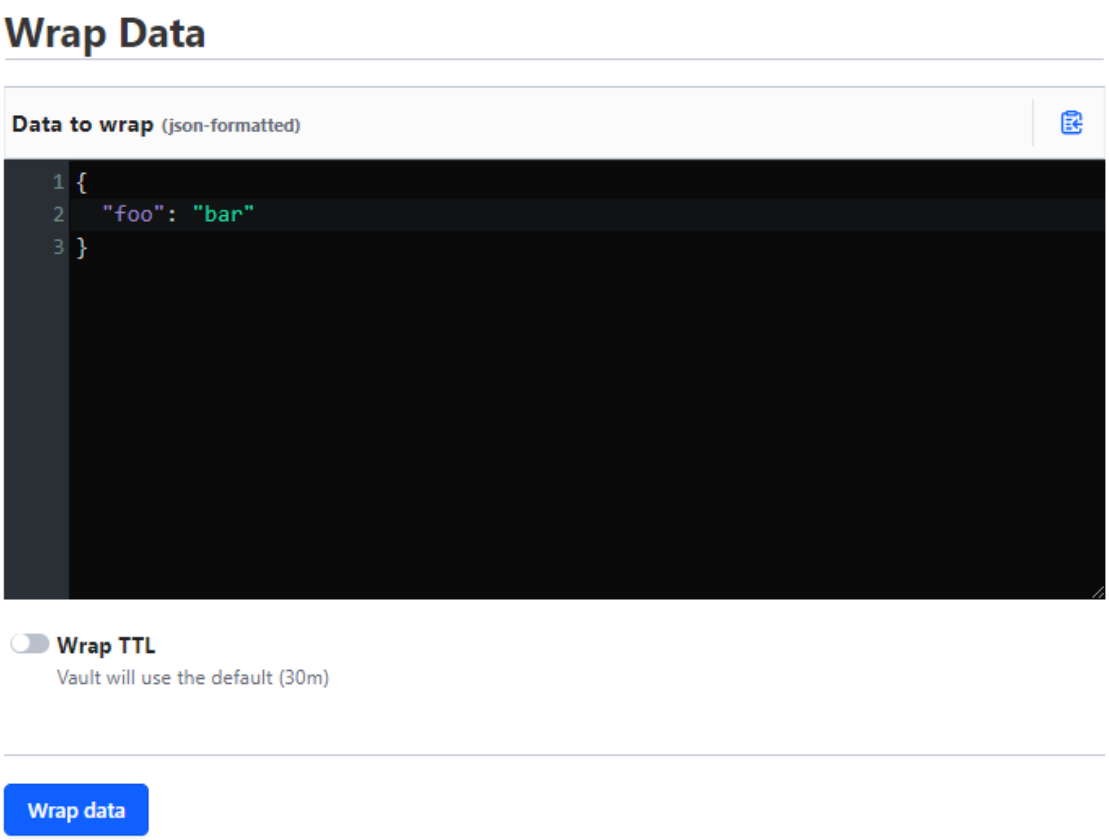
Salaisuuden paketoimisen voi tehdä sekä CLI:n, API:n tai UI:n kautta. Vault API:sta löytyy polku sys/wrapping/wrap, jonka avulla tehdään Vaultille pyyntö salaisuuden paketoinnista. Kuvassa 14 on Vaultin dokumentaatiossa esillä oleva esimerkki Response Wrapping -pyynnostä tehtynä komentorivin kautta. Pyynnössä lähetetään otsakkeena (header) X-Vault-Token eli pyynnössä tunnisteena käytettävä token ja X-Vault-Wrap-TTL eli toivottu TTL-arvo paketoitavalle salaisuudelle. Pyynnössä olevaan data-kohtaan liitetään paketoitava salaisuus. Kuvan esimerkissä data on annettu erillisen json-tiedoston muodossa. Viimeisellä rivillä on nähtävissä osoite, johon pyyntö lähetetään.

Esimerkissä on käytössä Vaultin testaamiseen ja kehittämiseen apuna käytettävä Vault dev-server, jonka vuoksi osoitteen alku on localhost-osoite (127.0.0.1), jota seuraa porttinumero (8200). Osoitteen seuraavana osana on käytettävän API:n versionumero (v1), jota seuraa API-polku haluttuun toimintoon. (HashiCorp n.d.-b.)



KUVA 14. Esimerkki Response Wrapping -pyynnöstä (HashiCorp n.d.-b).

Response Wrapping -ominaisuutta on mahdollista käyttää myös Web UI:n avulla. Kuvassa 15 on näkymä Vaultin web-käyttöliittymästä. Tekstikenttään voidaan syöttää paketoitava salainen tieto json-muodossa. Tekstikentän alapuolella on Wrap TTL -kytkinpainike, jonka avulla voidaan valita TTL-arvoksi joko oletusaika eli 30 minuuttia tai säätää omavalintainen ajanjakso. Klikkaamalla painiketta ”Wrap data”, Vaultille lähetetään pyyntö salaisen tiedon paketoimiseksi. (HashiCorp 2024.)



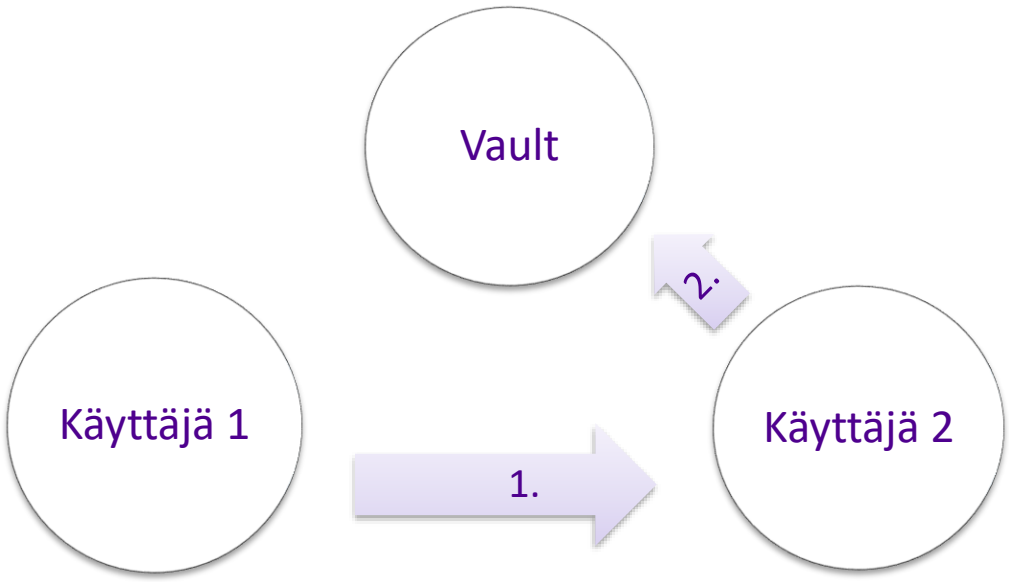
KUVA 15. Response Wrapping -ominaisuus Vaultin web-käyttöliittymässä (HashiCorp 2024).

Pyynnön lähettämisen jälkeen käyttäjälle palautetaan wrapping-token, joka on pitkä merkkijono satunnaisia merkkejä, kuten kuvassa 16 on nähtävillä (HashiCorp 2024). Tokenilla on etuliitteenä *hvs* service-tokenien mukaan ja etuliitteen jälkeinen merkkijono on aina vähintään 24 merkkiä pitkä, kuten myös muilla service-tokeneilla. Esimerkin tapauksesta näkee, että palautettu token on paljon pidempi kuin 24 merkkiä, mikä on tyypillistä wrapping-tokeneille. (HashiCorp n.d.-b; HashiCorp 2024.) Kun Vault on palauttanut tokenin, käyttäjä voi kopioida sen käyttöliittymästä talteen (HashiCorp 2024).

KUVA 16. Vaultin palauttama wrapping-token web-käyttöliittymässä (HashiCorp 2024).

4.5.3 Salaisen tiedon avaaminen

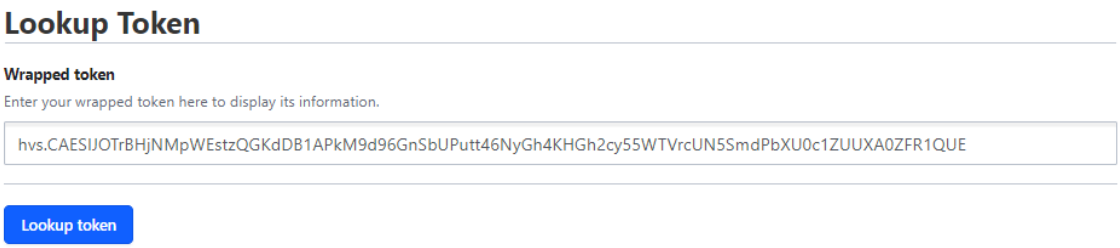
Response Wrapping -ominaisuuteen kuuluu salaisen tiedon paketoinnin lisäksi myös salaisen, aiemmin paketoidun tiedon avaaminen. Tätä toimintoa kutsutaan nimellä unwrapping. Kuviossa 3 on esitelty työnkulkua paketoinnin avaamiselle. Kuviossa käyttäjä 1 on henkilö, joka on aiemmin paketoinut tiedon Response Wrapping -ominaisuutta käyttäen ja saanut Vaultilta wrapping-tokenin. Käyttäjä 2 taas on henkilö, jolle salaista tietoa ollaan välittämässä. Kohdassa 1 käyttäjä 1 lähettää käyttäjälle 2 wrapping-tokenin valitsemallaan menetelmällä. Menetelmä voi olla esimerkiksi pikaviestisovellus tai muu käyttäjän valitsema palvelu. Lähetystavalla ei ole juurikaan väliä, koska viestissä ei välitetä salaista tietoa, vaan pelkkä viittaus siihen tokenin muodossa. Kuvion kohdassa 2 käyttäjä 2 avaa salaisen tiedon paketoinnin antamalla Vaultille käyttäjältä 1 saamansa wrapping-tokenin ja Vault palauttaa käyttäjälle 2 alkuperäisen salaisuuden. (HashiCorp n.d.-b.)



KUVIO 3. Response Wrapping -ominaisuus ja salaisuuden avaaminen (HashiCorp n.d.-b).

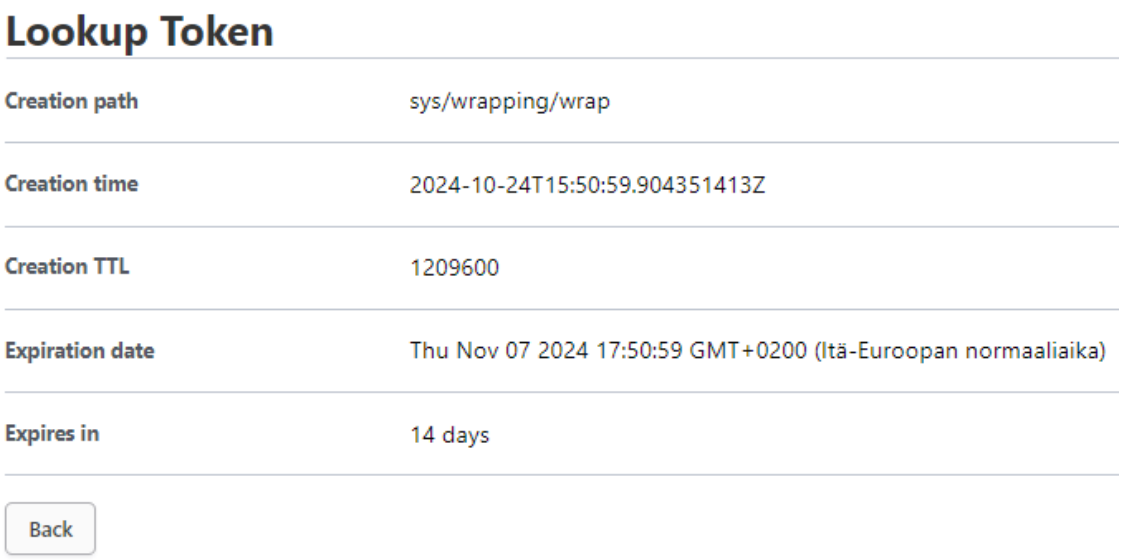
Ennen salaisen tiedon avaamista on myös mahdollista tarkastaa saadun tokenin tiedot tokenin tarkastustoiminnolla (token lookup). Toiminnon avulla voidaan

tarkastella esimerkiksi kauanko token on voimassa tai onko siihen liitetty tieto jo avattu. (HashiCorp n.d.-b.) Kuvassa 17 on nähtävillä Vaultin web-käyttöliittymän tokenin tarkastustoimintonäkymä. Token-tekstikenttään on syötetty aiemmin tiedon paketoinnista saatu wrapping-token. Klikkaamalla ”Lookup token”-painiketta Vaultille lähetetään pyyntö tarkastella annetun tokenin tietoja. (HashiCorp 2024.)



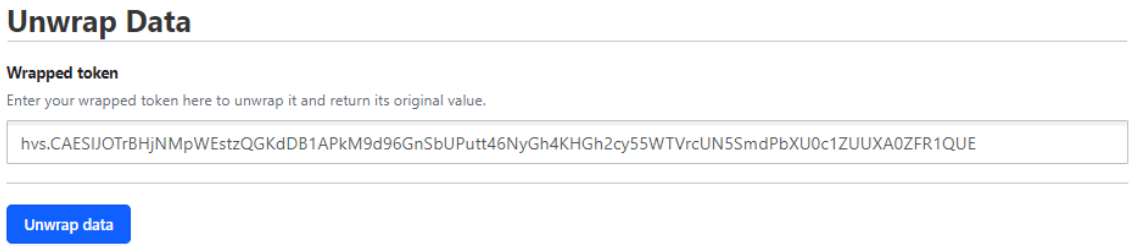
KUVA 17. Wrapping-tokenin tarkastustoiminto web-käyttöliittymässä (HashiCorp 2024).

Tarkastustoiminnon avulla tokenista saa selville monia hyödyllisiä tietoja, kuten tokenin luomisajan, viimeisen voimassaoloajakohdan ja lukeman siitä, kuinka kauan token on vielä voimassa. Lisäksi näkymästä saa selville API-polun, jonka avulla token on luotu sekä tokenille annetun TTL-arvon sekunteina. Kuvassa 18 on esillä aiemmassa näkymässä tekstikenttään syötetyn wrapping-tokenin tiedot. (HashiCorp 2024.)



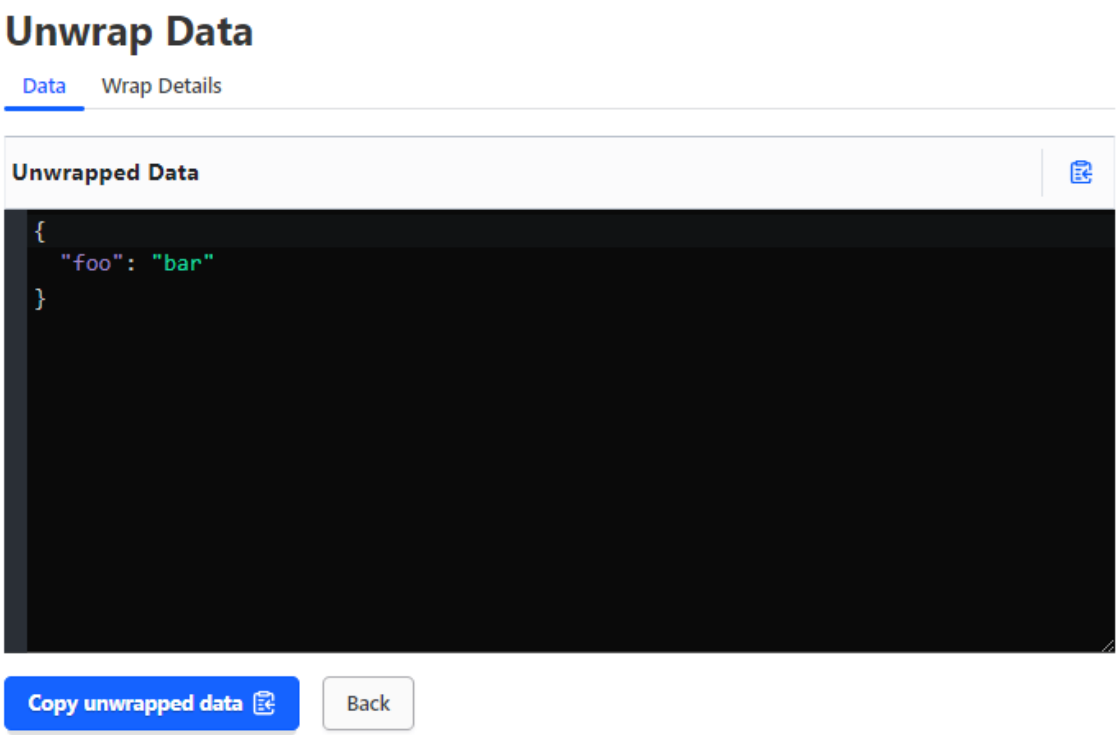
KUVA 18. Näkymä wrapping-tokenin tarkastustoimintoon syötetyn tokenin tietolistauksesta (HashiCorp 2024).

Kuten salaisen tiedon paketoiminen, myös tiedon avaaminen ja tokenin tarkastaminen on mahdollista sekä API:n, CLI:n että UI:n kautta. Vastaavasti Vault API:sta löytyy polku sys/wrapping/unwrap, jonka avulla tehdään Vaultille pyyntö salaisuuden avaamisesta. (HashiCorp n.d.-b.) Tokenin tarkastamista varten on polku sys/wrapping/lookup. Kuva 19 esittelee salaisen tiedon avaamisen Vaultin web-käyttöliittymän kautta. Teksikenttään syötetään wrapping-token ja ”Unwrap data”-painiketta klikkaamalla lähetetään Vaultille pyyntö salaisen tiedon avaamisesta. (HashiCorp 2024.)



KUVA 19. Salaisen tiedon avaaminen Vaultin web-käyttöliittymässä (HashiCorp 2024).

Pyynnön lähettämisen jälkeen käyttäjälle näytetään tokenin tallennustilasta haettu salainen data, kuten kuvan 20 esimerkissä. Käyttäjä voi kopioida tiedon käyttöliittymästä talteen. (HashiCorp 2024.) Kun salainen tieto on nyt avattu, siihen liitetty wrapping-token tietoineen tuhoutuu, eikä salaisuuteen pääse enää käsiksi toistamiseen (HashiCorp n.d.-b).



KUVA 20. Näkymä avatusta salaisesta tiedosta Vaultin web-käyttölittymässä (HashiCorp 2024).

5 KERTAKÄYTTÖVIESTIPALVELUN TOTEUTUS

5.1 Projektin tavoitteet ja vaatimukset

Tämän opinnäytetyön tuotoksena muodostui ensimmäinen versio kertakäyttöviestipalvelusta toimeksiantajayritys Haltu Oy:n käyttöön. Tavoitteena oli yhtenäistää ja vakioida käytettävät prosessit salaisten tietojen, kuten salasanojen, varmenteiden ja avainten, turvalliseen välittämiseen yrityksen sisällä. Palvelun avulla pyrittiin parantamaan turvallisuuskäytäntöjä vähentämällä riskejä, joita aiheuttavat epäyhtenäiset tai suojaamattomat tavat jakaa salaisia tietoja.

Kehitettävän palvelun tuli mahdollistaa kertakäyttöisten, vain lyhyen aikaa voimassa olevien viestien lähetys turvallisesti, jotta salaiset tiedot eivät jää pitkäksi aikaa säilytettäviksi tai altistu turvattomalle käsittelylle. Valmiin viestipalvelun sijaan haluttiin kehittää oma ratkaisu, jotta voidaan varmistua siitä, että kolmas osapuoli ei lue viestejä ja salaisia tietoja ei tallennu palveluntarjoajien palvelimille.

5.2 Projektin suunnittelu ja eteneminen

Kertakäyttöviestipalvelua lähdettiin suunnittelemaan valitsemalla ensin käytettävät teknologiat, painottaen tietoturvaa ja kustannustehokkuutta. Tavoitteena oli toteuttaa turvallinen palvelu, jonka kehittämiseen ei kuluisi liikaa henkilötyötunteja. Teknologioiden valinnassa tarkasteltiin aluksi, mitä alustoja ja palveluja yrityksellä oli jo käytössä, jotta niiden hyödyntäminen voisi nopeuttaa kehitystä ja vähentää kustannuksia.

Yrityksellä oli valmiina oma HashiCorp Vault -instanssi, joka tarjosi erinomaiset työkalut projektin tietoturvavaatimusten täyttämiseen. Sen hyödyntäminen mahdollisti ratkaisun vaivattoman integroinnin osaksi yrityksen olemassa olevaa tietoturvajärjestelmää, mikä teki siitä luontevan valinnan projektin tarpeisiin. Vaultin valinta perustui myös sen kykyyn varmistaa tietojen turvallinen hallinta ja

siirto, sekä mahdollisuuteen estää viestien uudelleenkäyttö sen Response Wrapping -ominaisuutta hyödyntämällä. Tämä ominaisuus minimoi tietovuotojen riskin ja varmistaa, että kolmannet osapuolet eivät pääse käsiksi viesteihin. Vaultin valmiin API:n ansiosta käyttöliittymän rakentaminen viestien tallentamista ja avaamista varten myös sujuisi tehokkaasti.

Ensimmäinen versio palvelusta päätettiin pitää yksinkertaisena, keskittyen vain päätoiminnallisuuksiin, jotta ratkaisu saataisiin käyttöön jo yrityksen työntekijöille. Käyttöliittymän kehityksessä suunniteltiin hyödynnettävän kevyitä teknologioita, kuten HTML:ää, CSS:ää ja JavaScriptiä. Tämä valinta perustui siihen, että ensimmäinen versio oli tarkoitettu vain sisäiseen käyttöön eikä sitä käytettäisi julkisessa verkossa. Tällöin käyttöliittymään ei tarvitsisi rakentaa monimutkaisia toiminnallisuuksia, kuten tunnistautumisjärjestelmiä.

Käyttöliittymän keveys mahdollisti nopean ja ketterän kehityksen ilman raskaita riippuvuuksia tai ohjelmistokehysratkaisuja, kuten Reactia tai Angularia. Tämä lähestymistapa helpotti myös tarvittavien muutosten tekemistä nopeasti ja tehokkaasti, mikä oli erityisen tärkeää ensimmäisen version kehityksessä. Lisäksi kevyt ja nopea käyttöliittymän toteutus minimoi kehitysaikaan liittyviä kustannuksia. Koska monimutkaisten ohjelmistokehysten asennuksia ja käyttöönottoa ei tarvittu, kehitystiimi pystyi keskittymään olennaiseen ja varmistamaan, että projekti valmistui suunnitellussa aikataulussa ja budjetissa.

Versionhallinta- sekä projektinhallintatyökaluna oli käytössä GitLab. GitLab on avoimen lähdekoodin versionhallintajärjestelmä ja ohjelmistokehitysalusta, jonka avulla voidaan hallita ja seurata ohjelmistoprojekteja, ja tehdä yhteistyötä kehitystiimin kanssa. GitLabin tietoturvaominaisuudet olivat projektin kannalta keskeisiä. Järjestelmä mahdollisti pääsyn rajoittamisen ainoastaan yrityksen työntekijöille, mikä vastasi projektin tietoturvavaatimuksia. GitLabin käyttö oli myös kustannustehokasta, sillä se oli jo yrityksen käytössä, eikä uusia työkaluja tarvinnut hankkia.

Versionhalllinnan lisäksi GitLabissa pidettiin yllä projektin työjonoa ja seurattiin kehityksen etenemistä. Siellä myös keskusteltiin ja vaihdettiin ajatuksia eri ratkaisuista ja kehityksen vaiheista ja toteutettiin koodikatselmointia. Projektin

edetessä kehitystiimi järjesti myös seuranta- ja suunnittelupalavereja, joissa käsiteltiin projektin etenemistä, mahdollisia esteitä sekä suunniteltiin tulevia kehitysvaiheita.

Viestipalvelua lähdettiin kehittämään Vaultin dev-serverin eli kehityskäyttöön suunnitellun serverin avulla. Vaultin dev-server mahdollisti nopean prototyypin luonnin ja paikallisen testaamisen ilman raskaita asennuksia tai monimutkaisia konfiguraatioita. Visual Studio Code -koodieditoria ja dev-serveriä hyödyntämällä luotiin ja testattiin funktiot, jotka lähettävät dataa Vaultin sys/wrapping/wrap -polkuun ja noutavat tallennetun tiedon sys/wrapping/unwrap -polun avulla. Palvelulle luotiin myös yksinkertainen käyttöliittymä, jossa tekstikenttään voitiin syöttää tallennettava viesti ja painikkeita painamalla tallentaa tai avata viesti.

Kertakäyttöviestipalvelulle annettiin työnimeksi Kertsi ja sille luotiin oma GitLab Pages -sivu. GitLab Pages mahdollistaa staattisen verkkosivun luomisen suoraan GitLab-repositoriosta. Tämä joustava ratkaisu mahdollisti käyttöliittymän nopean ja vaivattoman julkaisemisen yrityksen työntekijöiden käyttöön ilman monimutkaista käyttöönottoa. Verkkosivu saatiin samalla rajattua vain yrityksen sisäiseen käyttöön, sillä pääsy sivulle edellytti kirjautumista yrityksen GitLab-tilille, mikä paransi palvelun tietoturvaa ja vastasi projektin vaatimuksia. GitLab Pages oli myös taloudellisesti järkevä valinta, sillä se hyödynsi jo olemassa olevaa infrastruktuuria ilman lisäkustannuksia.

GitLab Pages:in avulla julkaistu versio käyttöliittymästä liitettiin seuraavaksi yhteen yrityksen oman Vault-instanssin kanssa. Tämä integrointi vaati erinäisiä asetuksia, kuten palvelun autentikoinnin ja valtuutusten määrittelyn Vaultin kanssa. Jotta palvelu voi tunnistautua tehdessään Vaultille pyyntöjä, se tarvitsee tokenin, joka liitetään mukaan pyyntöihin. Ensimmäisessä versiossa tämä ratkaistiin luomalla token, johon on liitetty vain ne oikeudet, joita palvelu tarvitsee toimiakseen. Token myös uusitaan säännöllisesti. Käytännössä tämä toteutettiin luomalla Vaultiin uusi turvallisuuskäytäntö, joka sisältää read, create ja update -oikeudet sys/wrapping/* polkuun. Tämän jälkeen luotiin token pohjautuen tähän turvallisuuskäytäntöön. Tokenista tehtiin orphan-token ja sille annettiin luonnin yhteydessä TTL-arvo, joka rajoittaa tokenin voimassaolon yhteen kuukauteen.

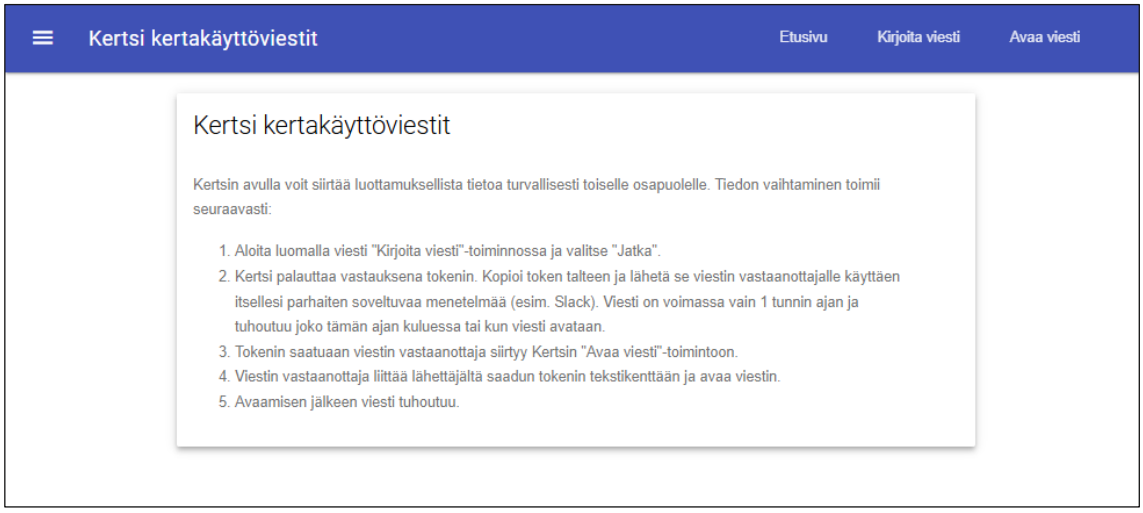
Jotta pyynnöt ja vastaukset voivat liikkua Vaultin ja kertakäyttöviestipalvelun välillä, vaadittiin myös muokkausta Vaultin CORS-asetuksiin. CORS (Cross-Origin Resource Sharing) on selainpohjainen turvallisuusmekanismi, joka estää verkkosivuja tekemästä pyyntöjä ulkopuolisille palvelimille ilman lupaa, mikä suojaa mahdollisilta haitallisilta hyökkäyksiltä. Kertakäyttöviestipalvelun ja Vaultin välinen yhteys edellytti, että Vaultin CORS-asetuksia säädettiin siten, että Vault voi vastaanottaa ja lähettää pyyntöjä palvelulta, joka saattaa olla eri verkkotunnuksessa. Tämän mahdollistamiseksi Vaultin CORS-konfiguraatioon lisättiin kertakäyttöviestipalvelun verkkotunnus sallituksi lähteeksi. Tämä muokkaus takasi, että Vault pystyy hyväksymään pyyntöjä kertakäyttöviestipalvelusta ja vastaamaan niihin turvallisesti.

Kun ensimmäiseen versioon kuuluvat toiminnallisuudet oli kehitetty, paranneltiin vielä käyttöliittymän ulkoasua lisäämällä värejä ja muotoiluja eri HTML-komponenteille. Viestin luominen ja avaaminen sekä palvelun lyhyt esittely siirrettiin myös samasta näkymästä omiin erillisiin näkymiinsä.

5.3 Ensimmäisen version ominaisuudet ja toiminta

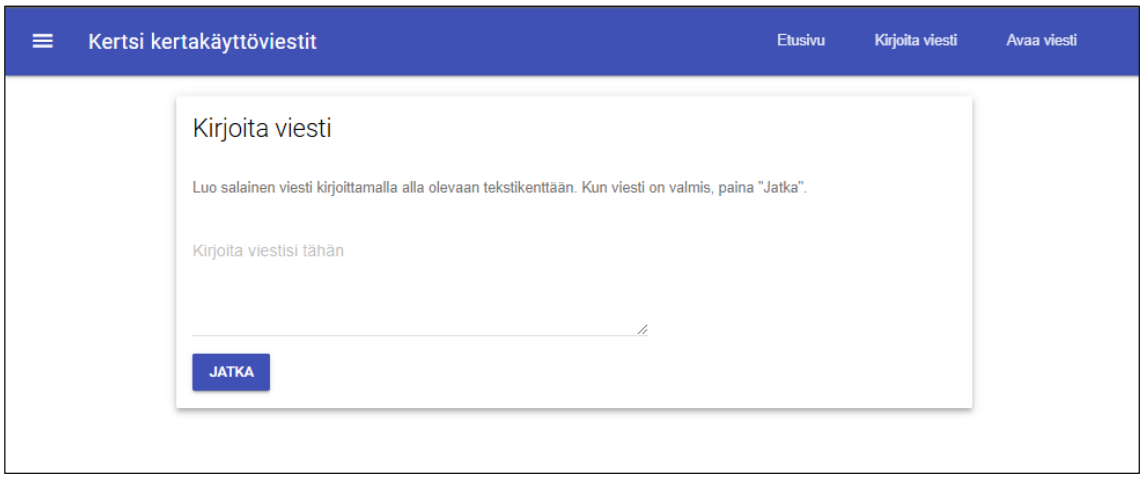
Ensimmäinen versio palvelusta sisältää pelkistetyt ydinominaisuudet eli viestin luomisen ja avaamisen. Palvelu tarjoaa käyttäjälle ohjeita tekstien avulla, mutta ei vielä näytä virheviestejä käyttöliittymässä. Käyttöliittymän sisältö ja ulkoasu eivät myöskään ole vielä lopulliset, vaan niitä hiotaan palvelun testauksessa nousevien huomioiden perusteella. Muodostunutta ensimmäistä versiota voidaankin pitää palvelun testiversiona, joka on vain yrityksen sisäisessä käytössä.

Kertakäyttöviestipalvelun etusivu sisältää lyhyen kuvauksen palvelusta sekä ohjeet sen käyttöön. Palvelun etusivu on esillä kuvassa 21. Näkymän yläreunassa on navigaatio, jonka avulla käyttäjä pääsee liikkumaan etusivun, viestin kirjoittamisen ja viestin avaamisen välillä. Vasemman yläkulman hampurilaisvalikosta aukeaa samat navigointivaihtoehdot kuin mitkä ovat näkyvillä yläpalkissa isommalla ruudulla.



KUVA 21. Kertakäyttöviestipalvelun etusivu.

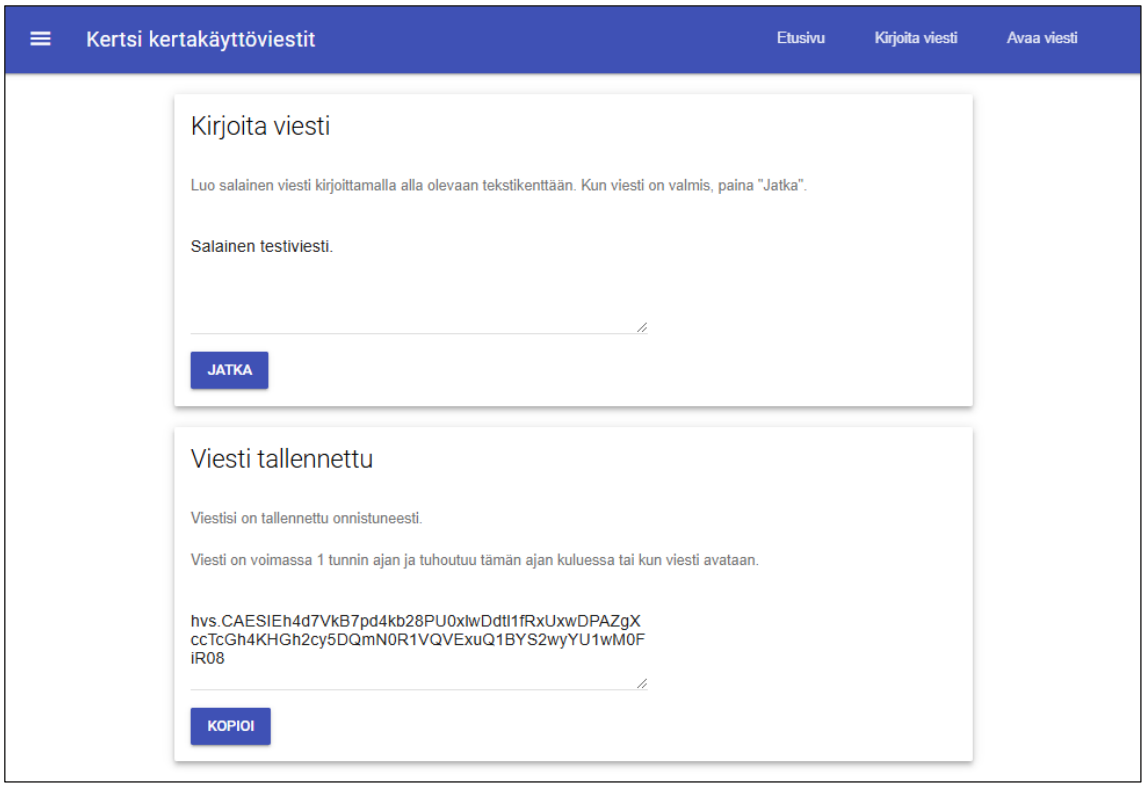
Viestin kirjoittamisen näkymässä käyttäjä voi kirjoittaa salaisen viestin sisällön kuvassa 22 näkyvään tekstikenttään. Palvelun ensimmäisessä versiossa viestin sisältöä ja pituutta ei vielä validoida mitenkään, eikä käyttäjä myöskään saa virheviestejä mahdollisista ongelmista. Kun käyttäjä on kirjoittanut viestin, valitsemalla ”Jatka” päästään viestin tallentamisen vaiheeseen.



KUVA 22. Viestin kirjoittaminen kertakäyttöviestipalvelussa.

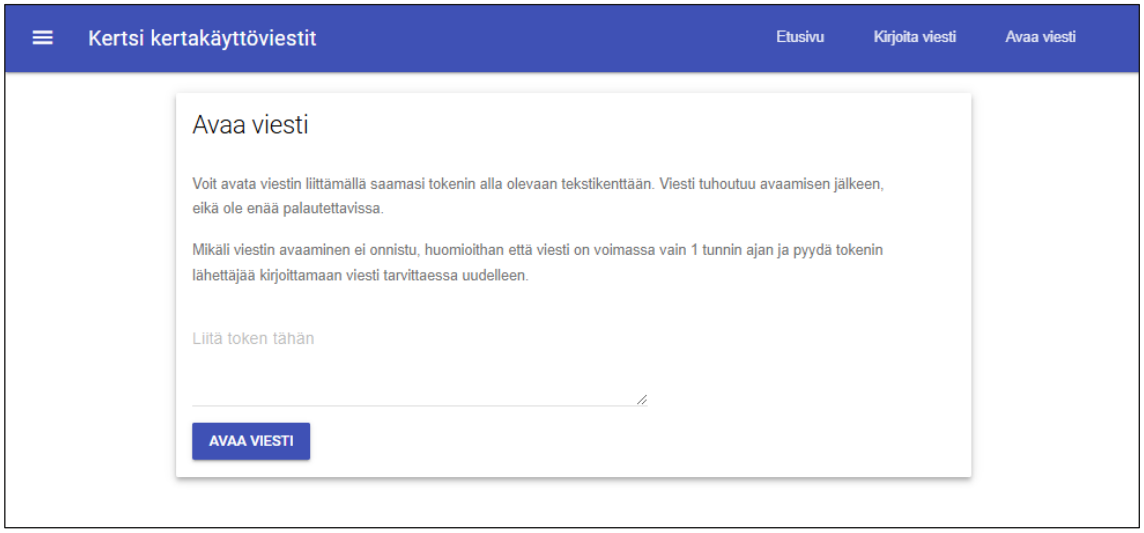
Viestin tallentamisen vaiheessa käyttäjälle ilmoitetaan tallentamisen onnistumisesta ja palautetaan token viestin avaamista varten. Taustalla Vault on luonut wrapping-tokenin, jonka tallennustilaan käyttäjän luoma salainen viesti on tallennettu. Kertakäyttöviestipalvelun ensimmäisessä versiossa viesti on voimasssa aina oletuksena yhden tunnin ajan. Tämän ajan kuluttua viesti tuhoutuu, vaikka sitä ei olisi vielä ehditty avata. Kuvassa 23 käyttäjä on luonut

salaisen viestin ja palvelu on tallentanut viestin onnistuneesti. Kirjoitetun viestin alapuolelle on ilmestynyt ohjeteksti käyttäjälle sekä tekstikenttä, joka sisältää viestin avaamiseen tarkoitetun tokenin. Käyttäjä voi nyt kopioida tokenin talteen ja lähettää sen valitsemallaan menetelmällä henkilölle, joka on tarkoitettu salaisen tiedon saajaksi.



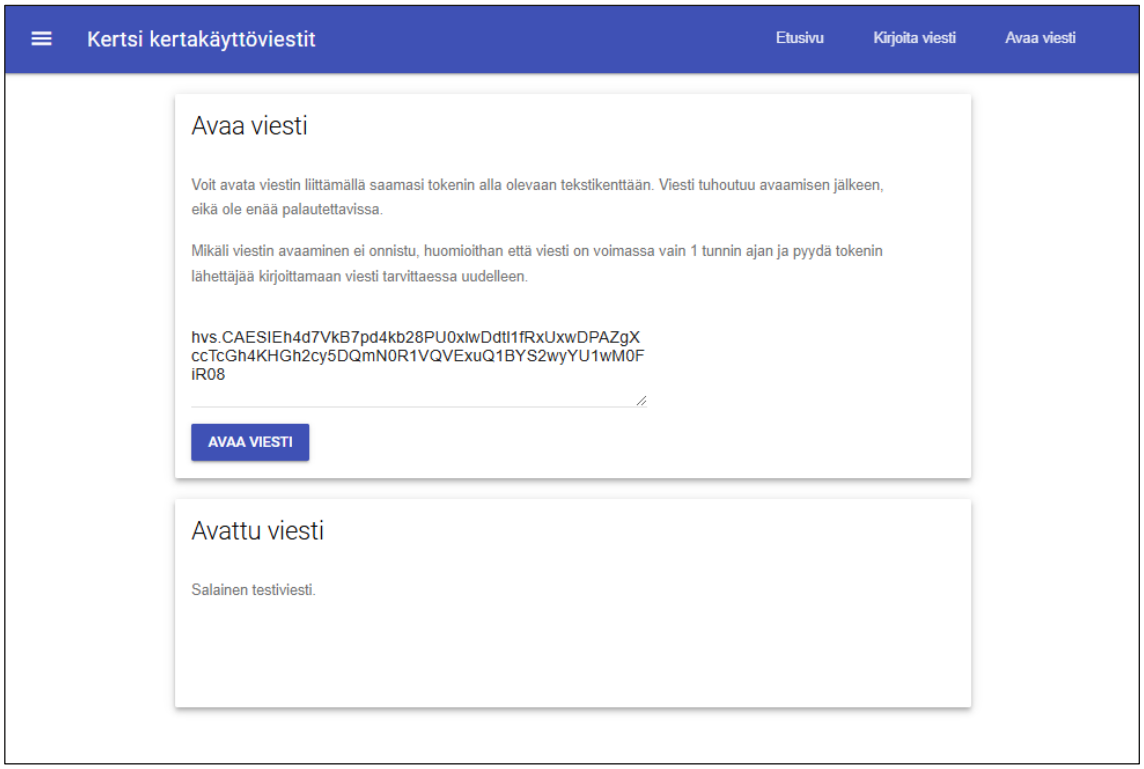
KUVA 23. Viestin tallentaminen ja tokenin kopioiminen.

Kun salaisen viestin vastaanottaja on saanut tokenin viestin lähettäjältä, siirrytään palvelun ”Avaa viesti”-näkymään viestin avaamiseksi. Näkymä sisältää ohjetekstin palvelun käyttäjälle ja tekstikentän viestin avaamisen tokenin liittämiseksi, kuten kuvassa 24 on nähtävillä. Tokenin liittämisen jälkeen käyttäjä voi avata salaisen viestin painamalla ”Avaa viesti”-painiketta.



KUVA 24. Viestin avaaminen kertakäyttöviestipalvelussa.

Mikäli token on oikea ja edelleen voimassa, avataan käyttäjän nähtäville alkuperäinen viesti. Tässä vaiheessa Vault on noutanut tokenin tallennustilasta salaisen tiedon ja palauttanut sen käyttäjälle. Samalla token ja kaikki siihen liittyvät tiedot tuhotaan. Kuvassa 25 käyttäjän liitettyä tokenin tekstikenttään ja avattuaan viestin, alapuolelle on ilmestynyt uusi alue, jossa avattu viesti on kokonaisuudessaan nähtävillä.



KUVA 25. Avattu viesti kertakäyttöviestipalvelussa.

5.4 Projektin tulevat vaiheet ja kehityskohteet

Kertakäyttöviestipalvelua on tarkoitus jatkokehittää ensimmäisestä versiosta eteenpäin. Tavoitteena on, että palvelu voidaan tulevaisuudessa ottaa käyttöön myös yrityksen ja sen sidosryhmien välillä. Seuraavana vaiheena on palvelun testaaminen ja käytettävyyden parantaminen testien perusteella. Tarkoituksena on myös suunnitella parannuksia nykyisiin ominaisuuksiin sekä lisätä uusia. Esimerkkejä näistä ovat käyttäjälle näkyvät virheviestit, mahdollisuus säätää viestin TTL-arvoa ja käyttää tokenin tarkastustoimintoa (token lookup).

On todennäköistä, että ensimmäisessä versiossa käytetyt kevyet teknologiat HTML, CSS ja JavaScript korvataan jossain vaiheessa ohjelmistokehyksellä (framework). Tämä saattaa olla tarpeen, koska palveluun suunnitellaan uusia ominaisuuksia ja muokataan ja laajennetaan nykyisiä. Ohjelmistokehys tarjoaa tehokkaampia työkaluja ja rakenteita skaalautuvuuden, ylläpidettävyyden ja kehityksen nopeuden parantamiseksi. Ohjelmistokehyksen käyttö mahdollistaisi myös monimutkaisempien toiminnallisuuksien, kuten dynaamisempien käyttöliittymien ja kehittyneempien integraatioiden, toteuttamisen. Samalla se parantaisi koodin hallittavuutta ja organisointia.

Yksi isompi kehittämisen kohde on palvelun tunnistautumis- ja valtuutusmenetelmät ja siihen liittyvä tokenin luominen Vaultiin lähetettäviä pyyntöjä varten. Ensimmäisessä versiossa token luodaan Vaultissa ja lisätään palveluun manuaalisesti säännöllisin väliajoin. Tämä prosessi tulisi automatisoida. Automatisoinnin toteuttamiseen on olemassa useita eri vaihtoehtoja, muun muassa Vaultin AppRole- tai OIDC-tunnistautumismenetelmän hyödyntäminen. Käyttäjä voisi kirjautua kertakäyttöviestipalveluun esimerkiksi Google-tunnuksilla ja kirjautumisesta välittyisi Vaultille token käytettäväksi pyyntöjen lähettämiseen. AppRolen avulla taas voitaisiin kehittää palvelun ja Vaultin välillä toimiva tunnistautuminen, jolloin ei välttämättä olisi tarvetta käyttäjän erilliselle kirjautumiselle.

Myös muotoa, jossa wrapping-token palautetaan viestin kirjoittaneelle käyttäjälle, voidaan jatkokehittää. Sen sijaan, että käyttäjä saa pelkän token-merkkijonon,

voitaisiin token upottaa esimerkiksi linkkiin, jonka avulla viestin vastaanottaja pääsisi suoraan palveluun avaamaan viestiä. Tämä nopeuttaisi ja helpottaisi palvelun käyttöä, kun välistä jäisi pois palvelun osoitteeseen navigointi, viestin avaamisnäkymän etsiminen ja tokenin kopiointi ja liittäminen tekstikenttään.

6 POHDINTA

Opinnäytetyön tavoitteena oli parantaa toimeksiantajayrityksen salaisten tietojen vaihtamisen turvallisuutta ja yhtenäistää käytäntöjä liittyen salaisten tietojen lähettämiseen ja vastaanottamiseen yrityksen sisällä. Tarkoituksena oli kehittää yrityksen sisäiseen käyttöön kertakäyttöviestipalvelu, jota voitaisiin myöhemmin jatkokehittää käytettäväksi myös yrityksen ja sen sidosryhmien välillä.

Opinnäytetyön toteutuksessa on pyritty varmistamaan sekä työn luotettavuus että toistettavuus. Kertakäyttöviestipalvelun suunnittelu, toteutus ja jatkokehityskohteet on dokumentoitu selkeästi, ja kaikki käytetyt teknologiat sekä päätöksenteon taustat on avattu läpinäkyvästi. Tämän vuoksi projekti on toistettavissa ja muokattavissa myös tulevaisuudessa, mikä parantaa sen arvoa ja käytettävyyttä pitkällä aikavälillä. Tärkeää on myös se, että palvelun kehitystyön pohjana ollut Vault-teknologia tarjoaa skaalautuvan ja turvallisen ratkaisun, jonka ominaisuuksia voidaan hyödyntää myös muiden järjestelmien ja prosessien kehittämisessä.

Yksi keskeinen osa salaisten tietojen käsittelyn turvallisuudesta muodostuu niitä käsittelevän henkilön tietämyksestä, huolellisuudesta ja prosessien noudattamisesta. Mikäli henkilön tietämys aiheesta on puuttellista ja tietoja käsittellään milloin mitenkin yhtenäisten prosessien puuttuessa, riski tietojen paljastumiselle on suuri. Tässä työssä läpikäydyt periaatteet ja salaisuuksien hallinnan käytännöt voivat toimia pohjana turvallisuuden kehittämiselle. Kertakäyttöviestipalvelu tarjoaa konkreettisen työkalun, joka parantaa tietoturvallisuutta ja estää tietojen väärinkäyttöä, mutta palvelun tehokkuus riippuu myös sen oikeasta käytöstä ja käyttäjien ymmärryksestä.

Ensimmäinen versio kertakäyttöviestipalvelusta toimii hyvänä pohjana salaisten tietojen vaihtamisen turvallisuuden kehittämiselle. Jatkokehittämällä palvelua ja ohjaamalla työntekijät – ja myöhemmin myös sidosryhmät – käyttämään palvelua salaisuuksien välitykseen, voidaan luoda yhtenäinen ja turvallinen prosessi salaisten tietojen lähettämiselle ja vastaanottamiselle. Opinnäytetyö tarjoaa

samalla dokumentaation palvelun kehittämisen etenemisestä, taustoista ja mahdollisista jatkokehityskohteista.

On tärkeää huomioida, että vaikka yhtenäinen prosessi ja turvallinen kertakäyttöviestipalvelu vievät jo pitkälle tietoturvallisen salaisuuksien vaihdon saavuttamisessa, tämä ei yksin riitä. Käyttäjien tietoisuuden lisääminen, jatkuva koulutus ja käytännön ohjeistukset turvalliseen viestintään ja salaisten tietojen käsittelyyn ovat keskeisiä. Erityisesti sen varmistaminen, että salaiset tiedot pysyvät turvassa, kun ne on noudettu kertakäyttöviestipalvelusta, on kriittinen osa prosessia, joka vaatii lisäkehitystä. On tärkeää estää tiedon vuotaminen tai väärinkäyttö sen jälkeen, kun tieto on saatu viestistä käyttöön. Tämä varmistaa, että palvelu ei jää pelkästään tekniseksi ratkaisuksi, vaan se tukee koko organisaation tietoturvakulttuuria ja mahdollistaa turvallisen tiedonvaihdon kaikissa vaiheissa.

LÄHTEET

Alvas, I. 2024. Dynamic secrets vs static secrets. Entro Security. Verkkosivu. Viitattu 13.10.2024. https://entro.security/blog/dynamic-secrets-vs-static-secrets/

Chapman, E. 2024. Mastering GitHub Actions. E-kirja. Viitattu 29.9.2024. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/mastering-github-actions/9781805128625/B21142_FM.xhtml

Chopra, U. 2024. What is a One-Time Secret: Is it Safe to Use?. Uniqkey Blog. Verkkosivu. Viitattu 25.10.2024. https://blog.uniqkey.eu/one-time-secret/

Chuvakin, A. & Viswanathan, A. 2023. Best Kept Security Secrets: Keeping secrets, the Secret Manager way. Google Cloud Blog. Verkkosivu. Viitattu 19.10.2024. https://cloud.google.com/blog/products/identity-security/best-kept-security-secrets-keeping-secrets-the-secret-manager-way

Cloudflare. 2024. What is secrets management?. Verkkosivu. Viitattu 20.10.2024. https://www.cloudflare.com/learning/security/glossary/secrets-management/

CyberArk. 2024a. What is Least Privilege?. Verkkosivu. Viitattu 20.10.2024. https://www.cyberark.com/what-is/least-privilege/

CyberArk. 2024b. What is Secrets Management?. Verkkosivu. Viitattu 20.10.2024. https://www.cyberark.com/what-is/secrets-management/

HashiCorp. n.d.-a. Secrets engines. Vault Tutorials. Verkkosivu. Viitattu 18.8.2024. https://developer.hashicorp.com/vault/tutorials/getting-started/getting-started-secrets-engines

HashiCorp. n.d.-b. Vault Documentation. HashiCorp Developer. Verkkosivu. Viitattu 24.10.2024. https://developer.hashicorp.com/vault/docs

HashiCorp. n.d.-c. Vault Pricing. Verkkosivu. Viitattu 15.6.2024. https://www.hashicorp.com/products/vault/pricing

HashiCorp. 2020. Vault Response Wrapping Makes The "Secret Zero" Challenge A Piece Of Cake. Verkkosivu. Viitattu 22.10.2024. https://www.hashicorp.com/resources/vault-response-wrapping-makes-the-secret-zero-challenge-a-piece-of-cake

HashiCorp. 2024. Vault. Versio 1.17.5. Ohjelmisto. https://developer.hashicorp.com/vault/install

Hasseltine, D. 2024. What is a One-Time Secret?. TeamPassword. Verkkosivu. Viitattu 25.10.2024. https://teampassword.com/blog/what-is-a-one-time-secret

Hospelhorn, S. 2023. Cracking the Code: How to Protect Secrets in Dev Environments. Cloud Security Alliance. Verkkosivu. Viitattu 19.10.2024.

https://cloudsecurityalliance.org/blog/2023/10/18/cracking-the-code-how-to-protect-secrets-in-dev-environments

IBM Cloud. 2024. What is a secret?. IBM Cloud Docs. Verkkosivu. Viitattu 13.10.2024. https://cloud.ibm.com/docs/secrets-manager?topic=secrets-manager-what-is-secret

Irez, Y. 2022. Vault Part 3 - Deeper Look Into Secrets Engines. Verkkosivu. Viitattu 22.10.2024. https://irezyigit.medium.com/vault-part3-deeper-look-into-secrets-engines-bb09d0041cb5

James, K. 2023. Replay Attack Vs. Man-in-the-Middle Attack. Verkkosivu. Viitattu 27.10.2024. https://cybersecurityforme.com/replay-attack-vs-man-in-the-middle-attack/

Jean-Mary, C. 2020. An Overview of X.509 Certificates. Luento. WSC Blender Program 21.9.2020. IBM Corporation. Viitattu 28.9.2024. https://www.ibm.com/support/pages/system/files/inline-files/An_Overview_of_x.509_certificates.pdf

Johnson, A. 2024. Mitigate cloud risk with Security Lifecycle Management. HashiCorp Blog. Verkkosivu. Viitattu 20.10.2024. https://www.hashicorp.com/blog/mitigate-cloud-risk-with-security-lifecycle-management

Krausen, B. 2019. Getting Started with HashiCorp Vault. Luento. O'Reilly videokurssi. Julkaisija Skylines Academy, LLC. Viitattu 6.10.2024. Vaatii käyttöoikeuden. https://learning.oreilly.com/course/getting-started-with/1018947658/

Lark Editorial Team. 2024. Self-Destructing Email. Lark Topics. Verkkosivu. Viitattu 27.10.2024. https://www.larksuite.com/en_us/topics/cybersecurity-glossary/self-destructing-email

Lundberg, J. 2019a. Essential Patterns of Vault — Part 1. HashiCorp Solutions Engineering Blog. Verkkosivu. Viitattu 28.9.2024. https://medium.com/hashicorp-engineering/essential-elements-of-vault-part-1-5a64d3de3be8

Lundberg, J. 2019b. How to identify and eliminate secrets sprawl on Azure with HashiCorp Vault. Microsoft Open Source Blog. Verkkosivu. Viitattu 20.10.2024. https://opensource.microsoft.com/blog/2019/04/04/tutorial-identify-eliminate-secrets-sprawl-hashicorp-vault-azure/

McTeer, D. & Krausen, B. 2020. Running HashiCorp Vault in Production. E-kirja. Viitattu 21.10.2024. Vaatii käyttöoikeuden. https://btkrausen.gumroad.com/l/vaultbook

Microsoft Learn. 2023. X.509 certificates. Verkkosivu. Viitattu 28.9.2024. https://learn.microsoft.com/en-us/azure/iot-hub/reference-x509-certificates

Myers, J. 2023. Vault – In Action and In The Wild. Teoksessa Krausen, B. 2023. The Best-Kept Secrets of HashiCorp Vault. A Collection of Practical Tips from

the HashiCorp Community. E-kirja. 19–26. Viitattu 20.6.2024. Vaatii käyttöoikeuden. https://btkrausen.gumroad.com/l/secretsofvault

Nield, D. 2022. 7 apps that will let you send disappearing messages. Popular Science. Verkkosivu. Viitattu 25.10.2024. https://www.popsci.com/send-self-destructing-messages/

Okta. 2024. Replay Attack: Process, Impacts, and Defense. Verkkosivu. Viitattu 27.10.2024. https://www.okta.com/identity-101/replay-attack/

Onetime Secret. n.d. Intro to Secret Links. Verkkosivu. Viitattu 25.10.2024. https://docs.onetimesecret.com/docs/secret-links

OWASP. 2024. Secrets Management Cheat Sheet. OWASP Cheat Sheet Series. Verkkosivu. Viitattu 20.10.2024. https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

Prowse, D. 2024. HashiCorp Certified Vault Associate. Luento. O'Reilly videokurssi. Julkaisija Pearson Publishing. Viitattu 5.10.2024. Vaatii käyttöoikeuden. https://learning.oreilly.com/course/hashicorp-certified-vault/9780138312923/

Red Hat. 2024. What is secrets management? Verkkosivu. Viitattu 13.10.2024. https://www.redhat.com/en/topics/devops/what-is-secrets-management

Sabharwal, N., Pandey, S. & Pandey, P. 2021. Infrastructure-as-Code Automation Using Terraform, Packer, Vault, Nomad and Consul. Hands-on Deployment, Configuration, and Best Practices. E-kirja. Viitattu 26.6.2024. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/infrastructure-as-code-automation-using/9781484271292/

Skyda. 2023a. Self Destructing Messaging Explained: The Ultimate Guide. Verkkosivu. Viitattu 27.10.2024. https://skyda.co/blog/self-deleting-messages

Skyda. 2023b. What Is a Digital Footprint? How To Reduce It?. Verkkosivu. Viitattu 27.10.2024. https://skyda.co/blog/what-is-digital-footprint

Soto Bueno, A. & Block, A. 2023. Kubernetes Secrets Management. E-kirja. Viitattu 26.6.2024. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/kubernetes-secrets-management/9781617298912/

Wickr. 2020. Top 4 Benefits of Ephemeral Messaging for Security Professionals. Verkkosivu. Viitattu 27.10.2024. https://wickr.com/top-4-benefits-of-ephemeral-messaging-for-security-professionals/

Witts, J. 2024. The Top 11 Secret Managers For Application Security. Expert Insights. Verkkosivu. Viitattu 19.10.2024. https://expertinsights.com/insights/the-top-secret-managers-for-application-security/

Wordpandit. n.d. Cubbyhole. Verkkosivu. Viitattu 22.10.2024. https://wordpandit.com/wpt_vocabulary/cubbyhole/

Young, N. 2023. Production-Level Deployment Considerations. Teoksessa Krausen, B. 2023. The Best-Kept Secrets of HashiCorp Vault. A Collection of Practical Tips from the HashiCorp Community. E-kirja. 27–44. Viitattu 26.6.2024. Vaatii käyttöoikeuden. https://btkrausen.gumroad.com/l/secretsofvault