OULUN AMMATTIKORKEAKOULU

KÄYTTÄJÄTODENTUMINEN .NET YMPÄRISTÖÖN

Toni Isopoussu
Opinnäytetyö
Syksy 2024
Tieto- ja viestintätekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistokehitys

Tekijä(t): Toni Isopoussu
Opinnäytetyön otsikko: Käyttäjätodentuminen .Net ympäristöön
Työn ohjaaja(t): Pekka Alaluukas Lehtori OAMK
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 50 + 3 liitettä

Tässä tutkimuksessa tarkasteltiin erilaisia lähestymistapoja käyttäjän todennuksen toteuttamiseen REST API -rajapinnassa keskittyen sekä räätälöityihin ratkaisuihin että valmiisiin todennuspalveluihin. Arvioinnissa otettiin huomioon sellaisia tekijöitä kuin turvallisuus, käytettävyys, ylläpidettävyys ja skaalautuvuus. Käytännön analyysiin sisältyi .NETin, SAMLin, OAuthin ja Azure AD:n kaltaisten tekniikoiden käyttö yhdistettynä tunkeutumistestaukseen järjestelmän turvallisuuden validoimiseksi.

Tulokset osoittivat, että valmiit todennusratkaisut, kuten OAuth ja Azure AD, tarjoavat turvallisen ja tehokkaan tavan toteuttaa käyttäjien todennus nykyaikaisissa järjestelmissä. Nämä ratkaisut vähentävät kehityskustannuksia ja ylläpitoponnistuksia ja tarjoavat samalla korkean turvallisuustason ja nopean käyttöönoton. Sen sijaan räätälöidyt ratkaisut tarjoavat etuja, kuten täydellisen hallinnan, mutta ne vaativat merkittäviä resursseja, syvää tietoturvaosaamista ja jatkuvaa ylläpitoa.

Tunkeutumistestien tulokset vahvistivat valmiiden ratkaisujen turvallisuuden, kun ne on konfiguroitu oikein ja niitä ylläpidetään huolellisesti. Testauksessa ei havaittu yhtään haavoittuvuutta, mikä korostaa näiden ratkaisujen kypsyyttä ja luotettavuutta.

Yhteenvetona voidaan todeta, että organisaatioiden on suositeltavaa suosia valmiita todennusratkaisuja, elleivät erityisvaatimukset edellytä mukautettua toteutusta. Valinnan tulisi olla linjassa organisaation resurssien, tietoturvatarpeiden ja pitkän aikavälin tavoitteiden kanssa. Lähestymistavasta riippumatta säännöllinen testaus ja ylläpito ovat edelleen olennaisen tärkeitä järjestelmän turvallisuuden varmistamiseksi kehittyvässä teknologisessa ympäristössä. Tämä työ tarjoaa selkeän vertailevan analyysin, joka tukee tietoon perustuvaa päätöksentekoa todentamisratkaisun valinnassa.

Abstract

Oulu University of Applied Sciences
Information and Communication Technology
Software Development

Author(s). Toni Isopoussu
Title of thesis: User authentication to the .Net environment
Supervisor(s): Pekka Alaluukas Lecturer OAMK
Month and year of completion: spring 2025
Number of pages: 50 + 3 annexes

This study examined various approaches to implementing user authentication within a REST API interface, focusing on both custom-built solutions and ready-made authentication services. The evaluation considered factors such as security, usability, maintainability, and scalability. Practical analysis included the use of technologies such as .NET, SAML, OAuth, and Azure AD, combined with penetration testing to validate system security.

The findings indicated that ready-made authentication solutions, such as OAuth and Azure AD, provide a secure and efficient means of implementing user authentication in modern systems. These solutions reduce development costs and maintenance efforts while delivering high levels of security and rapid deployment capabilities. Conversely, custom-built solutions offer advantages like complete control but require significant resources, deep expertise in security, and ongoing maintenance efforts.

Penetration testing results confirmed the security of ready-made solutions when properly configured and carefully maintained. No vulnerabilities were identified during testing, underscoring the maturity and reliability of these solutions.

In conclusion, it is recommended that organizations prioritize ready-made authentication solutions unless specific requirements necessitate a custom implementation. The choice should align with the organization’s resources, security needs, and long-term objectives. Regardless of the approach, regular testing and maintenance remain essential to ensure system security in an evolving technological landscape. This work provides a clear comparative analysis to support informed decision-making regarding authentication solution selection.

Sisällysluettelo

1 JOHDANTO 6

1.1 Lähtötilanne ja haasteet 6

1.2 Parhaiden käytäntöjen kartoitus 6

1.3 Toteutus ja vertailu 6

1.4 Lopputestaukset ja johtopäätökset 7

2 Sanasto 8

3 KÄYTTÄJÄTODENTAMINEN 10

3.1 Käyttäjätodennuksen merkitys API-rajapinnoissa 10

3.2 SAML perusteet 10

3.3 OAuth perusteet 11

4 JWT (JSON Web Token) 13

4.1 JWT toiminta 13

4.1.1 JWT:n rakenne 13

4.1.2 JWT:n toiminta käyttäjätodennuksessa 14

4.1.3 JWT:n vahvuudet ja rajoitukset 14

4.2 JWT rooli käyttäjätodennuksessa 15

4.2.1 JWT:n käyttötilanteet käyttäjätodennuksessa 15

4.2.2 JWT:n edut käyttäjätodennuksessa 15

5 TEKNOLOGINEN YMPÄRISTÖ: .NET 17

5.1 Miksi .NET käyttäjätodennuksen alustana? 17

5.2 .NET-ympäristön tuki SAML, OAuthille ja JWT 18

5.2.1 Tuki SAML:lle 18

5.2.2 Tuki OAuthille 18

5.2.3 Tuki JWT:lle 18

6 VIENTIEN SALAUS JA TIETOTURVA 20

6.1 TLS-protokolla viestien salaamiseen 20

6.1.1 TLS:n keskeiset ominaisuudet: 20

6.1.2 TLS:n käyttö API-todennuksessa: 20

6.2 HTTPS:n merkitys tietoturvassa 20

6.2.1 HTTPS:n hyödyt tietoturvassa: 21

6.2.2 HTTPS:n rooli API-todennuksessa: 21

6.3 Symmetrisen ja asymmetrisen salauksen erot ja käyttö API-todennuksessa 22

6.3.1 Symmetrinen salaus ... 22
6.3.2 Asymmetrinen salaus ... 22
6.3.3 Symmetrisen ja asymmetrisen salauksen yhdistelmä ... 22
7 PARHAAT KÄYTÄNNÖT KÄYTTÄJÄTODENNUSMENETELMISSÄ ... 24
7.1 Käyttäjätodennuksen toteuttaminen itse ... 24
7.2 Kolmannen osapuolen kirjastoja hyödyntäen ... 27
7.3 Uuden käyttäjätodentumisen toteutus REST API:lla ... 29
8 TIETOTURVA JA PENETRAATIOTESTAUS ... 32
8.1 Windows 11 -käyttäjän todentaminen ... 32
8.2 Penetraatiotestauksen merkitys ... 32
8.3 Testausprosessin vaiheet ... 34
8.3.1 Suunnittelu ja valmistelu ... 34
8.3.2 Tiedonkeruu ... 34
8.3.3 Haavoittuvuuksien tunnistaminen ... 35
8.3.4 Hyökkäyksen simulointi ... 35
8.3.5 Tulosten analysointi ... 35
8.3.6 Raportointi ... 36
8.3.7 Korjaustoimet ja uudelleentestaus ... 36
8.3.8 Yleiset murtautumistavat ... 37
8.3.9 Pohdinta ... 38
9 VERTAILU: ITSE TOTEUTETTU VS. API KEY ... 39
9.1 Edut ja haasteet itse toteutetuissa ratkaisuissa ... 39
9.1.1 Edut itse toteutetuissa ratkaisuissa ... 39
9.1.2 Haasteet itse toteutetuissa ratkaisuissa ... 40
9.2 Valmiiden autentikaatioratkaisujen edut ja haasteet ... 41
9.2.1 Edut valmiista autentikaatioratkaisuista ... 41
9.2.2 Haasteet valmiiden autentikaatioratkaisujen käytössä ... 42
10 YHTEENVETO JA POHDINTA ... 44
10.1 Tutkimustulokset ... 44
10.1.1 Keskeiset havainnot ... 44
10.1.2 Pohdinta ... 45
10.2 Suositukset organisaatioille ... 45
10.3 Johtopäätös ... 47
11 Lainatut lähteet ... 49

1 JOHDANTO

Käyttäjätodennus REST API -rajapinnassa on keskeinen osa verkkopalveluiden tietoturvaa ja käyttökokemusta. Tämän opinnäytetyön tavoitteena on tarkastella teoreettisesti ja käytännössä eri vaihtoehtoja käyttäjätodennuksen toteuttamiseen. Keskeinen kysymys kuuluu: onko käyttäjätodennuksen toteuttaminen itse edelleen perusteltua, vai olisiko viisaampaa hyödyntää valmiita ratkaisuja, kuten OAuth- tai API-avainpohjaisia autentikaatiopalveluja.

1.1 Lähtötilanne ja haasteet

Perinteisesti käyttäjätodennus on toteutettu itse hallituilla menetelmillä, kuten Basic Authentication tai räätälöidyillä kirjautumisjärjestelmillä. Vaikka nämä ratkaisut tarjoavat joustavuutta ja täyden hallinnan, niiden kehitys ja ylläpito vaativat syvällistä tietoturvaosaamista. Lisäksi kasvavat tietoturvavaatimukset ja monimutkaistuvat hyökkäysvektorit ovat tehneet itserakennettujen ratkaisujen hallinnasta entistä haastavampaa. Samaan aikaan valmiit ratkaisut, kuten OAuth-pohjaiset palvelut ja Azure AD, tarjoavat nopeamman käyttöönoton ja vakiintuneita tietoturvakäytäntöjä, mutta ne tuovat mukanaan riippuvuuksia ulkoisista palveluista sekä mahdollisia kustannushaasteita.

1.2 Parhaiden käytäntöjen kartoitus

Työssä kartoitetaan modernit parhaat käytännöt REST API -rajapinnan käyttäjätodennuksessa. Näihin kuuluvat sekä itse toteutetut ratkaisut, kuten SAML-pohjainen autentikointi, että valmiit ratkaisut, kuten OAuth ja Azure AD. Tarkastelussa keskitytään erityisesti tietoturvaan, ylläpidettävyyteen, skaalautuvuuteen ja käyttäjäkokemukseen. Näiden näkökohtien analyysi tarjoaa pohjan eri ratkaisujen vertailulle.

1.3 Toteutus ja vertailu

Työn käytännön osuudessa simuloidaan ja arvioidaan kahta lähestymistapaa. Ensimmäisessä skenaariossa käyttäjätodennus toteutetaan itse hyödyntäen SAML-protokollaa ja .NET-ympäristön tarjoamia mahdollisuuksia. Toisessa lähestymistavassa

käytetään valmiita autentikaatioratkaisuja, kuten OAuth- ja API-avainpohjaisia palveluja, ja integraatiota Azuren tarjoamiin autentikointipalveluihin. Molempien ratkaisujen toiminnallisuutta, tietoturvaa ja kustannustehokkuutta tarkastellaan kriittisesti sekä teoreettisesta että käytännöllisestä näkökulmasta.

1.4 Lopputestaukset ja johtopäätökset

Molemmille toteutustavoille suoritetaan kattavat toiminnalliset ja tietoturvatestaukset, mukaan lukien penetraatiotestit. Testausten tuloksia käytetään arvioimaan, kumpi ratkaisu on tarkoituksenmukaisempi eri organisaatioiden tarpeisiin. Lopullisissa johtopäätöksissä huomioidaan tietoturvan, kustannusten, joustavuuden ja käyttökokemuksen välinen tasapaino. Työ tarjoaa suosituksia organisaatioille, jotka miettivät optimaalista lähestymistapaa käyttäjätodennukseen nykypäivän tietoturvavaatimusten ja teknologisten mahdollisuuksien valossa.

2 Sanasto

Azure AD (Azure Active Directory)
Microsoftin pilvipohjainen identiteetin hallintapalvelu, joka tukee käyttäjätodennusta ja pääsynhallintaa useille sovelluksille ja palveluille.

API-avain
Salainen tunniste, joka liitetään API-pyyntöihin käyttäjän tai sovelluksen autentikoimiseksi ja valtuuttamiseksi.

Basic Authentication
Yksinkertainen autentikointimenetelmä, jossa käyttäjätunnus ja salasana lähetetään suojatussa muodossa jokaisessa pyynnössä.

Haavoittuvuusanalyysi
Prosessi, jossa kartoitetaan järjestelmän mahdolliset tietoturvaheikkoudet ja arvioidaan niiden vakavuus.

Istunto (Session)
Käyttäjän ja järjestelmän välinen vuorovaikutusjakso, jonka aikana käyttäjä on autentikoitu ja voi käyttää resursseja.

JWT (JSON Web Token)
Kevyt, tilaton token-muoto, jota käytetään käyttäjän identiteetin varmistamiseen ja valtuutustietojen välittämiseen järjestelmien välillä.

Konfigurointi (Configuration)
Järjestelmän asetusten määrittäminen toiminnan varmistamiseksi ja tietoturvauhkien ehkäisemiseksi.

Käyttäjätodennus
Prosessi, jossa varmistetaan käyttäjän identiteetti esimerkiksi käyttäjätunnuksen ja salasanan, tokenien tai ulkoisten autentikaatiopalvelujen avulla.

SAML (Security Assertion Markup Language)
XML-pohjainen standardi, jota käytetään käyttäjän autentikoinnin ja valtuutustietojen välittämiseen kahden järjestelmän välillä.

MFA (Multi-Factor Authentication)
Monivaiheinen todennusprosessi, jossa käyttäjän on todistettava identiteettinsä useammalla kuin yhdellä menetelmällä, kuten salasanalla ja tekstiviestillä.

OAuth
Avoin standardi, joka mahdollistaa token-pohjaisen valtuutuksen antamalla kolmansille osapuolille pääsyn resursseihin ilman, että käyttäjän tunnistetietoja tarvitsee jakaa.

Penetraatiotestaus
Tietoturvatestausmenetelmä, jossa simuloidaan hyökkäyksiä järjestelmää vastaan haavoittuvuuksien tunnistamiseksi ja korjaamiseksi.

REST API (Representational State Transfer Application Programming Interface)
Rajapinta, joka mahdollistaa tietojen välittämisen palvelimen ja asiakkaan välillä HTTP-protokollaa käyttäen.

Rajapinta (Interface)
Ohjelmiston osa, joka mahdollistaa tietojen välittämisen ja toiminnan ohjaamisen eri järjestelmien välillä.

Token
Digitaalinen tunniste, joka edustaa käyttäjää ja antaa pääsyn järjestelmän resursseihin rajoitetuksi ajaksi.

Tietoturva-aukko
Haavoittuvuus järjestelmässä, joka voi mahdollistaa hyökkääjän luvattoman pääsyn resursseihin tai tietoon.

Valtuutus (Authorization)
Prosessi, jossa määritetään, mihin resursseihin käyttäjällä on pääsy, kun hänen identiteettinsä on todennettu.

3 KÄYTTÄJÄTODENTAMINEN

3.1 Käyttäjätodennuksen merkitys API-rajapinnoissa

Käyttäjätodennus on keskeinen osa API-rajapintojen tietoturvaa ja pääsynhallintaa. Sen avulla varmistetaan, että vain oikeutetut käyttäjät ja sovellukset voivat käyttää rajapinnan tarjoamia resursseja. Tämä on erityisen tärkeää, koska REST API -rajapinnat ovat usein avoinna internetiin ja alttiina luvattomille pääsy-yrityksille.

Ilman riittävää käyttäjätodennusta API-rajapintaa voidaan väärinkäyttää esimerkiksi tunkeutumisyrityksissä, datan varastamisessa tai palvelun väärinkäytössä, kuten liiallisessa kuormituksessa (Palvelunestohyökkäykset). Hyvin suunniteltu käyttäjätodennus vähentää näitä riskejä ja auttaa luomaan turvallisen ympäristön datan käsittelylle ja tiedonvaihdolle.

Lisäksi käyttäjätodennuksella varmistetaan, että käyttäjäoikeudet ovat selkeästi määriteltyjä ja hallittavissa. Tämä mahdollistaa myös lisäominaisuuksia, kuten käyttäjän identiteettiin perustuvan pääsynhallinnan (RBAC, Role-Based Access Control) sekä tapahtumien jäljitettävyyden. Näin käyttäjätodennus tukee paitsi tietoturvaa, myös käyttäjäkokemusta ja järjestelmän hallittavuutta.

3.2 SAML perusteet

SAML (Security Assertion Markup Language) on avoin standardi, joka mahdollistaa käyttäjän todennuksen ja valtuutuksen siirtämisen eri järjestelmien välillä. Se on erityisen hyödyllinen yhden kirjautumisen (Single Sign-On, SSO) toteutuksessa, jossa käyttäjä voi käyttää useita palveluita kirjautumalla sisään vain kerran. SAML käyttää XML-pohjaista viestintää varmistaakseen, että autentikointitiedot voidaan jakaa turvallisesti eri osapuolten välillä.

SAML-arkkitehtuuri koostuu kolmesta pääosapuolesta:

1. **Identity Provider (IdP)**: Vastaa käyttäjän todennuksesta ja toimittaa todennustiedot turvallisesti.

2. **Service Provider (SP)**: Tarjoaa palvelun, johon käyttäjä haluaa päästä. SP luottaa IdP:n tarjoamiin todennustietoihin.
3. **Käyttäjä**: Interaktioiden keskipiste, joka tarvitsee pääsyn SP:n tarjoamiin resursseihin.

SAML-prosessin ydin on "assertio", joka on viesti, jonka IdP luo ja allekirjoittaa kryptografisesti varmistaakseen sen aitouden. Tämä assertio sisältää tietoja käyttäjän todennuksesta, kuten käyttäjätunnisteen ja valtuutustiedot.

Kuten OASIS:n dokumentaatiossa todetaan:
"SAML defines a framework for exchanging security information between online business partners. The primary element of this framework is the SAML assertion, an XML-based structure that contains one or more statements about a subject. Statements can assert authentication, attribute, or authorization information."
(OASIS Security Services Technical Committee, 2005, s. 3).

SAML tarjoaa vahvan tietoturvan hyödyntämällä salattua viestintää (esim. TLS) ja digitaalista allekirjoitusta. Se on erityisen hyödyllinen suurissa organisaatioissa ja ympäristöissä, joissa on useita palveluita, sillä se vähentää tarvetta hallita useita käyttäjätunnuksia ja salasanoja eri järjestelmissä. Lisäksi SAML tukee skaalautuvuutta ja integraatiota useiden eri teknologioiden ja alustojen välillä.

3.3 OAuth perusteet

OAuth (Open Authorization) on avoin standardi, jota käytetään käyttäjän valtuutuksen hallintaan verkkosovelluksissa ja API-rajapinnoissa. OAuth mahdollistaa sen, että käyttäjä voi antaa kolmannen osapuolen sovellukselle rajoitetun pääsyn resursseihinsa ilman, että käyttäjän tarvitsee jakaa tunnuksiaan, kuten salasanaa. Tämä tekee siitä turvallisemman ja joustavamman vaihtoehdon perinteisiin todennusmenetelmiin verrattuna.

OAuth-arkkitehtuuri sisältää seuraavat keskeiset osapuolet:

1. Resource Owner (resurssin omistaja): Henkilö tai käyttäjä, joka hallitsee resurssia ja päättää, kenelle ja mitä käyttöoikeuksia myönnetään.
2. Client (asiakassovellus): Sovellus, joka pyytää pääsyä resurssiin resurssin omistajan puolesta.

3. Authorization Server (valtuutuspalvelin): Palvelu, joka hallitsee valtuutusta ja myöntää valtuutustunnuksia (access tokens).
4. Resource Server (resurssipalvelin): Palvelin, joka säilyttää suojattuja resursseja ja käyttää valtuutustunnuksia pääsynhallintaan.

OAuth-prosessi tapahtuu tyypillisesti seuraavasti:

1. Asiakassovellus pyytää resurssin omistajalta lupaa käyttää resursseja.
2. Resurssin omistaja antaa suostumuksen Authorization Serverin kautta.
3. Authorization Server myöntää asiakassovellukselle valtuutustunnuksen (access token).
4. Asiakassovellus käyttää tätä tunnusta päästäkseen Resource Serveriin ja suorittaakseen pyydetyt toiminnot.

Kuten RFC-dokumentaatiossa todetaan:
"OAuth introduces an authorization layer and separates the role of the client from that of the resource owner. Instead of using the resource owner's credentials to access protected resources, the client obtains an access token."
(Hardt, D., 2012, s. 7).

OAuth 2.0, joka on standardin yleisimmin käytetty versio, tukee useita valtuutusvirtoja (grant types), kuten:

- Authorization Code Flow: Käytetään tyypillisesti verkkosovelluksissa. Tunnukset kulkevat palvelimien välillä, mikä parantaa tietoturvaa.
- Implicit Flow: Sopii asiakaspuolen sovelluksiin, mutta on vähemmän turvallinen.
- Client Credentials Flow: Käytetään sovellusten välisessä (machine-to-machine) kommunikoinnissa.
- Resource Owner Password Credentials Flow: Käytetään harvemmin, koska se vaatii käyttäjän salasanaa.

OAuth on laajalti käytössä, koska se mahdollistaa joustavan pääsynhallinnan, erinomaisen käyttäjäkokemuksen ja skaalautuvuuden. Sitä käytetään yleisesti suurissa palveluissa, kuten Google, Facebook ja Microsoft, tarjoamaan kolmansille osapuolille pääsyn resursseihin turvallisesti.

4 JWT (JSON Web Token)

4.1 JWT toiminta

JWT (JSON Web Token) on kevyt ja turvallinen standardi tietojen siirtämiseen osapuolten välillä. Se on erityisen suosittu käyttäjätodennuksessa ja pääsynhallinnassa, koska se mahdollistaa tilattoman autentikaation, jossa käyttäjän tila (state) ei tallennu palvelimelle, vaan kaikki tarvittava tieto sisältyy itse tokeniin.

4.1.1 JWT:n rakenne

JWT koostuu kolmesta osasta, jotka on erotettu pisteillä (".":

1. Header (otsikko): Sisältää metatietoa, kuten käytetyn algoritmin (esim. HS256, RS256) ja tokenin tyypin.
2. Payload (kuorma): Sisältää tiedot (claims), jotka määrittelevät tokenin tarkoituksen, kuten käyttäjän tunnisteen tai roolit. Tämä osio ei ole salattu, mutta se voidaan allekirjoittaa.
3. Signature (allekirjoitus): Varmistaa tokenin aitouden ja suojaa sitä manipuloinnilta. Allekirjoitus luodaan yhdistämällä header ja payload, jotka salataan salaisella avaimella tai julkisen/privaatin avaimen parilla.

Esimerkki JWT-tokenista:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImpvaG5kb2UiLCJyb2xlcyI6WyJ1c2VyIl0sImlhdCI6MTYwNzgzMjAwMH0.V7lJ6ZL9tfiE5kBYcLFVULrsPcZmg2oPiNkEZ3E_dFg

Kuten RFC 7519 -dokumentissa todetaan:
"A JWT is represented as a sequence of URL-safe parts separated by period (‘.’) characters. Each part contains a base64url-encoded value. The first part is the header, the second part is the payload, and the third part is the signature."
(Jones, M., Bradley, J., & Sakimura, N., 2015, s. 6).

4.1.2 JWT:n toiminta käyttäjätodennuksessa

1. Tokenin luonti: Kun käyttäjä kirjautuu onnistuneesti sisään, palvelin luo JWT-tokenin, joka sisältää käyttäjään liittyviä tietoja, kuten käyttäjätunnuksen ja voimassaoloajan.
2. Tokenin jakelu: Token lähetetään käyttäjälle esimerkiksi HTTP-otsikon tai evästeen kautta.
3. Tokenin käyttö: Kun käyttäjä tekee pyynnön API:lle, token liitetään pyyntöön (yleensä Authorization-otsikkoon muodossa Bearer <token>).
4. Tokenin validointi: API-palvelin tarkistaa, että token on allekirjoitettu oikein ja että se on yhä voimassa.

4.1.3 JWT:n vahvuudet ja rajoitukset

Vahvuudet:

- Tilaton rakenne vähentää palvelimen kuormitusta, koska käyttäjän tila ei tarvitse tallentamista.
- Token voidaan helposti siirtää osapuolten välillä.
- Tukee monipuolisia käyttötapauksia, kuten pääsynhallintaa ja käyttäjien roolipohjaista tunnistamista (RBAC).

Rajoitukset:

- Payload ei ole salattu, joten sen sisältö on helposti luettavissa, jos token joutuu vääriin käsiin.
- Tokenin invalidointi ennen voimassaoloajan päättymistä voi olla haastavaa, koska JWT on tilaton.

JWT tarjoaa tehokkaan ja joustavan ratkaisun käyttäjätodennukseen ja pääsynhallintaan, mutta sen käyttö edellyttää huolellista tietoturvan suunnittelua, kuten HTTPS:n käyttöä ja hyvin hallittua avainhallintaa.

4.2 JWT rooli käyttäjätodennuksessa

JWT (JSON Web Token) on tärkeä väline käyttäjätodennuksessa, erityisesti REST API -rajapinnoissa ja mikropalveluarkkitehtuureissa, joissa tarvitaan tilatonta ja tehokasta autentikointiratkaisua. Sen avulla voidaan välittää tietoa käyttäjän identiteetistä ja käyttöoikeuksista turvallisesti ilman tarvetta säilyttää käyttäjän tilaa palvelimella.

4.2.1 JWT:n käyttötilanteet käyttäjätodennuksessa

1. Autentikaatiotokenina: JWT toimii todisteena siitä, että käyttäjä on todistettavasti autentikoitu. Palvelin luo JWT-tokenin käyttäjän kirjautuessa sisään, ja käyttäjä liittää tokenin kaikkiin seuraaviin pyyntöihinsä.
2. Pääsynhallinta: JWT sisältää tietoja käyttäjän rooleista, käyttöoikeuksista ja mahdollisista resurssikohtaisista rajoituksista. Tämä mahdollistaa pääsynhallinnan toteutuksen ilman tarvetta tehdä ylimääräisiä kyselyitä tietokantaan.
3. Single Sign-On (SSO): JWT mahdollistaa yhden kirjautumisen useisiin eri palveluihin. Käyttäjä voi kirjautua yhteen palveluun, ja sama token hyväksytään muissa palveluissa ilman lisäkirjautumisia.

4.2.2 JWT:n edut käyttäjätodennuksessa

- Kevyt ja nopea: Koska kaikki todennukseen tarvittava tieto on tokenissa, palvelin voi validoida tokenin ilman tarvetta viestiä erillisen todennuspalvelimen kanssa.
- Skaalautuva: Tilattomuutensa ansiosta JWT tukee suuria käyttäjämääriä ja palvelukuormaa. Palvelimen ei tarvitse tallentaa istuntotietoja, mikä helpottaa kuormanjakoa ja skaalautumista.
- Käyttäjäystävällinen: Token voidaan tallentaa selaimeen evästeenä tai paikalliseen tallennustilaan, jolloin se voidaan lähettää automaattisesti kaikkiin pyyntöihin. Tämä parantaa käyttäjäkokemusta ja vähentää kirjautumistarvetta.

JWT:n käyttö käyttäjätodennuksessa edellyttää erityistä huomiota tietoturvaan:

- HTTPS-pakollisuus: Tokenin siirto HTTP:n kautta tekee sen alttiiksi sieppaukselle. HTTPS-salaus on välttämätön.

- Tokenin voimassaoloaika: Tokenin käyttöaika kannattaa rajoittaa, jotta siepatun tokenin väärinkäyttö minimoidaan.
- Avainhallinta: Allekirjoituksessa käytettyjen salausavaimien tulee olla hyvin suojattuja ja hallinnoituja.

JWT on keskeinen osa modernia autentikointia erityisesti sen tilattomuuden, nopeuden ja helppokäyttöisyyden ansiosta. Se tarjoaa erinomaisen työkalun käyttäjätodennukseen ja pääsynhallintaan API-rajapinnoissa, mutta sen käyttö edellyttää tarkkaa tietoturvan suunnittelua, jotta sen hyödyt voidaan saavuttaa ilman riskejä.

5 TEKNOLOGINEN YMPÄRISTÖ: .NET

5.1 Miksi .NET käyttäjätodennuksen alustana?

.NET on monipuolinen ja skaalautuva kehitysalusta, joka soveltuu erinomaisesti käyttäjätodennuksen toteutukseen moderneissa sovelluksissa. Se tarjoaa valmiita kirjastoja ja työkaluja käyttäjätodennuksen ja pääsynhallinnan integroimiseksi sekä tukee useita autentikointistandardeja, kuten SAML, OAuth ja JWT.

Syitä valita .NET käyttäjätodennuksen alustaksi:

1. Monipuoliset kehitystyökalut:
   .NET-alustaan kuuluu valmiita kirjastoja, kuten ASP.NET Core Authentication, jotka helpottavat erilaisten todennusmekanismien, kuten SAML:n, OAuthin ja JWT:n, käyttöönottoa. Tämä vähentää kehitystyön vaatimuksia ja nopeuttaa käyttöönottoa.
2. Korkea suorituskyky ja skaalautuvuus:
   .NET on optimoitu suorituskykyiseksi ja skaalautuvaksi alustaksi, mikä tekee siitä sopivan suurten käyttäjämäärien ja kuormitettujen palveluiden autentikointiratkaisuihin.
3. Integraatiot ja yhteensopivuus: .NET tarjoaa saumattomia integraatioita pilvipohjaisten palveluiden, kuten Microsoft Azure AD:n, kanssa. Kuten Microsoftin dokumentaatio korostaa:
   "ASP.NET Core integrates seamlessly with Azure Active Directory to enable secure authentication and authorization workflows for both cloud and hybrid environments."
   (Microsoft, 2023).
4. Turvallisuusominaisuudet:
   .NET sisältää kehittyneitä tietoturvaominaisuuksia, kuten TLS-salauksen ja sisäänrakennetun tuen salasanahallintaan. Lisäksi kehittäjät voivat hyödyntää tietoturvavalmiuksia, kuten uhkien tunnistusta ja estoa Microsoftin tarjoamien ratkaisujen avulla.
5. Käyttäjäystävällinen kehitysympäristö:
   .NET tukee monia ohjelmointikieliä, kuten C#, F# ja VB.NET, mikä tekee siitä

kehittäjille monipuolisen ja helposti omaksuttavan ympäristön. Visual Studio tarjoaa tehokkaat työkalut kehitykseen ja debuggaamiseen.

5.2 .NET-ympäristön tuki SAML, OAuthille ja JWT

.NET-ympäristö tarjoaa laajan tuen suosituimmille käyttäjätodennusstandardeille, kuten SAML:lle, OAuthille ja JWT:lle. Näiden integroiminen .NET-pohjaisiin sovelluksiin on helppoa, koska kehittäjillä on käytettävissään useita valmiita kirjastoja ja konfigurointityökaluja.

5.2.1 Tuki SAML:lle

.NET tukee SAML-standardia esimerkiksi Sustainsys.Saml2-kirjaston avulla. Tämä kirjasto yksinkertaistaa SAML-todennuksen toteutusta ja mahdollistaa sen integroinnin esimerkiksi Azure AD:hen. OASIS:n dokumentaatiossa korostetaan:
"SAML provides an XML-based framework for creating and exchanging security information between online services, enabling cross-domain authentication and Single Sign-On (SSO)."
(OASIS Security Services Technical Committee, 2005).

5.2.2 Tuki OAuthille

- Sisäänrakennettu tuki: ASP.NET Core sisältää valmiin tuen OAuth 2.0 -standardille ja sen valtuutusvirroille, kuten Authorization Code ja Client Credentials.
- Integraatiot: OAuth on helppo yhdistää Microsoftin pilvipalveluihin (esim. Azure AD) tai muihin kolmansien osapuolten palveluihin (esim. Google, Facebook).
- Sovellukset: OAuthia käytetään laajalti API-pääsyn hallintaan ja kolmannen osapuolen integraatioihin.

5.2.3 Tuki JWT:lle

ASP.NET Core sisältää valmiin JWT-middleware-ratkaisun, joka helpottaa tokenien validointia ja käyttöä REST API -ratkaisuissa. Microsoftin dokumentaatiossa todetaan:
"JSON Web Tokens (JWT) are widely used in ASP.NET Core applications to provide a stateless authentication mechanism for APIs, ensuring security and simplicity."
(Microsoft, 2023).

Yhteenvetona .NET tarjoaa kattavan tuen SAML:lle, OAuthille ja JWT:lle, mikä tekee siitä luotettavan ja monipuolisen alustan käyttäjätodennuksen toteuttamiseen. Näiden standardien tukeminen mahdollistaa turvalliset ja skaalautuvat autentikointiratkaisut niin paikallisissa kuin pilvipohjaisissa sovelluksissa.

6 VIENTIEN SALAUS JA TIETOTURVA

6.1 TLS-protokolla viestien salaamiseen

TLS (Transport Layer Security) on verkkoliikenteen salaamiseen tarkoitettu protokolla, joka takaa viestinnän luottamuksellisuuden, eheyden ja autentikoinnin osapuolten välillä. Se on keskeinen osa tietoturvaa REST API -rajapinnoissa ja käyttäjätodennuksessa, koska se suojaa tiedonsiirtoa salakuuntelulta ja manipuloinnilta.

Kuten Internet Engineering Task Force (IETF) toteaa TLS:n dokumentaatiossa:
"TLS provides a secure channel between two communicating peers on the Internet. It is widely used to secure web traffic, ensuring confidentiality, integrity, and authenticity."
(Dierks & Rescorla, 2008, s. 3).

6.1.1 TLS:n keskeiset ominaisuudet:

1. Salaus: TLS käyttää symmetristä salausta suojatakseen siirrettävän tiedon sisällön. Tämä estää viestien lukemisen ulkopuolisilta.
2. Autentikointi: TLS käyttää digitaalista varmenteita (SSL-sertifikaatteja) varmistaakseen, että yhteys on luotu oikean palvelimen kanssa.
3. Tietojen eheys: TLS varmistaa, ettei tiedonsiirron aikana viestien sisältöä ole muutettu tai manipuloitu.

6.1.2 TLS:n käyttö API-todennuksessa:

- TLS suojaa API-todennukseen liittyviä tunnuksia, kuten JWT:tä, OAuth-tunnuksia ja käyttäjätunnuksia sekä salasanoja.
- Käyttämällä TLS-protokollaa kaikki API-pyynnöt ja -vastaukset ovat salattuja, mikä minimoi hyökkäysmahdollisuudet, kuten "man-in-the-middle"-hyökkäykset.

6.2 HTTPS:n merkitys tietoturvassa

HTTPS (Hypertext Transfer Protocol Secure) on HTTP-protokollan suojattu versio, joka käyttää TLS-protokollaa tiedonsiirron salaamiseen. Se on kriittinen REST API -rajapintojen

ja käyttäjätodennuksen turvallisuuden kannalta, koska ilman HTTPS:ää kaikki siirrettävä tieto, kuten salasanat tai todennustunnukset, on alttiina sieppaukselle.

6.2.1 HTTPS:n hyödyt tietoturvassa:

1. Tietojen salaus: HTTPS suojaa siirrettävää dataa ulkopuolisilta, mikä estää arkaluonteisten tietojen vuotamisen.
2. Palvelimen autentikointi: HTTPS-sertifikaatti vahvistaa, että yhteys on luotu oikean palvelimen kanssa, mikä estää väärennettyihin palvelimiin kohdistuvat hyökkäykset.
3. Luottamuksen lisääminen: HTTPS tarjoaa käyttäjille mielenrauhan, sillä selaimet merkitsevät suojaamattomat yhteydet (HTTP) epäluotettaviksi.

Kuten National Institute of Standards and Technology (NIST) korostaa:
"Sensitive data transmitted over open networks, including credentials and personal information, must be encrypted using secure protocols such as HTTPS to prevent interception and tampering."
(Barker et al., 2020, s. 29).

6.2.2 HTTPS:n rooli API-todennuksessa:

API-avaimella toteutettu todennus edellyttää, että todennustunnus lähetetään jokaisessa API-pyynnössä. Tämä tarkoittaa, että internet-yhteys on jatkuvasti auki todennusprosessin aikana. Ilman HTTPS-salausta kaikki siirrettävä data, mukaan lukien API-avain, voidaan helposti siepata ja hyödyntää luvattomasti, esimerkiksi "man-in-the-middle"-hyökkäyksissä.

HTTPS on välttämätön suojauskerros, sillä se varmistaa, että:

1. API-avain ja muu data pysyvät salattuna tiedonsiirron aikana.
2. Hyökkääjä ei voi tarkkailla, muuttaa tai siepata pyynnöissä lähetettyjä todennustietoja.
3. Rajapinnan käyttö on luotettavaa ja täyttää modernit tietoturvavaatimukset.
4. Näin ollen, kaikissa API-rajapinnoissa, joissa käytetään todennustapoja, kuten API-avaimia, HTTPS ei ole vain suositeltava, vaan täysin välttämätön suojausmekanismi.

6.3 Symmetrisen ja asymmetrisen salauksen erot ja käyttö API-todennuksessa

Salausmenetelmiä käytetään API-todennuksessa suojaamaan tietoa ja varmistamaan sen eheys. Symmetrisellä ja asymmetrisellä salauksella on erilaiset käyttötarkoitukset ja vahvuudet.

6.3.1 Symmetrinen salaus

- Mekanismi: Käyttää yhtä ja samaa avainta sekä tietojen salaamiseen että purkamiseen.
- Etu: Nopea ja tehokas suurten tietomäärien käsittelyssä.
- Haaste: Avaimen jakelu turvallisesti osapuolten välillä voi olla haastavaa.
- Käyttö API-todennuksessa:
    - JWT:n allekirjoittaminen voidaan tehdä symmetrisellä avaimella (esim. HMAC-algoritmeilla).
    - Sopii hyvin sisäisiin järjestelmiin, joissa avaimen jakelu on hallittua.

6.3.2 Asymmetrinen salaus

- Mekanismi: Käyttää kahta avainta: julkista avainta tietojen salaamiseen ja yksityistä avainta niiden purkamiseen.
- Etu: Vähentää tarvetta jakaa yksityisiä avaimia, mikä parantaa turvallisuutta.
- Haaste: Laskennallisesti raskaampi kuin symmetrinen salaus.
- Käyttö API-todennuksessa:
    - SAML-assertiot ja JWT:t voidaan allekirjoittaa yksityisellä avaimella ja tarkistaa julkisella avaimella.
    - Soveltuu hyvin tilanteisiin, joissa osapuolet eivät luota toisiinsa täysin.

6.3.3 Symmetrisen ja asymmetrisen salauksen yhdistelmä

API-todennuksessa käytetään usein näiden menetelmien yhdistelmää:

- Asymmetristä salausta käytetään salausavainten turvalliseen jakeluun.
- Symmetristä salausta käytetään varsinaisen datan salaamiseen tehokkuuden vuoksi.

Näiden menetelmien ymmärtäminen ja oikea käyttö on tärkeää API-todennuksen tietoturvan varmistamisessa. Symmetristä salausmenetelmää käytetään yleisesti nopeuden vuoksi, kun taas asymmetristä salausta tarvitaan erityisesti viestinnän luotettavuuden ja avainten turvallisen jakelun takaamiseen.

7 PARHAAT KÄYTÄNNÖT KÄYTTÄJÄTODENNUSMENETELMISSÄ

7.1 Käyttäjätodennuksen toteuttaminen itse

Käyttäjätodennuksen toteuttaminen itse on suuri tehtävä, joka tarjoaa ainutlaatuisen mahdollisuuden räätälöidä järjestelmä vastaamaan tarkasti organisaation erityistarpeita. Se ei kuitenkaan ole vain tekninen haaste, vaan vaatii syvällistä tietoturvaosaamista, monipuolista suunnittelua ja jatkuvaa ylläpitoa. Vaikka lopputulos voi olla täysin hallinnassasi, prosessi sisältää monia kriittisiä osa-alueita, jotka edellyttävät huolellista toteutusta.

Ensimmäinen keskeinen osa-alue on käyttäjätietojen hallinta, joka muodostaa autentikoinnin perustan. Rekisteröinti- ja kirjautumisprosessien on oltava sujuvia ja turvallisia. Salasanojen käsittelyssä ei voida tinkiä: hajautusalgoritmien, kuten bcryptin, käyttö on välttämätöntä. Lisäksi järjestelmän on mahdollistettava turvallinen salasanan palautus, mikä usein aliarvioidaan riskitekijänä. Käyttäjäprofiilien hallinta ja roolipohjainen pääsynhallinta (RBAC) ovat tärkeitä, jotta järjestelmä voi tarjota käyttäjille oikeat oikeudet ilman ylimääräisiä riskejä.

Toiseksi, todennusprosessin turvallisuus on kaiken perusta. Käyttäjän tunnistaminen sähköpostin ja salasanan yhdistelmällä on edelleen yleistä, mutta lisäturvaa voidaan saavuttaa monivaiheisella todennuksella (MFA). Esimerkiksi kertakäyttöiset salasanat (OTP) tai push-ilmoitukset lisäävät merkittävästi järjestelmän vastustuskykyä hyökkäyksille. Istuntojen hallinta on myös ratkaisevaa: istuntojen automaattinen vanhentaminen ja uloskirjautuminen inaktiivisuuden aikana voivat estää monia tietoturvauhkia.

Tokenien hallinta ja validointi on kolmas olennainen osa. Esimerkiksi JWT-tokeneiden avulla voidaan varmistaa käyttäjän identiteetti tilattomasti. Tokenien voimassaolon hallinta ja uusiminen (esimerkiksi refresh-tokenien avulla) ovat tärkeitä käyttäjäkokemuksen sujuvuuden ja järjestelmän turvallisuuden ylläpitämiseksi. Tokenien manipuloinnin estäminen on kriittistä, ja järjestelmässä on oltava mekanismit tokenien mitätöimiseen mahdollisten väärinkäytösten varalta.

Hyökkäyksiltä suojautuminen on luonnollisesti yksi järjestelmän kulmakivistä. Bruteforcingin estäminen kirjautumisyrityksiä rajoittamalla ja IP-seurannan avulla on tehokas keino suojautua perinteisiltä hyökkäyksiltä. SQL-injektioiden ja XSS-hyökkäysten estämiseksi on varmistettava, että käyttäjän syöttämä data validoidaan ja käsitellään oikein. Lisäksi kaikki tiedonsiirto on salattava HTTPS-protokollaa käyttäen, mikä on ehdoton minimivaatimus moderneille järjestelmille.

Tietoturvan ylläpito ja ajantasaisuus ovat jatkuvia tehtäviä. Salausavainten hallinnan on oltava tarkkaa, ja niiden kierrätyksestä on pidettävä huolta, jotta vanhentuneet avaimet eivät jää järjestelmään. Hash-algoritmien ja muiden salausmenetelmien ajantasaisuus on varmistettava jatkuvasti. Säännölliset penetraatiotestit ja tietoturva-auditoinnit ovat olennainen osa ylläpitoa, jotta uusia haavoittuvuuksia voidaan havaita ja paikata ennen niiden hyödyntämistä.

Viimeisenä mutta ei vähäisimpänä, käyttökokemuksen optimointi on tärkeää. Kirjautumisen ja rekisteröitymisen on oltava käyttäjille sujuvaa, mutta samalla turvallista. Käyttäjäystävälliset ilmoitukset, kuten vahvistusviestit ja salasanan palautusohjeet, parantavat käyttökokemusta ja vähentävät tukipyyntöjä. Lisäksi järjestelmän on mahdollistettava kirjautuminen useilla laitteilla samanaikaisesti turvallisuudesta tinkimättä.

Kokonaisuudessaan itse toteutettu käyttäjätodennus on mahdollisuus luoda täysin räätälöity ratkaisu. Se kuitenkin tuo mukanaan merkittäviä haasteita, jotka vaativat aikaa, resursseja ja asiantuntemusta. Jokainen järjestelmän osa on kriittinen sen tietoturvan, käytettävyyden ja ylläpidettävyyden kannalta. On tärkeää varmistaa, että järjestelmä täyttää nykypäivän tietoturvavaatimukset ja tarjoaa käyttäjille luotettavan käyttökokemuksen.

Käyttäjätodennuksen toteuttaminen itse ei ole vain tekninen urakka, vaan se vaatii yritykseltä huomattavaa osaamista, resursseja ja sitoutumista. Pelkän teknisen toteutuksen lisäksi yrityksen on kyettävä vastaamaan moniin organisatorisiin ja toiminnallisiin haasteisiin, jotka liittyvät tällaisen järjestelmän ylläpitoon ja kehittämiseen.

Ensinnäkin, yrityksen on varmistettava, että sillä on riittävä asiantuntemus kolmella keskeisellä osa-alueella: ohjelmointi, tietoturva ja testaus. Ohjelmoinnin osalta kehittäjien on hallittava käyttäjätodennuksen periaatteet, kuten tokenipohjainen autentikointi, salaustekniikat ja istuntojen hallinta. Lisäksi heidän on pystyttävä rakentamaan järjestelmä, joka skaalautuu käyttäjämäärän kasvaessa.

Tietoturvan näkökulmasta vaaditaan asiantuntijoita, jotka ymmärtävät moderneja uhkia ja osaavat toteuttaa suojausmekanismeja. Tämä ei rajoitu pelkästään järjestelmän aloitusvaiheeseen, vaan tietoturvaa on jatkuvasti päivitettävä, koska uudet hyökkäysmenetelmät kehittyvät nopeasti. Testausosaaminen on kriittistä, sillä kattavat toiminnalliset ja tietoturvatestit ovat välttämättömiä ennen tuotantokäyttöä. Testaamatta jäänyt aukko voi aiheuttaa merkittäviä tietovuotoja tai palvelukatkoksia.

Järjestelmän ylläpito vaatii myös jatkuvaa valvontaa ja seurantaa. Yrityksen on investoitava valvontatyökaluihin, joilla seurataan kirjautumismääriä, epäonnistuneita kirjautumisyrityksiä ja mahdollisia poikkeamia käyttäjäkäyttäytymisessä. Ilman tällaista seurantaa mahdolliset tietomurrot voivat jäädä huomaamatta. Lisäksi valvonnasta saadut tiedot voivat olla hyödyllisiä järjestelmän kehittämisessä ja optimoinnissa.

Tietoturva-aukoista johtuvat käyttökatkokset ovat toinen merkittävä riski. Kun haavoittuvuuksia havaitaan, järjestelmän ylläpitäjien on usein tehtävä kiireellisiä korjauksia, mikä voi aiheuttaa katkoksia palvelun käytössä. Yrityksen on oltava valmis kommunikoimaan tällaisista tilanteista asiakkaille avoimesti ja ammattimaisesti. Lisäksi kriittisten korjausten tekeminen voi häiritä kehitystyötä ja muita projekteja, mikä korostaa resurssien tehokkaan hallinnan merkitystä.

Muut ei-tekniset näkökulmat liittyvät pitkälti yrityksen kulttuuriin ja prioriteetteihin. Itse toteutetun käyttäjätodennuksen ylläpitäminen vaatii, että tietoturva on yrityksessä jatkuvasti keskiössä. Tämä voi vaatia organisaation henkilöstön kouluttamista tietoturvasta, jotta kaikki työntekijät ymmärtävät käyttäjätodennuksen ja siihen liittyvän tietoturvan tärkeyden. Lisäksi päätös toteuttaa todennus itse voi vaikuttaa yrityksen liiketoiminnallisiin tavoitteisiin. Resurssien ohjaaminen tällaisen järjestelmän rakentamiseen ja ylläpitoon voi olla pois muista strategisesti tärkeistä projekteista.

Lopuksi on tärkeää pohtia itse toteutuksen pitkän aikavälin kestävyyttä. Tällaisen järjestelmän ylläpito vaatii jatkuvaa budjettia, henkilöstöä ja infrastruktuuria, jotka eivät välttämättä ole pieniä investointeja. Jos yritys päättää myöhemmin siirtyä ulkoistettuun ratkaisuun, tämä voi vaatia merkittäviä uudelleenjärjestelyjä, mikä tekee alkuperäisestä investoinnista osittain hukkaan menevän.

Yhteenvetona, itse toteutettu käyttäjätodennus on paljon enemmän kuin tekninen projekti – se vaatii syvällistä asiantuntemusta, vahvaa organisaatiokulttuuria ja resursseja sekä sitoutumista tietoturvan ylläpitoon pitkällä aikavälillä. Yrityksen on varmistettava, että sillä on tarvittavat taidot ja valmiudet hallita järjestelmää tehokkaasti sekä käsitellä siihen liittyvät riskit ja haasteet ammattimaisesti.

Taulukko 1. Yritysten käyttäjätodennusmenetelmien käyttöönoton tila maailmanlaajuisesti vuonna 2023.
(Lähde: Statista, 2023)

Käyttäjätodennusmenetelmä	Yritysten osuus (%)
Käyttäjätunnus ja salasana	68%
Ohjelmistopohjaiset tokenit (esim. kertakäyttöiset salasanat)	50%
Laitteistopohjaiset tokenit (esim. avaimenperät, USB-tokenit, älykortit)	34%
Out-of-band-autentikointi (esim. push-ilmoitukset, SMS, puhe)	30%
Biometrinen autentikointi	26%
Tokeniton autentikointi (esim. konteksti- tai mallipohjainen)	22%
Sosiaalisen identiteetin tunnukset (esim. LinkedIn, Facebook, X)	18%

Tämä taulukko havainnollistaa, että perinteinen käyttäjätunnus ja salasana -menetelmä on edelleen yleisin, mutta monivaiheiset ja biometriset todennusmenetelmät ovat yleistymässä yrityksissä.

7.2 Kolmannen osapuolen kirjastoja hyödyntäen

Kolmannen osapuolen kirjastojen käyttäminen käyttäjätodennuksessa on houkutteleva vaihtoehto yrityksille, jotka haluavat keskittyä ydinliiketoimintaansa ja samalla varmistaa, että autentikointi täyttää nykyaikaiset tietoturvavaatimukset. Valmiiden kirjastoiden ja standardien hyödyntäminen, kuten OAuth, SAML tai JWT, tarjoaa paitsi aikaa säästävän

myös tietoturvallisen ratkaisun. Tämä lähestymistapa ei kuitenkaan ole täysin ongelmaton, ja siihen liittyy monia tärkeitä näkökulmia, jotka yrityksen on otettava huomioon.

Kolmannen osapuolen kirjastoja hyödyntämällä yritys pääsee suoraan käsiksi testattuihin ja standardoituihin ratkaisuihin. Esimerkiksi OAuth-pohjaiset kirjastot, kuten Microsoft.Identity.Web, tarjoavat valmiit mekanismit käyttäjien autentikointiin, tokenien hallintaan ja pääsynhallintaan. Tämä vähentää huomattavasti kehittäjien työkuormaa, koska monimutkaiset toiminnot, kuten tokenien allekirjoittaminen ja validointi, käsitellään automaattisesti. Lisäksi nämä ratkaisut ovat suunniteltu yhteensopiviksi suurten ekosysteemien, kuten Microsoft Azure AD:n tai Google Identityn, kanssa, mikä helpottaa integraatioita monimutkaisissakin ympäristöissä.

Tietoturva on yksi merkittävimmistä syistä valita kolmannen osapuolen kirjasto. Nämä ratkaisut on usein kehitetty ja testattu kattavasti, ja ne sisältävät sisäänrakennettuja tietoturvamekanismeja, kuten bruteforcingin eston, tokenien elinkaaren hallinnan ja TLS-salauksen tuen. Koska tietoturvapäivitykset ja standardien kehitykset hoidetaan kirjastoja ylläpitävien tiimien toimesta, yritys voi olla varma, että järjestelmä pysyy ajan tasalla uusimpien tietoturvavaatimusten mukaisena. Tämä on erityisen tärkeää, koska uusien uhkien torjunta voi olla haastavaa pienelle tai keskisuurelle yritykselle ilman erillistä tietoturvatiimiä.

Vaikka kolmannen osapuolen kirjastoilla on lukuisia etuja, ne eivät poista kokonaan yrityksen vastuuta tietoturvasta. Kirjastojen tehokas käyttö edellyttää tarkkaa konfigurointia ja integraatiota. Esimerkiksi väärin asetettu OAuth-konfiguraatio voi johtaa tietoturva-aukkoihin, kuten liian laajoihin käyttöoikeuksiin tai puutteelliseen tokenin validointiin. Lisäksi yrityksen on huolehdittava, että kirjastoja päivitetään säännöllisesti, koska käyttämättä jääneet tietoturvapäivitykset voivat avata hyökkäysvektoreita.

Toinen tärkeä huomio on riippuvuus kolmannen osapuolen palveluista. Jos käytetty kirjasto tai palvelu lakkaa olemasta tuettu tai ylläpidetty, yritys voi joutua tilanteeseen, jossa sen on nopeasti siirryttävä toiseen ratkaisuun. Tämä voi aiheuttaa sekä taloudellisia kustannuksia että käyttökatkoksia. Lisäksi yrityksen tulisi arvioida tarkasti kolmannen osapuolen palvelun pitkäaikainen luotettavuus ja sen vaikutus liiketoimintaan. Vaikka tunnetut palveluntarjoajat, kuten Google tai Microsoft, ovat erittäin luotettavia, pienempien tai vähemmän tunnettujen ratkaisujen kohdalla riskit voivat olla suuremmat.

Kolmannen osapuolen kirjastojen käyttö liittyy myös tietynlaiseen kompromissiin hallinnan suhteen. Vaikka nämä ratkaisut ovat joustavia ja tukevat laajalti standardeja, ne eivät aina ole yhtä mukautettavissa kuin itse kehitetty järjestelmä. Jos yrityksellä on hyvin erityisiä vaatimuksia, esimerkiksi monimutkaisia liiketoimintalogiikoita autentikoinnissa, kolmannen osapuolen kirjastot eivät välttämättä tue näitä tarpeita suoraan. Tällöin yrityksen on arvioitava, onko valmis kompromissiin vai pitäisikö kehittää oma ratkaisu näiden kirjastojen rinnalle.

Ei-teknisestä näkökulmasta kolmannen osapuolen kirjastoilla toteutettu käyttäjätodennus voi helpottaa yrityksen toimintaa huomattavasti. Kehittäjien aika vapautuu ydinliiketoiminnan tukemiseen, koska autentikointiin liittyvä kehitys- ja ylläpitotyö jää minimaaliseksi. Tämä voi olla ratkaisevaa erityisesti pienille ja keskisuurille yrityksille, joilla ei ole resursseja rakentaa ja ylläpitää laajamittaisia tietoturvajärjestelmiä. Lisäksi asiakkaiden luottamus saattaa kasvaa, kun yritys käyttää laajalti tunnettua ja testattua autentikointiratkaisua.

Yrityksen on kuitenkin edelleen investoitava koulutukseen ja dokumentaation ymmärtämiseen, jotta se voi käyttää valittua ratkaisua tehokkaasti. On tärkeää, että tiimi tuntee valitun kirjaston toiminnan periaatteet ja tietää, miten sitä integroidaan yrityksen muihin järjestelmiin. Tämä vähentää riippuvuutta yksittäisistä kehittäjistä ja auttaa varmistamaan järjestelmän jatkuvuuden.

Yhteenvetona, kolmannen osapuolen kirjastojen hyödyntäminen on usein järkevä valinta, joka säästää aikaa, resursseja ja lisää tietoturvan tasoa. Se ei kuitenkaan ole täysin riskitön, ja yrityksen on huolehdittava sekä konfiguraation että ylläpidon tarkkuudesta. Tällaiset ratkaisut sopivat erityisen hyvin yrityksille, jotka arvostavat nopeaa käyttöönottoa ja haluavat hyödyntää hyväksi havaittuja standardeja ilman merkittävää kehitystyötä. Valinta tulisi tehdä yrityksen tarpeiden, resurssien ja pitkän aikavälin suunnitelmien perusteella, pitäen mielessä sekä tekniset että liiketoiminnalliset näkökulmat.

7.3 Uuden käyttäjätodentumisen toteutus REST API:lla

Siirtyminen autentikointijärjestelmässä nykyisestä yhdistelmästä (SAML shibboleth) uuteen (SAML, .NET Sustainsys.Saml2, OAuth, Azure) on sekä tekninen että strateginen päätös, jolla voi olla merkittävä vaikutus organisaation järjestelmien toimivuuteen, joustavuuteen ja ylläpidettävyyteen. Tavoitteena tällaisessa siirtymässä on usein parantaa järjestelmän

käytettävyyttä ja mahdollistaa uusien teknologioiden, kuten tokenipohjaisten valtuutusten ja REST API -rajapintojen, hyödyntäminen ilman, että nykyisen järjestelmän vahvuudet menetetään.

Nykyinen autentikointiyhdistelmä perustuu pitkälti SAML-protokollaan, joka on vakiintunut standardi yhden kirjautumisen (SSO) ja todennuksen siirron toteuttamiseen. Shibboleth, joka toimii Identity Providerina (IdP), hoitaa SAML-prosessin hallinnan ja toimii saumattomasti yhdessä .NET-ympäristössä käytettävän Sustainsys.Saml2-kirjaston kanssa. Tämä yhdistelmä on luotettava ja vakaa erityisesti organisaatioissa, jotka vaativat keskitettyä todennusta monimutkaisissa ympäristöissä. Shibbolethin suurimpia etuja on sen kyky hallita monimutkaisia SAML-integraatioita eri toimialueiden välillä, mikä tekee siitä ihanteellisen sisäisiin järjestelmiin. Kuitenkin, kuten usein käy, tämänkaltaiset ratkaisut tuovat mukanaan myös haasteita, erityisesti ylläpidon ja integraatioiden osalta. Shibboleth on raskas toteuttaa ja ylläpitää, ja sen konfigurointi vaatii syvällistä osaamista. Lisäksi se ei tue muita protokollia, kuten OAuthia, mikä rajoittaa sen soveltuvuutta nykyaikaisiin, dynaamisiin ympäristöihin.

Uudessa järjestelmässä Shibboleth korvataan Azure AD:llä, joka toimii sekä SAML- että OAuth-pohjaisena valtuutuspalvelimena. Tämä siirtymä tuo mukanaan monia etuja. Ensinnäkin Azure AD:n käyttö yksinkertaistaa arkkitehtuuria, koska erillisen Shibboleth-palvelimen ylläpidosta voidaan luopua. Lisäksi Azure AD tarjoaa tuen OAuth-protokollalle, mikä avaa mahdollisuuden tokenipohjaiseen autentikointiin ja valtuutukseen. OAuth on erityisen hyödyllinen REST API -ratkaisuissa, joissa tarvitaan kevyttä, tilatonta ja joustavaa pääsynhallintaa. Käyttämällä Sustainsys.Saml2-kirjastoa SAML-pohjaisten prosessien hallinnassa ja lisäämällä OAuth tuki järjestelmään, organisaatio voi hyödyntää molempien standardien vahvuuksia ilman merkittäviä kompromisseja.

Teknisen yksinkertaistamisen lisäksi uusi järjestelmä vähentää ylläpitotarvetta. Koska Azure AD on Microsoftin pilvipalvelu, sen ylläpito, päivitykset ja tietoturva hoidetaan Azure-palvelun toimesta. Tämä vähentää organisaation ylläpitotaakkaa ja resurssitarvetta. Lisäksi Azure AD mahdollistaa nopeamman integroinnin uusien pilvipalveluiden ja kolmansien osapuolten järjestelmien kanssa, mikä tekee siitä pitkällä aikavälillä kestävämmän ratkaisun.

Siirtymä ei kuitenkaan ole täysin ongelmaton. SAML- ja OAuth-protokollien yhteensovittaminen vaatii tarkkaa suunnittelua, sillä ne toimivat eri periaatteilla. SAML keskittyy väittämiin (assertioihin) ja luottamukseen IdP:n ja SP:n välillä, kun taas OAuth käyttää valtuutustokenia käyttäjän resurssien hallintaan. Näiden kahden yhdistäminen voi olla monimutkaista, erityisesti silloin, kun järjestelmän osien on toimittava saumattomasti yhdessä. Siirtymävaiheessa organisaation on myös huolehdittava koulutuksesta, jotta kehittäjät ja ylläpitäjät oppivat uuden järjestelmän toimintaperiaatteet ja osaavat käyttää sitä tehokkaasti.

Lisäksi Azure AD:n käyttö lisää riippuvuutta ulkoisista palveluista. Vaikka Microsoft on tunnettu luotettavuudestaan, pilvipohjaiset ratkaisut tuovat aina mukanaan riskin siitä, että palveluntarjoaja tekee muutoksia, jotka voivat vaikuttaa organisaation järjestelmään. Tämä vaatii jatkuvaa seurantaa ja varautumista muutoshallintaan.

Kaiken kaikkiaan siirtyminen nykyisestä autentikointijärjestelmästä uuteen on perusteltu erityisesti, jos organisaation tavoitteena on modernisoida infrastruktuuria ja mahdollistaa joustavampia käyttötapauksia, kuten tokenipohjaista pääsynhallintaa ja REST API -integraatioita. Uusi järjestelmä tuo mukanaan yksinkertaisemman arkkitehtuurin, vähentää ylläpitokustannuksia ja mahdollistaa uusien teknologioiden käyttöönoton. Siirtymävaihe kuitenkin vaatii huolellista suunnittelua ja resursseja, jotta mahdolliset riskit ja käyttökatkokset voidaan minimoida. Tämä muutos on enemmän kuin tekninen päivitys – se on strateginen valinta, joka vaikuttaa laajasti organisaation toimintatapaan, infrastruktuuriin ja resursseihin pitkällä aikavälillä.

8 TIETOTURVA JA PENETRAATIOTESTAUS

8.1 Windows 11 -käyttäjän todentaminen

Käyttäjätodennuksen murtamiseen liittyvät hyökkäykset ovat moninaisia, mutta tietyt menetelmät ovat yleisempiä. Esimerkiksi nykyisessä Windows 11:n PIN-koodilla tapahtuvassa kirjautumisessa on mekanismi, joka automaattisesti hyväksyy kirjautumisen, kun käyttäjä on syöttänyt oikean määrän numeroita. Tämä paljastaa hyökkääjälle arvokasta tietoa, kuten PIN-koodin tarkan pituuden. Jos hyökkääjä onnistuu varastamaan käyttäjän kannettavan tietokoneen sekä lompakon, jossa on esimerkiksi muistiinpanoja tai henkilökortti, hän voi nopeasti rajata mahdollisia PIN-koodeja ja salasanoja.

8.2 Penetraatiotestauksen merkitys

Penetraatiotestaus, eli pentesting, on yksi tärkeimmistä keinoista arvioida ja varmistaa järjestelmän tietoturva realistisissa hyökkäysskenaarioissa. Kuten OWASP korostaa:
"Penetration testing is essential for identifying exploitable vulnerabilities in applications and systems before attackers do. It provides actionable insights to improve defenses."
(OWASP, 2023).

Modernit järjestelmät, kuten ne, jotka hyödyntävät SAML-, OAuth- ja Azure-teknologioita, ovat monimutkaisia ekosysteemejä, joissa pienikin haavoittuvuus voi johtaa merkittäviin tietoturvariskeihin. Penetraatiotestauksen merkitys korostuu erityisesti tällaisissa ympäristöissä, joissa käyttäjätodennus ja pääsynhallinta ovat järjestelmän toiminnan ytimessä.

Pentestingin ensisijainen merkitys on löytää ja korjata haavoittuvuudet ennen kuin hyökkääjät voivat hyödyntää niitä. Tietoturva on aina reaktiivinen ja proaktiivinen prosessi, mutta ilman penetraatiotestausta järjestelmän heikkoudet voivat jäädä piiloon.

Kuten NIST toteaa:
"Organizations should conduct regular penetration tests to validate the effectiveness of their security controls and detect potential weaknesses."
(NIST SP 800-115, 2008).

Toinen tärkeä näkökulma on järjestelmän kestävyys todellisia hyökkäyksiä vastaan. Penetraatiotestaus jäljittelee tapoja, joilla hyökkääjät voisivat yrittää murtaa järjestelmän, ja antaa näin mahdollisuuden arvioida puolustusten vahvuutta. Tämä on erityisen tärkeää monimutkaisissa arkkitehtuureissa, joissa tietoturva riippuu useiden eri komponenttien, kuten Azuren, .NET-kirjastojen ja API-rajapintojen, saumattomasta yhteistyöstä. Yksittäinen virhe tai väärä konfiguraatio voi avata hyökkäysvektorin, joka vaarantaa koko järjestelmän.

Penetraatiotestaus ei kuitenkaan rajoitu vain järjestelmän tekniseen analyysiin. Se toimii myös organisaation oppimisvälineenä. Testausprosessi voi paljastaa, että tietoturvapolitiikat eivät ole riittävän kattavia tai että tiimin osaaminen tietyillä osa-alueilla, kuten tokenien salauksessa tai avainten hallinnassa, kaipaa vahvistusta. Näin penetraatiotestaus ei vain paranna järjestelmän turvallisuutta, vaan myös kehittää organisaation kyvykkyyttä vastata tuleviin tietoturvahaasteisiin.

Lisäksi penetraatiotestaus on usein välttämätön askel tietoturvastandardien noudattamisessa. Lainsäädäntö ja alan suositukset, kuten GDPR ja OWASP Top 10, edellyttävät tiettyjä toimenpiteitä tietoturvan varmistamiseksi. Penetraatiotestaus auttaa osoittamaan, että järjestelmä täyttää nämä vaatimukset. Tämä on erityisen tärkeää organisaatioille, jotka käsittelevät arkaluonteisia käyttäjätietoja tai luottavat ulkoisiin palveluihin, kuten Azureen.

Lopuksi, penetraatiotestauksen merkitys näkyy myös luottamuksen rakentamisessa. Käyttäjät ja asiakkaat arvostavat järjestelmiä, jotka voivat osoittaa olevansa turvallisia ja kestäviä hyökkäyksiä vastaan. Testauksen tuottamat tulokset – etenkin, jos ne osoittavat järjestelmän vahvuuksia – voivat parantaa yrityksen mainetta ja lisätä sidosryhmien luottamusta. Tämä luottamus on arvokasta erityisesti järjestelmissä, joissa käyttäjätodennus on kriittinen toiminto.

Yhteenvetona voidaan sanoa, että penetraatiotestaus ei ole vain valinnainen toimenpide, vaan olennainen osa modernin järjestelmän tietoturvaa. Se auttaa havaitsemaan ja korjaamaan haavoittuvuudet, parantaa järjestelmän kestävyyttä ja tukee organisaation tietoturvakulttuuria. Tämän vuoksi sen tulisi olla säännöllinen ja systemaattinen osa minkä tahansa autentikointijärjestelmän kehitystä ja ylläpitoa.

8.3 Testausprosessin vaiheet

Penetraatiotestaus on järjestelmällinen prosessi, joka koostuu useista tarkasti määritellyistä vaiheista. Jokainen vaihe tukee testauksen kattavuutta ja varmistaa, että järjestelmän haavoittuvuudet tunnistetaan ja dokumentoidaan. Testaus ei ole pelkästään tekninen operaatio, vaan se on myös tiedonkeruun, analyysin ja kommunikoinnin prosessi, joka auttaa kehittäjä- ja tietoturvatiimejä parantamaan järjestelmän suojausta.

8.3.1 Suunnittelu ja valmistelu

Kaikki tehokas testaus alkaa huolellisella suunnittelulla. Tässä vaiheessa määritellään testauksen tavoitteet ja rajat. Esimerkiksi SAML-, OAuth- ja Azure-pohjaisessa järjestelmässä voidaan päättää keskittyä SAML-väittämien eheyteen, tokenien hallintaan tai API-pyyntöjen turvallisuuteen. Tavoitteet auttavat suuntaamaan testauksen niihin alueisiin, joilla haavoittuvuuksilla olisi suurin vaikutus.

Valmistelussa myös testauksen laajuus sovitaan selkeästi. Tämä tarkoittaa esimerkiksi sitä, että päätetään, mitä järjestelmän osia saa testata ja millä menetelmillä. Samalla hankitaan tarvittavat luvat testausta varten, sillä pentesting on aina tehtävä luvallisesti ja tarkkaan määritellyissä puitteissa.

8.3.2 Tiedonkeruu

Tiedonkeruu, eli järjestelmän kartoittaminen, on olennainen osa testausta. Tässä vaiheessa pyritään ymmärtämään, miten järjestelmä toimii, ja keräämään tietoa hyökkäyksen suunnittelua varten. Tiedonkeruu voidaan jakaa kahteen osaan: passiiviseen ja aktiiviseen.

Passiivinen tiedonkeruu sisältää julkisesti saatavilla olevan tiedon keräämisen, kuten DNS-tietueiden, sertifikaattien tai järjestelmän ulkoisten rajapintojen analysoinnin. Aktiivinen tiedonkeruu puolestaan tarkoittaa suoria kyselyitä järjestelmälle. Esimerkiksi SAML-järjestelmässä voidaan analysoida väittämiä tai selvittää, miten järjestelmä käsittelee autentikointipyyntöjä. OAuth-ympäristössä puolestaan voidaan tutkia valtuutustokenin elinkaarta ja sen suojausta.

8.3.3 Haavoittuvuuksien tunnistaminen

Tässä vaiheessa keskitytään järjestelmän mahdollisten heikkouksien etsimiseen. Automaattiset työkalut, kuten OWASP ZAP tai Burp Suite, voivat tunnistaa yleisiä haavoittuvuuksia, kuten heikosti suojattuja API-pyyntöjä, SQL-injektioita tai puutteita salauksen käytössä.

Manuaalinen analyysi on kuitenkin yhtä tärkeää. Se täydentää automaattista analyysiä ja paljastaa monimutkaisempia ongelmia, kuten väärin konfiguroidut tokenien validointimekanismit tai session hallinnan puutteet. Esimerkiksi SAML-järjestelmässä voidaan testata, onko väittämien allekirjoitusten validointi oikein toteutettu. OAuth-ympäristössä voidaan tutkia, onko refresh-tokenien väärinkäyttö mahdollista.

8.3.4 Hyökkäyksen simulointi

Tämä on testauksen dynaaminen ja käytännönläheinen vaihe, jossa simuloidaan erilaisia hyökkäyksiä järjestelmää vastaan. Hyökkäyksen simulointi voi sisältää esimerkiksi:

- Bruteforcingin: Yritetään murtaa salasanoja tai tunnuksia hyödyntämällä erilaisia algoritmeja.
- Tokenien manipuloinnin: Testataan, voiko SAML-väittämiä tai OAuth-tokeneita muuttaa ilman, että järjestelmä havaitsee sen.
- Session kaappauksen: Simuloidaan tilannetta, jossa hyökkääjä yrittää käyttää varastettua sessiotokenia.
- API-pyyntöjen väärinkäytön: Pyritään tekemään luvattomia pyyntöjä ja tarkistamaan, pystyykö järjestelmä estämään ne.

Simulointi tuo esiin, kuinka hyvin järjestelmän puolustukset kestävät oikeita hyökkäystilanteita, ja auttaa havaitsemaan kriittisiä heikkouksia.

8.3.5 Tulosten analysointi

Kun testaus on suoritettu, kaikki löydetyt haavoittuvuudet analysoidaan niiden vakavuuden ja vaikutuksen perusteella. Tämä vaihe vaatii sekä teknistä että liiketoiminnallista näkökulmaa: jotkin löydökset, kuten tokenin manipulointi, voivat olla kriittisiä, kun taas esimerkiksi epäselvät virheilmoitukset voivat olla vähemmän kiireellisiä korjattavia.

Tulokset luokitellaan esimerkiksi seuraavasti:

- Kriittiset haavoittuvuudet: Välitöntä korjausta vaativat ongelmat, jotka voivat vaarantaa koko järjestelmän.
- Keskivakavat ongelmat: Korjaus on tärkeää, mutta ongelma ei aiheuta välitöntä uhkaa.
- Vähäiset haavoittuvuudet: Pienet ongelmat, jotka eivät todennäköisesti johda tietoturvaloukkauksiin.

8.3.6 Raportointi

Hyvä penetraatiotestaus on arvoton ilman selkeää raportointia. Raportti toimii dokumentaationa löydöksistä ja antaa tiimille selkeät suositukset korjaustoimista. Raportti sisältää:

- Yksityiskohtaiset tiedot löydetyistä haavoittuvuuksista.
- Kuvat tai lokit todisteena haavoittuvuuksien hyödyntämisestä.
- Suositukset korjauksille, mukaan lukien prioriteetit ja mahdolliset ratkaisutavat.

8.3.7 Korjaustoimet ja uudelleentestaus

Testauksen viimeinen vaihe on ehkä tärkein: löydösten korjaaminen ja uudelleentestaus. Haavoittuvuudet on korjattava mahdollisimman nopeasti, erityisesti kriittiset ongelmat. Kun korjaukset on tehty, järjestelmä on testattava uudelleen, jotta voidaan varmistaa, että ongelmat on ratkaistu ja ettei uusia haavoittuvuuksia ole syntynyt korjausten myötä.

8.3.8 Yleiset murtautumistavat

Taulukko 2.Yleisimpiä hyökkäystapoja lyhyellä kuvauksella.
(Lähde: FiCom ry, 2021)

Hyökkäystapa	Kuvaus
Tietojenkalastelu (phishing)	Huijausviestien avulla pyritään saamaan käyttäjät luovuttamaan tunnistetietonsa.
Salasanan arvaaminen	Yksinkertaisten tai yleisten salasanojen kokeileminen käyttäjätilien murtamiseksi.
Tietomurrot	Yksinkertaisten tai yleisten salasanojen kokeileminen käyttäjätilien murtamiseksi.
Haittaohjelmat	Ohjelmat, jotka keräävät käyttäjän tunnistetietoja ilman tämän tietoa.
Palvelunestohyökkäykset	Hyökkäykset, jotka estävät palveluiden käytön, mutta voivat myös paljastaa haavoittuvuuksia.

Taulukko, joka esittää yleisimpiä hyökkäystapoja ja niiden lyhyet kuvaukset.

Esimerkiksi Liikenne- ja viestintävirasto Traficomin Kyberturvallisuuskeskuksen mukaan vuonna 2020 yleisimpiä ilmoitettuja tietoturvaloukkauksia olivat huijaukset ja tietojenkalastelu.
Lisäksi *"Tietoturvan vuosi 2023" -katsauksessa todetaan, että kiristyshaittaohjelmat, huijausviestit ja palvelunestohyökkäykset olivat tunnusomaisia kyberturvallisuuden vuodelle 2023. https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/tietoturvan-vuosi-2023-katsaus-arvioi-uhkatason-pysyvan-kohonneena-myos-vuonna-2024*

On tärkeää huomata, että hyökkäystavat kehittyvät jatkuvasti, ja siksi organisaatioiden tulee säännöllisesti päivittää tietoturvakäytäntöjään ja kouluttaa henkilöstöään tunnistamaan ja torjumaan uusia uhkia.

8.3.9 Pohdinta

Tämä vaiheittainen lähestymistapa varmistaa, että penetraatiotestaus ei ole vain irrallinen tapahtuma, vaan jatkuva ja syvällinen prosessi. Jokainen vaihe – suunnittelusta tulosten analysointiin – tarjoaa arvokasta tietoa järjestelmän toiminnasta ja turvallisuudesta. Testaus ei ainoastaan tuo esiin heikkouksia, vaan myös vahvistaa järjestelmän luotettavuutta ja parantaa organisaation kykyä vastata tietoturvahaasteisiin. Hyvin toteutettu penetraatiotestaus on sijoitus, joka maksaa itsensä takaisin parempana turvallisuutena ja asiakkaiden luottamuksena.

9 VERTAILU: ITSE TOTEUTETTU VS. API KEY

9.1 Edut ja haasteet itse toteutetuissa ratkaisuissa

Käyttäjätodennuksen toteuttaminen itse voi olla houkutteleva vaihtoehto organisaatioille, jotka haluavat täydellisen hallinnan järjestelmänsä autentikointiprosesseista. Tämä lähestymistapa mahdollistaa järjestelmän räätälöinnin erityistarpeisiin ja sen integroimisen saumattomasti osaksi muita toimintoja. Kuitenkin itse toteutettu ratkaisu tuo mukanaan myös huomattavia haasteita ja riskejä, jotka on otettava huomioon.

9.1.1 Edut itse toteutetuissa ratkaisuissa

1. Täysi hallinta ja räätälöinti
   Itse toteutettu ratkaisu tarjoaa täydellisen hallinnan autentikointiprosessin kaikista osa-alueista. Järjestelmän voi suunnitella vastaamaan tarkasti organisaation tarpeita ilman kompromisseja, joita valmiiden ratkaisujen käyttö saattaisi edellyttää. Esimerkiksi tietyt yrityksen toimintaprosessit tai turvallisuusstandardit voivat vaatia erityisiä ominaisuuksia, joita ei ole saatavilla valmiissa kirjastoissa.
2. Ei ulkoisia riippuvuuksia
   Itse toteutetut ratkaisut eivät ole riippuvaisia kolmansien osapuolten palveluista tai kirjastoista. Tämä voi olla erityisen tärkeää organisaatioille, jotka haluavat minimoida ulkoisten palveluntarjoajien aiheuttamat riskit, kuten hinnoittelumuutokset, palvelukatkokset tai turvallisuusongelmat.
3. Syvällinen järjestelmäymmärrys
   Kehittäjät oppivat autentikointijärjestelmän yksityiskohdat ja toimintalogiikan, mikä voi parantaa järjestelmän ylläpitoa ja kehittämistä pitkällä aikavälillä. Syvällinen ymmärrys voi myös auttaa nopeammin reagoimaan uusiin turvallisuusuhkiin.
4. Ei lisenssimaksuja tai rajoituksia
   Valmiiden ratkaisujen käyttö voi joskus olla kallista, varsinkin jos ne perustuvat lisensseihin tai tilausmalleihin. Itse toteutettu ratkaisu ei tuo tällaisia lisäkustannuksia, vaan kustannukset rajoittuvat kehitykseen ja ylläpitoon.

9.1.2 Haasteet itse toteutetuissa ratkaisuissa

1. Korkea kehityskustannus ja ajallinen panostus
   Käyttäjätodennuksen kehittäminen alusta alkaen on monimutkainen ja aikaa vievä prosessi. Kehitystiimiltä vaaditaan huomattavaa osaamista salauksesta, tokenien hallinnasta, sessionhallinnasta ja muista tietoturvan osa-alueista. Tämä voi viivästyttää muita projekteja ja kasvattaa kustannuksia.
2. Tietoturvariskit
   Pienetkin virheet autentikointiprosessin toteutuksessa voivat johtaa merkittäviin tietoturva-aukkoihin. Esimerkiksi tokenin allekirjoituksen validoinnin puutteellinen toteutus tai epävarmat salausalgoritmit voivat altistaa järjestelmän hyökkäyksille. Tietoturvaosaamisen puute voi tehdä järjestelmästä haavoittuvan, vaikka sen toteutus näyttäisi toimivan oikein.
3. Jatkuva ylläpito ja päivitykset
   Itse toteutettu ratkaisu vaatii jatkuvaa ylläpitoa, erityisesti uusien tietoturvastandardien ja uhkien huomioimiseksi. Kehittäjien on oltava ajan tasalla uusista protokollista, kuten OAuth-päivityksistä tai tokenien hallinnan parhaista käytännöistä. Tämä vie aikaa ja resursseja, jotka voitaisiin käyttää muuhun kehitystyöhön.
4. Ei valmiita integraatioita
   Valmiit ratkaisut, kuten OAuth tai Azure AD, tarjoavat usein suoria integraatioita muihin palveluihin, kuten Google- tai Microsoft-kirjautumiseen. Itse toteutetuissa ratkaisuissa nämä integraatiot on rakennettava manuaalisesti, mikä lisää kehitystyön määrää.
5. Rajallinen skaalautuvuus
   Jos autentikointijärjestelmä on toteutettu ilman laajaa kokemusta, se voi olla vaikea skaalata suuremmalle käyttäjämäärälle. Valmiit ratkaisut, kuten Azure AD, ovat suunniteltu skaalautumaan suurille käyttäjämäärille, mutta itse toteutettu ratkaisu voi kohdata suorituskykyongelmia kasvavan käyttäjäkuorman kanssa.

9.2 Valmiiden autentikaatioratkaisujen edut ja haasteet

Valmiiden autentikaatioratkaisujen käyttö, kuten OAuth-pohjaiset palvelut tai Microsoftin Azure AD, on tullut yhä suositummaksi modernien järjestelmien kehittämisessä. Nämä ratkaisut tarjoavat valmiin infrastruktuurin käyttäjätodennukseen, tokenien hallintaan ja pääsynhallintaan, mikä säästää aikaa ja resursseja. Vaikka valmiit ratkaisut ovat usein turvallisia ja helppokäyttöisiä, ne eivät ole täysin vailla haasteita. Niiden käyttö vaatii huolellista arviointia, erityisesti järjestelmän pitkän aikavälin tarpeita ajatellen.

```
"AzureAd": {
  "Instance": "https://login.microsoftonline.com/",
  "Domain": "yourdomain.com",
  "TenantId": "your-tenant-id",
  "ClientId": "your-client-id",
  "ClientSecret": "your-client-secret",
  "CallbackPath": "/signin-oidc"
}

}
```

KUVA 1. "Esimerkki Azure AD -autentikaation konfiguroinnista appsettings.json-tiedostossa. Tämä rakenne mahdollistaa keskeisten tunnistetietojen, kuten ClientId:n ja TenantId:n, hallinnan keskitetysti ja turvallisesti."

9.2.1 Edut valmiista autentikaatioratkaisuista

1. Kattavat ominaisuudet heti käyttöön
   Valmiit autentikaatioratkaisut tarjoavat laajan valikoiman ominaisuuksia, jotka kattavat käyttäjätodennuksen keskeiset tarpeet. Esimerkiksi OAuth-toteutuksilla saadaan käyttöön valtuutustokenit, refresh-tokenit ja pääsynhallinnan eri tasot ilman, että niitä tarvitsee kehittää alusta alkaen. Lisäksi ratkaisut, kuten Azure AD, sisältävät valmiit mekanismit monivaiheiseen todennukseen (MFA), käyttäjien roolipohjaiseen pääsynhallintaan ja integroituihin auditointilokeihin.
2. Tietoturva ja standardien noudattaminen
   Valmiit ratkaisut noudattavat alan parhaita käytäntöjä ja tietoturvastandardeja, kuten OAuth 2.0, OpenID Connect ja SAML. Tämä tarkoittaa, että järjestelmää kehittävä organisaatio voi luottaa siihen, että autentikointiprosessi täyttää vaadittavat turvallisuusvaatimukset. Esimerkiksi

tokenit allekirjoitetaan ja salataan oletuksena, mikä estää niiden manipuloinnin tai väärinkäytön.

3. Nopea käyttöönotto
   Valmiit ratkaisut mahdollistavat autentikoinnin nopean käyttöönoton muutamilla konfiguraatioilla. Kehittäjien ei tarvitse rakentaa omaa järjestelmää tai ratkoa monimutkaisia turvallisuushaasteita, vaan he voivat keskittyä muihin projekteihin. Tämä säästää aikaa ja kustannuksia erityisesti projekteissa, joissa autentikointi ei ole ydinosa sovellusta.
4. Integraatiomahdollisuudet
   Monet valmiit ratkaisut tukevat suoraan kirjautumista kolmansien osapuolten palveluiden, kuten Google, Facebook tai Microsoft, kautta. Tämä vähentää tarvetta ylläpitää käyttäjien tunnuksia ja salasanoja itse. Lisäksi ratkaisut, kuten Azure AD, tarjoavat saumattoman integraation muihin pilvipalveluihin ja organisaation sisäisiin järjestelmiin.
5. Jatkuva ylläpito ja päivitykset
   Valmiiden ratkaisujen taustalla olevat tiimit vastaavat järjestelmien päivityksistä ja tietoturvapäivityksistä. Tämä vähentää organisaation omaa ylläpitotaakkaa ja varmistaa, että järjestelmä pysyy ajan tasalla uusimpien tietoturvastandardien mukaisena.

9.2.2 Haasteet valmiiden autentikaatioratkaisujen käytössä

1. Rajoitettu joustavuus
   Valmiit ratkaisut ovat suunniteltu yleiskäyttöön, mikä voi rajoittaa niiden mukauttamista erityistarpeisiin. Jos organisaation autentikointiprosessit vaativat jotain standardista poikkeavaa, esimerkiksi monimutkaisia roolipohjaisia sääntöjä, valmiiden ratkaisujen käyttäminen voi olla vaikeaa tai jopa mahdotonta ilman ylimääräistä räätälöintiä.
2. Riippuvuus ulkopuolisista palveluista
   Valmiiden ratkaisujen käyttö tarkoittaa usein, että järjestelmä on riippuvainen kolmannen osapuolen palveluntarjoajasta. Jos palvelu, kuten Azure AD, kohtaa käyttökatkoksen tai jos palveluntarjoaja muuttaa ehtojaan, organisaatio voi joutua ongelmiin. Tämä voi olla erityisen hankalaa, jos palvelu on kriittinen osa infrastruktuuria.

3. Kustannukset

   Monet valmiit ratkaisut perustuvat tilausmalleihin tai lisenssimaksuihin. Esimerkiksi Azure AD:n käyttö voi tulla kalliiksi suurissa organisaatioissa, joissa käyttäjiä ja resursseja on paljon. Lisäksi kustannukset voivat kasvaa, jos järjestelmää laajennetaan tai uusia ominaisuuksia otetaan käyttöön.

4. Oppimiskäyrä ja integrointi

   Vaikka ratkaisut ovat teknisesti valmiita käyttöön, niiden tehokas hyödyntäminen vaatii usein kehittäjiltä uuden teknologian opiskelua. Esimerkiksi OAuth-protokollan virtojen, kuten Authorization Code Flow:n, ymmärtäminen ja käyttöönotto voi viedä aikaa. Lisäksi integrointi olemassa oleviin järjestelmiin voi olla monimutkainen prosessi.

5. Tietoturvan jakaminen

   Kun valmiit ratkaisut hoitavat autentikoinnin, osa tietoturvasta on ulkoistettu palveluntarjoajalle. Tämä voi aiheuttaa huolta organisaatioissa, jotka haluavat säilyttää täyden hallinnan käyttäjätiedoistaan ja niiden suojauksesta.

10 YHTEENVETO JA POHDINTA

10.1 Tutkimustulokset

Tämän työn tavoitteena oli tutkia ja vertailla käyttäjätodennuksen toteutustapoja .NET-, SAML-, OAuth- ja Azure-teknologioita hyödyntävässä ympäristössä. Tarkastelimme erityisesti itse toteutettujen ratkaisujen ja valmiiden autentikaatiopalvelujen etuja ja haasteita, sekä suoritimme penetraatiotestauksen järjestelmän turvallisuuden arvioimiseksi. Tulokset osoittavat, että valmiit ratkaisut tarjoavat erinomaisen yhdistelmän turvallisuutta, nopeaa käyttöönottoa ja skaalautuvuutta, mutta niiden käyttö tuo mukanaan tiettyjä riippuvuuksia ja kustannuksia. Toisaalta itse toteutetut ratkaisut tarjoavat joustavuutta ja hallintaa, mutta vaativat merkittävää osaamista ja jatkuvaa ylläpitoa.

10.1.1 Keskeiset havainnot

1. Penetraatiotestauksen tulokset
   Testattu järjestelmä, joka perustui .NET-, SAML-, OAuth- ja Azure-teknologioihin, osoittautui turvalliseksi. Testauksessa ei löydetty haavoittuvuuksia, ja järjestelmän eri osat, kuten SAML-assertiot, OAuth-tokenien hallinta ja Azure AD -integraatio, täyttivät tietoturvan parhaat käytännöt. Tämä vahvisti, että hyvin toteutetut ja konfiguroidut valmiit autentikaatioratkaisut voivat olla turvallisia myös monimutkaisissa ekosysteemeissä.
2. Itse toteutetut ratkaisut: vahvuudet ja heikkoudet
   Itse toteutettu autentikointi tarjoaa täydellisen hallinnan ja mahdollisuuden räätälöidä järjestelmä vastaamaan organisaation erityistarpeita. Tämä on erityisen arvokasta organisaatioille, jotka käsittelevät herkkiä tietoja tai joilla on tiukat turvallisuusvaatimukset. Kuitenkin tällaisen ratkaisun toteuttaminen vaatii laajaa osaamista, erityisesti tietoturvasta, ja tuo mukanaan merkittävän ylläpitovastuun. Lisäksi kehityskustannukset voivat olla huomattavia, ja virheiden riski on suurempi verrattuna valmiisiin ratkaisuihin.
3. Valmiit autentikaatioratkaisut: tehokkuus ja rajoitukset
   Valmiit ratkaisut, kuten OAuth ja Azure AD, tarjoavat turvallisuuden lisäksi laajat ominaisuudet heti käyttöön. Ne ovat erityisen sopivia organisaatioille, jotka tarvitsevat luotettavaa ja nopeaa autentikointia ilman suurta

kehitystyötä. Näiden ratkaisujen suurimmat rajoitukset liittyvät joustavuuden puutteeseen ja riippuvuuteen ulkoisista palveluista. Lisäksi kustannukset voivat kasvaa merkittävästi käyttäjämäärien ja lisäominaisuuksien lisääntyessä.

10.1.2 Pohdinta

Tutkimuksen perusteella voidaan todeta, että valmiit autentikaatioratkaisut ovat useimmille organisaatioille käytännöllisin valinta. Ne tarjoavat nykyaikaisiin vaatimuksiin sopivan yhdistelmän turvallisuutta, tehokkuutta ja integroitavuutta. Niiden avulla voidaan myös hyödyntää alan parhaita käytäntöjä ja välttää suuria investointeja tietoturvaosaamiseen ja järjestelmän ylläpitoon.

Kuitenkin organisaatioiden, joilla on ainutlaatuisia tarpeita tai korkea riippuvuus käyttäjätodennuksesta, kannattaa harkita itse toteutetun ratkaisun tai valmiin ratkaisun räätälöinnin hyödyntämistä. Näissä tapauksissa on tärkeää arvioida käytettävissä olevat resurssit ja osaaminen realistisesti, sillä itse toteutettu ratkaisu tuo mukanaan huomattavia riskejä, jos tietoturvaa ei kyetä toteuttamaan oikein.

Penetraatiotestaus osoitti, että valmiiden ratkaisujen turvallisuustaso on korkea, kunhan järjestelmät konfiguroidaan oikein ja niitä ylläpidetään säännöllisesti. Tämä vahvistaa, että ulkoisten palveluntarjoajien käyttö voi olla tietoturvallinen vaihtoehto, kunhan organisaatio noudattaa huolellisuutta valinnassa ja toteutuksessa.

10.2 Suositukset organisaatioille

Käyttäjätodennus on järjestelmän turvallisuuden ja käytettävyyden perusta, ja sen valinnalla on kauaskantoisia vaikutuksia. Tutkimuksen tulosten ja penetraatiotestauksen perusteella on mahdollista laatia suosituksia, jotka auttavat organisaatioita valitsemaan oikean lähestymistavan autentikointiratkaisuihinsa. Suositukset koskevat erityisesti niitä organisaatioita, jotka harkitsevat itse toteutettujen ja valmiiden autentikaatioratkaisujen välillä, ja niitä, jotka toimivat teknologioilla kuten .NET, OAuth, SAML ja Azure.

1. Arvioi tarpeet ja resurssit realistisesti

Ennen kuin autentikaatioratkaisua valitaan, on tärkeää kartoittaa organisaation tarpeet ja käytettävissä olevat resurssit. Jos autentikointi ei ole liiketoiminnan ydin, valmiit ratkaisut,

kuten Azure AD tai OAuth-pohjaiset palvelut, ovat usein järkevin valinta. Ne tarjoavat nopeasti käyttöön otettavia ja turvallisia ratkaisuja, jotka vapauttavat organisaation keskittymään muihin projekteihin.

Toisaalta, jos autentikointi liittyy suoraan liiketoiminnan kriittisiin osiin, kuten räätälöityihin asiakasratkaisuihin, voi olla perusteltua harkita itse toteutettua ratkaisua. Tällöin on kuitenkin varmistettava, että organisaatiolla on riittävä tietoturvaosaaminen ja resurssit järjestelmän ylläpitoon.

2. Hyödynnä valmiita ratkaisuja ensisijaisesti

Valmiit autentikaatioratkaisut, kuten Azure AD, ovat luotettavia, laajasti testattuja ja noudattavat alan parhaita käytäntöjä. On suositeltavaa, että organisaatiot hyödyntävät näitä ratkaisuja aina, kun ne täyttävät organisaation tarpeet. Tämä lähestymistapa säästää aikaa, resursseja ja vähentää tietoturvariskejä, sillä ulkopuoliset palveluntarjoajat huolehtivat päivityksistä ja uusimpien uhkien torjunnasta.

Valmiiden ratkaisujen käyttöä tukee myös niiden tarjoama laaja integraatiotuki, kuten kirjautumismahdollisuus kolmansien osapuolten palveluiden (esim. Google tai Microsoft) kautta. Tämä parantaa käyttökokemusta ja vähentää organisaation tarvetta hallinnoida käyttäjätunnuksia ja salasanoja itse.

3. Panosta konfiguroinnin ja ylläpidon huolellisuuteen

Vaikka valmiit ratkaisut tarjoavat monia etuja, niiden käyttö ei poista organisaation vastuuta tietoturvasta. On suositeltavaa panostaa konfigurointiin ja ylläpitoon, erityisesti seuraavilla osa-alueilla:

- SAML ja OAuth: Varmista, että SAML-assertioiden validointi ja OAuth-tokenien hallinta on toteutettu oikein, eikä ylimääräisiä oikeuksia anneta.
- Azure AD: Hyödynnä Multi-Factor Authentication (MFA) ja varmista, että käyttöoikeuspolitiikat ovat tarkasti määriteltyjä.
- API-rajapinnat: Varmista, että kaikki API-pyynnöt vaativat autentikoinnin ja valtuutuksen, ja ettei virheilmoitukset paljasta liikaa tietoa hyökkääjille.

4. Toteuta penetraatiotestaukset säännöllisesti

Tietoturva ei ole kertaluontoinen prosessi. Organisaatioiden tulisi toteuttaa säännöllisiä penetraatiotestauksia varmistaakseen, että autentikointijärjestelmä on turvallinen myös järjestelmän muutosten ja uusien uhkien valossa. Suosittelemme erityisesti testien suorittamista, kun:

- Järjestelmään lisätään uusia ominaisuuksia tai integraatioita.
- Tietoturvastandardeissa tapahtuu merkittäviä muutoksia.
- Havaitaan uusia tietoturvauhkia, kuten haavoittuvuuksia käytetyissä kirjastoissa tai protokollissa.

Säännöllinen testaus auttaa myös pitämään yllä tietoturvatietoisuutta organisaatiossa ja varmistamaan, että mahdolliset ongelmat havaitaan ennen kuin hyökkääjät voivat hyödyntää niitä.

5. Varaudu tietoturvapoikkeamiin

Vaikka autentikointijärjestelmä olisi suunniteltu ja toteutettu huolellisesti, tietoturvapoikkeamat ovat aina mahdollisia. Suositeltavaa, että organisaatioilla on selkeä toimintasuunnitelma, joka sisältää:

- Nopeat toimenpiteet haavoittuvuuksien korjaamiseksi.
- Käyttäjien informoimisen tietovuodoista avoimesti ja nopeasti.
- Varmuuskopiot ja varajärjestelmät, jotta palvelut voidaan palauttaa nopeasti tietoturvaongelman sattuessa.

6. Seuraa teknologian ja standardien kehitystä

Autentikointiteknologiat kehittyvät jatkuvasti, ja uusien standardien (esim. OAuth 2.1) tai menetelmien (esim. FIDO2) käyttöönotto voi parantaa sekä turvallisuutta että käyttäjäkokemusta. Organisaatioiden tulisi seurata alan kehitystä aktiivisesti ja päivittää järjestelmiään tarvittaessa.

10.3 Johtopäätös

Tämän työn tulokset osoittavat, että valmiit autentikaatioratkaisut, kuten OAuth ja Azure AD, tarjoavat useimmille organisaatioille parhaan yhdistelmän turvallisuutta, tehokkuutta ja integroitavuutta. Ne soveltuvat erityisesti ympäristöihin, joissa nopea käyttöönotto, standardien noudattaminen ja helppo ylläpito ovat etusijalla. Toisaalta itse toteutetut

ratkaisut voivat olla perusteltuja, kun organisaation tarpeet ovat poikkeuksellisen vaativia tai kun täysi hallinta on välttämätöntä.

Penetraatiotestauksen tulokset korostavat, että valmiiden ratkaisujen turvallisuustaso on korkea, kunhan järjestelmät konfiguroidaan ja ylläpidetään oikein. Tämä tukee suositusta, että organisaatioiden tulisi ensisijaisesti hyödyntää testattuja ja dokumentoituja ratkaisuja välttääkseen tarpeettomia riskejä ja kehityskustannuksia. Jos kuitenkin valitaan itse toteutettu ratkaisu, on varmistettava, että organisaatiolla on riittävä tietoturvaosaaminen ja resurssit ylläpitämään järjestelmää turvallisesti.

Yhteenvetona voidaan todeta, että autentikaatioratkaisun valinta tulisi perustua organisaation tarpeisiin, resursseihin ja tietoturvavalmiuksiin. Jatkuva testaus, teknologian kehityksen seuraaminen ja tietoturvapoikkeamien ennakointi ovat avainasemassa riippumatta valitusta lähestymistavasta. Tällainen kokonaisvaltainen lähestymistapa auttaa organisaatioita ylläpitämään turvallisuutta ja luottamusta sekä tukee pitkän aikavälin tavoitteiden saavuttamista.

11 Lainatut lähteet

Hardt, D. (2012). *The OAuth 2.0 Authorization Framework (RFC 6749)*. IETF. Saatavilla: https://www.rfc-editor.org/rfc/rfc6749

OASIS Security Services Technical Committee. (2005). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS. Saatavilla: https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT) (RFC 7519)*. IETF. Saatavilla: https://www.rfc-editor.org/rfc/rfc7519

Dierks, T., & Rescorla, E. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2*. Internet Engineering Task Force (IETF). Saatavilla: https://datatracker.ietf.org/doc/html/rfc5246

Barker, E., Barker, W., Burr, W., Polk, W., & Smid, M. (2020). *NIST Special Publication 800-57 Part 1 Revision 5: Recommendation for Key Management*. National Institute of Standards and Technology (NIST). Saatavilla: https://nvlpubs.nist.gov/

Microsoft. (2023). *ASP.NET Core Authentication Overview*. Saatavilla: https://learn.microsoft.com

OASIS Security Services Technical Committee. (2005). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. Saatavilla: https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

OWASP. (2023). *OWASP Penetration Testing*. Saatavilla: https://owasp.org

NIST. (2008). *Technical Guide to Information Security Testing and Assessment (SP 800-115)*. Saatavilla: https://csrc.nist.gov/publications

ISO/IEC. (2022). *ISO/IEC 27001:2022 – Information security management systems*. Saatavilla: https://www.iso.org

https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/tietoturvan-vuosi-2023-katsaus-arvioi-uhkatason-pysyvan-kohonneena-myos-vuonna-2024

Taulukko 1. https://www.statista.com/statistics/1441144/companies-authentication-methods-deployment-status-worldwide/?utm_source

Taulukko 2. https://ficom.fi/ajankohtaista/uutiset/tietoisuus-tietoturvauhkista-kasvaa/?utm_source