Kanditutkielma

Tietojenkäsittelytieteen kandiohjelma

Salasanojen tiivisteiden murtamisen menetelmät ja niiltä suojautuminen

Riku Rauhala

29.12.2023

Matemaattis-luonnontieteellinen tiedekunta

Helsingin yliopisto

Yhteystiedot

PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

Sähköpostiosoite: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen kandiohjelma

Tekijä — Författare — Author

Riku Rauhala

Työn nimi — Arbetets titel — Title

Salasanojen tiivisteiden murtamisen menetelmät ja niiltä suojautuminen

Ohjaajat — Handledare — Supervisors

Nikolaj Tatti, Iiro Kumpulainen

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kanditutkielma	29.12.2023	21 sivua

Tiivistelmä — Referat — Abstract

Salasanat ovat yleisin käyttäjän todentamiseen käytetty menetelmä. Salasanat tallennetaan tavallisesti tietokantaan kryptografisina tiivisteinä, jotka on laskettu yksisuuntaisen tiivistefunktion avulla. Tiivisteitä sisältäviä tietokantoja päätyy toisinaan tietomurtojen seurauksena ulkopuolisten haltuun. Koska salasanoilla suojattava tieto saattaa olla hyvinkin arvokasta, verkossa toimivilla rikollisilla on mielenkiintoa alkuperäisten salasanojen selvittämiseen vuotaneiden tiivisteiden avulla.

Tässä kandidaatin tutkielmassa perehdytään yleisimpiin salasanojen tiivisteiden murtamisen menetelmiin sekä niiltä suojautumiseen. Tutkielma luo kattavan yleiskatsauksen aihepiiriin ja tarjoaa työkaluja sovelluskehittäjille salasanojen turvalliseen säilyttämiseen sekä käyttäjille vahvan salasanan valintaan. Tutkielmassa esitellään salasanojen tiivisteiden murtamisen perusteet ja käydään läpi yleisimmät hyökkäysmenetelmät: väsytyshyökkäys, sanakirjahyökkäys, sääntöihin perustuva mukautettu hyökkäys sekä sateenkaaritaulukko.

Ohjelmistokehittäjää suositellaan tallentamaan palvelunsa salasanat tietokantaan tiivisteinä, jotka on laskettu käyttämällä nykyaikaista tiivistealgoritmia kuten bcrypt tai Argon2. Käyttäjän tulisi valita salasanakseen mahdollisimman suuresta merkistöstä mahdollisimman pitkä, satunnaisesti generoitu merkkijono. Sovelluskehittäjän tulee lisätä palveluunsa tuki monivaiheiselle tunnistautumiselle ja käyttäjän on osattava ottaa se käyttöön.

ACM Computing Classification System (CCS)
Security and privacy → Security services → Authentication
Security and privacy → Software and application security

Avainsanat — Nyckelord — Keywords

salasanat, tietoturva, todennus

Säilytyspaikka — Förvaringsställe — Where deposited

Helsingin yliopiston kirjasto

Muita tietoja — övriga uppgifter — Additional information

Sisällys

1 Johdanto

Internetin aikakaudella rekisteröityminen erilaisiin palveluihin on muodostunut tavalliseksi ja usein jopa välttämättömäksi osaksi ihmisten elämää. Ohjelmistokehityksen näkökulmasta eräs keskeinen haaste on käyttäjän todentaminen. Rekisteröitynyt käyttäjä on pystyttävä tunnistamaan luotettavasti.

Salasanat ovat yleinen käyttäjän todentamiseen käytetty menetelmä. Vaikka ohjelmiston tietoturva olisi muuten kunnossa, salasanaan perustuva kirjautumistoiminnallisuus tarjoaa pahantahtoiselle hyökkääjälle mahdollisuuden yrittää murtautua sisään palveluun ja päästä luvatta käsiksi käyttäjän tietoihin. Puutteet salasanojen säilyttämisessä tai huonosti valittu salasana voivat vaarantaa koko palvelun tietoturvan.

Salasanojen käyttöön liittyvät ongelmat johtuvat usein käyttäjästä itsestään. Ihmisillä on taipumusta valita lyhyitä, helposti arvattavia salasanoja, jotka saattavat jopa löytyä sellaisenaan sanakirjasta (Shen et al., 2016). Salasanaan saattaa myös sisältyä julkisesti saatavilla olevaa käyttäjään liittyvää henkilökohtaista tietoa, kuten vuosilukuja tai lemmikkien nimiä (Bosnjak et al., 2018).

Vaikka vaihtoehtoisia tapoja käyttäjän todentamiseen on kehitetty, salasanat ovat edelleen yleisimmin käytetty autentikointimenetelmä erityisesti verkkosovelluksissa ja tietokoneohjelmissa (Zimmermann ja Gerber, 2020). Salasanat ovat houkutteleva kohde verkossa toimiville rikollisille, sillä niitä käytetään usein suojaamaan arvokasta tietoa (Shen et al., 2016). Tämän vuoksi salasanojen tietoturva on erityisen tärkeä ja mielenkiintoinen tutkimuskohde. Koska selkeää korvaajaa salasanoille suosituimpana todennusmenetelmänä ei ole lähitulevaisuudessa nähtävissä, on sekä sovelluskehittäjien että käyttäjien etu tuntea yleisimmät hyökkäystekniikat ja osata suojautua niitä vastaan.

Tässä tutkielmassa perehdytään salasanojen kryptografisten tiivisteiden (engl. *cryptographic hash*) murtamisen menetelmiin ja niiltä suojautumiseen sekä sovelluskehittäjän, että käyttäjän näkökulmista. Tutkimusta ohjaa kaksi tutkimuskysymystä, joita tutkielman rakenne noudattaa. Tutkimuskysymykset on määritelty seuraavasti.

> **Tutkimuskysymys 1**: Mitä menetelmiä voidaan käyttää salasanojen tiivisteiden murtamiseen?
>
> **Tutkimuskysymys 2**: Miten näiltä menetelmiltä voidaan suojautua?

Aluksi luvussa 2 perehdytään siihen, mitä tiivisteiden murtaminen oikeastaan tarkoittaa ja mihin murtamiseen tähtäävät menetelmät perustuvat. Luvussa 3 syvennytään aiheeseen käsittelemällä tarkemmin yleisimpiä tiivisteiden murtamiseen käytettyjä keinoja. Luku vastaa tutkimuskysymykseen 1. Pääpaino on teknisissä, ohjelmallisesti toteutettavissa tavoissa murtaa salasanojen tiivisteitä. Salasanojen murtamisesta (engl. *password cracking*) puhuttaessa tarkoitetaan nimenomaan tiivisteiden murtamista, eli alkuperäisen käyttäjän valitseman salasanan selvittämistä tiivisteitä vertailemalla. Salasanoja voidaan pyrkiä selvittämään myös muilla keinoilla, kuten käyttäjää manipuloimalla (engl. *social engineering*) tai näppäintallentimilla (engl. *keylogger*), mutta nämä keinot eivät kuulu tämän tutkielman piiriin.

Kun tavanomaiset tiivisteiden murtamisen menetelmät on käsitelty, luvussa 4 esitetään miten niiltä voidaan suojautua ja vastataan tutkimuskysymykseen 2. Sekä käyttäjän, että sovelluskehittäjän näkökulmat on otettu huomioon, sillä molempien toiminnalla on vaikutus sovelluksen tietoturvaan. Suojautumiskeinojen kohdalla myös pohditaan sitä, mikä on sovelluskehittäjän vastuulla ja mitä käyttäjän on itse osattava ottaa huomioon.

Lopuksi luvussa 5 käydään vielä lyhyesti läpi keskeiset tulokset ja pohditaan niiden vaikutusta.

2 Tiivisteiden murtaminen

Salasanoja ei säilytetä tietokannassa sellaisenaan luettavassa muodossa eli selvätekstinä (engl. *plaintext*), vaan salatussa muodossa. Salaaminen toteutetaan tavallisesti laskemalla käyttäjän valitsemasta salasanasta kryptografinen tiiviste, joka tallennetaan tietokantaan. Näin toimitaan, jotta mahdollisesti vuotanut salasanatietokanta ei tarjoa hyökkääjälle suoraan pääsyä jokaisen käyttäjän kirjautumistietoihin. Myöskään palvelun ylläpitäjällä ei ole mahdollisuutta lukea käyttäjiensä salasanoja. Tavallista kaksisuuntaista salausta ei käytetä, sillä salausta ei ole missään vaiheessa tarkoitus purkaa. Kirjautumistilanteessa käyttäjän syöttämästä salasanasta lasketaan uudelleen tiiviste, jota verrataan tietokannasta löytyvään vastineeseen.

Hyökkääjällä tarkoitetaan tämän tutkielman kontekstissa henkilöä tai ryhmää, joka pyrkii murtamaan salasanojen tiivisteitä. Salasanan tiivisteen murtamisella tarkoitetaan menetelmää, jolla yritetään saada salatussa muodossa säilytetty, alkuperäinen salasana selville. Käyttäjällä tarkoitetaan salasanan valinnutta palveluun rekisteröitynyttä henkilöä.

Tiivistefunktiot soveltuvat hyvin salasanojen suojaamiseen, sillä operaatio on yksisuuntainen: tiivisteen perusteella ei ole mahdollista päätellä alkuperäistä käyttäjän valitsemaa salasanaa. Sisäänkirjautumistoiminnallisuus toteutetaan siten, että kirjautumista yrittävän käyttäjän syöttämästä salasanasta lasketaan uudelleen tiiviste, jota verrataan tietokantaan tallennettuun tiivisteeseen. Kryptografisen tiivisteen laskemiseen on olemassa erilaisia menetelmiä ja hyvin valitulla algoritmilla voidaan hidastaa hyökkääjän toimintaa (Andersson, 2023). Tähän palataan myöhemmin vielä luvussa 4.1.

Jotta salasana voidaan murtaa, tulee hyökkääjällä ensin olla pääsy kohteena olevan salasanan tiivisteeseen tai listaan tiivisteitä. Tämän tutkielman puitteissa ei perehdytä keinoihin murtautua tietokantoihin, vaan hyökkääjän käytössä oletetaan olevan lista tiivisteistä, joita yritetään murtaa. Listoja salasanojen tiivisteistä on julkisesti saatavilla Internetissä erilaisten tietomurtojen ja -vuotojen jäljiltä. Suurimmat verkkoon päätyneet listat sisältävät jopa satoja miljoonia salasanojen tiivisteitä (Heen ja Neumann, 2017). Nämä listat ovat hyökkääjälle erittäin hyödyllisiä, sillä niitä voidaan käyttää sellaisinaan murtamisen apuna tai niitä analysoimalla voidaan muodostaa listoja yleisimmistä salasanoista sekä päätellä, millaisia salasanoja ihmisillä on tapana valita.

On myös huomioitava, että salasanojen murtaminen ei aina tarkoita laittoman toiminnan

valmistelua. Ohjelmistokehittäjät voivat itse testata kehittämänsä palvelun tai käyttämiensä algoritmien heikkouksia. Samoja murtotekniikoita voidaan käyttää myös omien unohtuneiden salasanojen löytämiseksi. Toiminta kuitenkin muuttuu laittomaksi, jos toisen käyttäjän murretulla salasanalla kirjaudutaan luvattomasti palveluun (Finlex, 2015).

Tiivisteiden murtamista kutsutaan *offline*-hyökkäykseksi kun se tapahtuu hyökkääjän omalla laitteistolla (Grassi et al., 2017). Tiivisteiden murtaminen paikallisesti on erittäin tehokasta verrattuna *online*-hyökkäykseen, jossa hyökkäys toteutettaisiin verkon yli suoraan palvelua vastaan. Internetin kautta palvelun omaa autentikointipalvelua vastaan suoritettavat hyökkäykset on mahdollista automatisoidusti havaita tietoliikennettä analysoimalla. Automatisoituja online-hyökkäyksiä voidaan yrittää torjua esimerkiksi kirjautumisyritysten määrää rajoittamalla tai vaatimalla käyttäjää läpäisemään CAPTCHA-testi (OWASP, 2023b). Offline-hyökkäyksessä hyökkääjän toimintaa rajoittavat vain käytössä oleva laskentateho, käytetyt murtamismenetelmät sekä tiivistefunktion ominaisuudet.

Seuraavassa luvussa (luku 3) esiteltävät murtamismenetelmät perustuvat algoritmiin 2.1 (OWASP, 2023c).

Algoritmi 2.1 Salasanan selvittäminen tiivisteitä vertaamalla

```
1: procedure VERTAATIIVISTEITÄ(tiivisteKohde, salasanaKandidaatit)
2:     for each salasanaKandidaatti in salasanaKandidaatit do
3:         tiivisteKandidaatti ← LASKETIIVISTE(salasanaKandidaatti)
4:         if tiivisteKandidaatti = tiivisteKohde then
5:             Output: salasanaKandidaatti
6:             break
7:         end if
8:     end for
9: end procedure
```

Koska tiivistefunktio on yksisuuntainen, eli tiivisteestä ei voida suoraan salausta purkamalla palauttaa alkuperäistä arvoa, perustuu murtaminen kryptografisten tiivisteiden vertailemiseen. Algoritmissa 2.1 funktio *vertaaTiivisteitä* saa parametreina murrettavan salasanan tiivisteen *tiivisteKohde* sekä listan selväkielisistä salasanoista *salasanaKandidaatit*. Jokaiselle salasanakandidaatille lasketaan tiiviste käyttämällä samaa algoritmia eli tiivistefunktiota, jolla kohteena olevan salasanan tiiviste on laskettu. Tiivisteitä vertaillaan keskenään ja jos ne täsmäävät, on salasana murrettu onnistuneesti. Kun salasana on murrettu, laskenta voidaan keskeyttää ja mahdollisesti aloittaa uudestaan jollakin toisella

tiivisteellä, mikäli murrettavia salasanoja on useampia.

Teoriassa kaksi keskenään erilaista salasanaa voivat tuottaa saman tiivisteen, mutta nykyaikaisten tiivistealgoritmien kohdalla tämä on erittäin harvinaista. Lähtökohtaisesti täsmäävä tiiviste tarkoittaa siis onnistunutta murtamisyritystä. Algoritmi tiivistefunktion laskemiseksi tulee valita siten, että tiivisteiden törmääminen (engl. *hash collision*) on laskennallisesti hyvin epätodennäköistä (Menezes et al., 1997). Kyseessä on itse asiassa eräs tiivistefunktiolle asetettava keskeinen vaatimus. Tähän palataan tarkemmin luvussa 4.1.

Salasanan tiivisteen murtamiseen tähtäävät hyökkäysmetodit perustuvat pohjimmiltaan havaintoon siitä, että halutaan muodostaa mahdollisimman optimaalinen lista salasanakandidaateista. Lista kokeiltavista salasanoista voi olla määritelty valmiiksi ennalta tai se voidaan muodostaa ohjelmallisesti. Seuraavassa luvussa tutustutaan tarkemmin erilaisiin hyökkäysmenetelmiin eli keinoihin murtaa salasanojen tiivisteitä.

3 Murtamisen menetelmät

Salasanojen selvittämiseksi on kehitetty erilaisia menetelmiä. Näistä murtamistekniikoista voidaan käyttää myös tietoturvan parissa yleistä termiä hyökkäys. Tässä luvussa käsitellään muutamia yleisiä kryptografisten tiivisteiden murtamiseen käytettyjä menetelmiä. Erilaisia tekniikoita on kehitetty, sillä ne lähestyvät samaa ongelmaa hieman eri näkökulmasta. Riippuen kohteena olevan salasanan rakenteesta, hyökkäyksen teho voi olla vaihteleva. Hyökkääjä saattaakin hyödyntää useita tekniikoita suorittamalla niitä peräkkäin tai niiden toimintaa yhdistelemällä.

Hyökkäykset suoritetaan käytännössä aina automaattisesti tietokoneen avulla, sillä vain yhden tiivisteen murtamiseksi saatetaan joutua kokeilemaan miljoonia tai miljardeja erilaisia salasanakandidaatteja. Riittävän vahvan salasanan kohdalla on myös mahdollista, että mikään tunnettu menetelmä ei pysty murtamaan sitä kohtuullisessa ajassa.

3.1 Väsytyshyökkäys

Väsytyshyökkäys (engl. *brute-force attack*) on yksinkertainen salasanojen murtamiseen käytetty menetelmä. Hyökkääjältä ei vaadita erityistä nokkeluutta, vaan hyökkäys perustuu käytössä olevan laitteiston laskentatehoon (Bosnjak et al., 2018). Väsytyshyökkäyksen ideana on kokeilla järjestelmällisesti jokaista mahdollista vaihtoehtoa.

Vaikka väsytyshyökkäys eli *raakahyökkäys* on luonteeltaan systemaattista ja hyvin yksinkertaisesti toteutettavissa, voidaan sitäkin tehostaa kiinnittämällä huomiota käytettyyn merkistöön. Mikäli palvelun salasanoille asettamat vaatimukset ovat hyökkääjän tiedossa, voidaan tätä tietoa käyttää hyödyksi hyökkäyksessä. Jos esimerkiksi palvelu on rajoittanut salasanoissaan sallittujen merkkien joukon aakkosnumeeriseen merkistöön, voidaan myös väsytyshyökkäyksessä käytetty merkkien joukko rajoittaa vastaavasti.

Teoriassa väsytyshyökkäyksellä on mahdollista murtaa mikä tahansa salasana. Hyökkäyksen tehokkuus kuitenkin laskee nopeasti salasanan pituuden kasvaessa. Väsytyshyökkäyksellä voidaankin murtaa tehokkaasti lyhyitä salasanoja, mutta pidempien salasanojen murtamiseksi tarvitaan muita keinoja.

3.2 Sanakirjahyökkäys

Sanakirjahyökkäys (engl. *dictionary attack*) on väsytyshyökkäystä hienostuneempi menetelmä, joka perustuu ennalta määritellyn sanalistan käyttöön. Hyökkäystä varten valikoitu lista eli sanakirja voi olla esimerkiksi suomen tai englannin kielen sanakirja tai se voi olla luettelo tiedossa olevista oikeista salasanoista. Tällainen lista voi sisältää esimerkiksi yleisimmät salasanat tai tietomurtojen perusteella oikeassa käytössä olleita salasanoja.

Sanakirjahyökkäyksen tehokkuutta on mahdollista optimoida valitsemalla sopiva salasanakandidaattien lista. Jos esimerkiksi murrettavan salasanan valinneen käyttäjän kieli on tiedossa, voidaan hyökkäyksen onnistumisen todennäköisyyttä kasvattaa valitsemalla lista salasanoja, joiden tiedetään olevan kohteena olevan käyttäjän kieltä puhuvien ihmisten valitsemia (Bonneau, 2012).

Tehokas sanakirja useiden salasanojen murtamiseen on lista yleisistä salasanoista. Eräs näistä listoista on salasanojen hallintaohjelmistoa kehittävän Nordpassin kokoama lista, joka perustuu useisiin julkisesti saatavilla oleviin vuotaneita salasanoja sisältäviin lähteisiin (Nordpass, 2023). Listan viisikymmentä yleisintä salasanaa on esitelty taulukossa 3.1. Kuten huomataan, yleisimmät salasanat koostuvat pitkälti pelkästään numeroista ja kirjaimista. Listaa hallitsevat peräkkäiset numerosarjat (kuten "123456") ja yleiset kirjautumiseen liittyvät luonnollisen kielen sanat (kuten "admin" ja "password"). Myös tavallisella QWERTY-näppäimistöllä lähekkäin sijaitsevien merkkien muodostamat salasanat (kuten "qwerty" ja "1q2w3e4t") ovat suosittuja.

Taulukko 3.1: 50 yleisintä salasanaa (Nordpass, 2023)

#	Salasana	#	Salasana	#	Salasana	#	Salasana	#	Salasana
1	123456	11	UNKNOWN	21	1111	31	abc123	41	123123123
2	admin	12	1234567	22	P@ssw0rd	32	Aa@123456	42	11223344
3	12345678	13	123123	23	root	33	abcd1234	43	987654321
4	123456789	14	111111	24	654321	34	1q2w3e4r	44	demo
5	1234	15	Password	25	qwerty	35	123321	45	12341234
6	12345	16	12345678910	26	Pass@123	36	err	46	qwerty123
7	password	17	000000	27	******	37	qwertyuiop	47	Admin@123
8	123	18	admin123	28	112233	38	87654321	48	1q2w3e4r5t
9	Aa123456	19	********	29	102030	39	987654321	49	11111111
10	1234567890	20	user	30	ubnt	40	Eliska81	50	pass

Salasanojen lähteenä olevasta palvelusta ja salasanan valinneiden käyttäjien käyttämästä kielestä ja maantieteellisestä sijainnista riippuen yleisimpien salasanojen järjestys voi olla erilainen. Yleisimmät salasanat noudattavat kuitenkin pitkälti samaa kaavaa. Esimerkiksi OWASP:n kokoaman listan kärki eroaa hieman Nordpassin vastaavasta, mutta koostuu sekin tavallisista englanninkielisistä sanoista sekä numeroista (OWASP, 2023a). Tulokset eivät ole juurikaan yllättäviä, sillä satunnaisesti valittu riittävän pitkä merkkijono on todennäköisemmin ainutlaatuinen eikä päädy yleisimpien salasanojen listalle. Vahvan salasanan valitsemista käsitellään tarkemmin luvussa 4.4.

3.3 Mukautettu hyökkäys

Salasanoja voidaan yrittää myös murtaa erilaisten sääntöjen avulla. Sääntöihin perustuva hyökkäys (engl. *rule-based attack*) on hienostuneempi hyökkäysmuoto, jossa käyttäjien mahdollisesti salasanoihinsa tekemät muutokset otetaan huomioon. Koska sääntöihin perustuva hyökkäysmenetelmä on hyökkääjän vapaasti konfiguroitavissa, sitä voidaan kutsua myös mukautetuksi tai sovelletuksi hyökkäykseksi.

Mukautettua hyökkäystä voidaan käyttää murtamaan salasanoja, jotka eivät löydy sellaisinaan sanakirjasta tai joiden murtaminen veisi liian kauan pelkän väsytyshyökkäyksen avulla. Sääntöjen avulla voidaan myös generoida sanakirjasta löytyvistä sanoista nopeasti erilaisia variaatioita.

Eräs mahdollinen käyttöskenaario on hyökkäyksen mukauttaminen kohteena olevan palvelun tiedossa olevaa salasanakäytäntöä (ks. luku 4.5) vastaan. Jos esimerkiksi palvelun salasanakäytäntö vaatii, että salasanasta on löydyttävä neljä numeroa, saattaa hyökkääjä arvella käyttäjän mahdollisesti valinneen jonkin tavallisen sanakirjasta löytyvän sanan ja lisänneen sen perään oman syntymävuotensa. Tällöin voidaan suorittaa muuten tavallinen sanakirjahyökkäys, mutta lisätä jokaisen sanan perään luku esimerkiksi väliltä 1900–1999. Tässä tapauksessa kokeiltavien mahdollisten salasanojen määrä ei kasva eksponentiaalisesti, kuten väsytyshyökkäyksen tapauksessa, vaan säännön avulla kokeiltavien salasanakandidaattien määrä on vain satakertainen.

Tiivisteiden murtamiseen käytetty Hashcat-ohjelmisto tarjoaa useita sääntöjä, joilla salasanoista voidaan luoda nopeasti erilaisia variaatioita. Tällaisia sääntöjä ovat esimerkiksi kaikkien kirjainten muuttaminen isoiksi, tiettyjen kirjainten korvaaminen halutuilla erikoismerkeillä ja numeroiden lisääminen sanan alkuun tai loppuun (Hashcat, 2023).

3.4 Sateenkaaritaulukko

Salasanoja on myös mahdollista selvittää käyttämällä listaa valmiiksi lasketuista tiivisteistä. Tätä menetelmää kutsutaan nimellä sateenkaaritaulukko (engl. *rainbow table*). Kyseessä on kompromissi laskentatehon ja tallennustilan välillä: sateenkaaritaulukkoa käyttämällä hyökkääjän ei tarvitse käyttää laskentatehoa tiivisteiden laskemiseen, mutta käytössä on oltava riittävästi tallennustilaa taulukon arvojen säilyttämiseen. Sateenkaaritaulukon toiminta perustuu siihen, että sama tiivistefunktio laskee annetulle salasanalle aina saman tiivisteen. Nämä tiivisteet voidaan ottaa talteen ja käyttää myöhemmin hyökkäyksen toteuttamisessa. Nykyaikainen sateenkaaritaulukko esiteltiin vuonna 2003 perustuen jo vuonna 1980 julkaistuun menetelmään (Oechslin, 2003).

Sateenkaaritaulukon toiminta perustuu havaintoon siitä, että tiivistefunktio palauttaa annetulla syötteellä aina saman arvon. Yleisesti käytetyille tiivistefunktioille voidaan siis laskea valmiiksi taulukoita, jotka sisältävät erilaisia salasanakandidaatteja vastaavat tiivisteet. Näitä taulukoita voidaan säilöä ja käyttää myöhemmin hyödyksi useissa hyökkäyksissä. Erityisesti jonkin tietyn heikon salasanan tiiviste tietyllä tiivistealgoritmilla on siis mahdollisesti laskettu jo valmiiksi ennen kuin mitään tietomurtoa on edes tapahtunut.

On huomattava, että sateenkaaritaulukkoa käytettäessä menetelmä alkuperäisen salasanan löytämiseksi muuttuu hieman. Koska tiivisteet on laskettu jo valmiiksi, niitä ei tarvitse enää erikseen laskea jokaisen salasanakandidaatin kohdalla. Algoritmi 3.1 esittelee mahdollisen tavan käyttää sateenkaaritaulukkoa. Käytetystä tietorakenteesta riippuen selväkielinen salasana voi olla osa taulukkoa tai voidaan toteuttaa funktio *HaeSalasana*, joka noutaa tiivistettä vastaavan selväkielisen salasanan kun täsmäävä tiiviste on löytynyt.

Algoritmi 3.1 Salasanan selvittäminen sateenkaaritaulukon avulla

```
1: procedure VERTAATIIVISTEITÄ(tiivisteKohde, tiivisteKandidaatit)
2:     for each tiivisteKandidaatti in tiivisteKandidaatit do
3:         if tiivisteKandidaatti = tiivisteKohde then
4:             Output: HAESALASANA(tiivisteKandidaatti)
5:             break
6:         end if
7:     end for
8: end procedure
```

4 Suojautuminen

Salasanojen tiivisteiden murtamiselta voidaan myös suojautua. Osa vastuusta on sovelluskehittäjällä, jonka tehtävä on suojata tietokantaan tallennettuja salasanoja sekä estää niiden joutuminen vääriin käsiin. Vuotanut salasanatiivisteiden tietokanta tekee aina murtamisesta huomattavasti helpompaa hyökkääjälle, jonka toimintaa rajoittavat vuodon jo tapahduttua enää käytettävissä olevan laitteiston laskentateho sekä hyökkääjän oma tietotaito ja oveluus. Hyökkäys verkon yli palvelua vastaan on paljon monimutkaisempaa ja helpommin torjuttavissa.

Sovelluskehittäjän on mahdollista yrittää ohjata käyttäjää valitsemaan vahva salasana sekä asettaa salasanalle vaatimuksia. Salasanan valitseminen on kuitenkin lopulta käyttäjän itsensä vastuulla. Koska tiedetään, että käyttäjät valitsevat yleensä heikkoja salasanoja, on sovelluskehittäjän vastuulla kuitenkin korostunut käytännön merkitys.

4.1 Tiivistefunktiot

Kryptografisen tiivisteen laskemiseen on kehitetty erilaisia algoritmeja eli tiivistefunktioita, joiden soveltuvuus salasanojen turvaamiseksi on vaihteleva. On ohjelmistokehittäjän vastuulla valita käyttötarkoitukseen sopiva algoritmi. Salasanat tulee säilyttää tietokannassa aina tiivisteinä, ei selvätekstinä. Näin tulee menetellä vaikka sovelluskehittäjä uskoisi omaan kykyynsä pitää tietokannan data suojassa. Tiivistealgoritmin valitseminen ja käyttäminen on välttämätöntä, sillä on syytä varautua ennalta salasanatietokannan vuotamiseen.

Tiivisteiden laskemiseen käytetyn algoritmin tulee täyttää kolme vaatimusta ollakseen turvallinen (Dang, 2012). Ensinnäkin, tiivisteiden törmäämisen tulee olla käytännössä mahdotonta. Tätä ominaisuutta kutsutaan törmäyskestävyydeksi (engl. *collision resistance*). Vaatimuksen tarpeellisuus on ilmeistä, sillä kahden eri salasanan ei tulisi missään tilanteessa tuottaa samaa kryptografista tiivistettä. Mikäli näin olisi, mahdollistaisi tiivisteiden vertailemiseen perustuva kirjautumistoiminnallisuus käyttäjän todentamisen myös jollakin muulla merkkijonolla kuin alkuperäisellä käyttäjän valitsemalla salasanalla.

Toiseksi, on oltava laskennallisesti mahdotonta (engl. *computationally infeasible*) löytää tii-

visteen perusteella alkuperäistä arvoa (Dang, 2012). Toisin sanoen, tiivistefunktion on oltava yksisuuntainen operaatio. Tätä ominaisuutta kutsutaan alkukuvakestävyydeksi (engl. *preimage resistance*).

Vastaavasti kolmas tiivistefunktiolle asettava vaatimus on, että tiivisteen perusteella ei ole mahdollista löytää jotakin toista selväkielistä salasanaa, jolla olisi sama tiiviste. (Dang, 2012). Tätä ominaisuutta nimitetään toiseksi alkukuvakestävyydeksi (engl. *second preimage resistance*).

Salasanojen säilyttämiseen käytettävän tiivistefunktion on lisäksi oltava hieman yllättäen toiminnaltaan riittävän hidas (OWASP, 2023c). Kun algoritmin suorittaminen on laskennallisesti ja ajallisesti vaativampaa, hyökkääjän toiminta hidastuu. Vaatimus perustuu siihen, että yhden tiivisteen laskeminen käyttäjän kirjautuessa sisään ei aiheuta vielä huomattavia ongelmia, mutta miljoonia tiivisteitä laskevan hyökkääjän toiminta hidastuu huomattavasti.

Vaatimukset täyttäviä, suositeltuja nykyaikaisia algoritmeja ovat Argon2id, scrypt sekä bcrypt (OWASP, 2023c).

4.2 Kryptografinen suola ja pippuri

Eräs tapa hidastaa hyökkääjän toimintaa on käyttää kryptografista suolaa. Tiivisteen laskeminen suolan avulla varmistaa sen, että tiiviste ei ole ennalta arvattavissa vaikka käytetty tiivistefunktio ja alkuperäinen salasana tunnettaisiin. Tämä tarkoittaa sitä, että kryptografisen suolan käyttäminen tarjoaa suojaa erityisesti sateenkaaritaulukkoa vastaan, sillä suolan kanssa laskettu tiiviste ei voi löytyä taulukosta jo valmiiksi laskettuna. Kuten luvussa 3.4 todettiin, sateenkaaritaulukon toiminta perustuu siihen, että sama tiivistefunktio tuottaa samalla salasanalla aina saman tiivisteen. Taulukossa 4.1 on esimerkki siitä miten kaksi käyttäjää saattavat valita saman salasanan, jolloin tietokantaan tallentuu molempien kohdalle täysin identtinen tiiviste. Taulukon tiiviste on kuvitteellinen esimerkki eikä sitä ole laskettu millään oikealla tiivistefunktiolla.

Taulukko 4.1: Tiivisteen muodostuminen identtisellä salasanalla

Käyttäjä	Salasana	Tiiviste
Matti	salainen123	Z59HDNEC5TZJMNK1N2JA4SBD6Y9HD
Maija	salainen123	Z59HDNEC5TZJMNK1N2JA4SBD6Y9HD

Koska kahdella identtisellä salasanalla on samalla tiivistefunktiolla laskettuna myös sama kryptografinen tiiviste, hyökkääjä voi yhden tiivisteen murtamalla selvittää kerralla jopa tuhansien käyttäjien salasanan. Kun jokaiselle salasanalle valitaan mielivaltaisesti eri suola, saadaan identtisille salasanoille muodostettua eri tiivisteet. Tämä hidastaa hyökkääjän toimintaa, sillä nyt jokainen tiiviste on murrettava erikseen. Taulukosta 4.2 nähdään, että suolaa käyttämällä kahden käyttäjän valitsema identtinen salasana johtaa keskenään täysin erilaisten tiivisteiden muodostumiseen.

Taulukko 4.2: Tiivisteen muodostuminen suolaa käyttämällä

Käyttäjä	Salasana	Suola	Tiiviste
Matti	salainen123	oP811rA3e@g9	R325CGJ19OEO9DTZYNPTIX7P8WJJ2
Maija	salainen123	Gy453.95£?mY	99D1FFES5Z5ILYTD8FKAO3H5U88TC

Kryptografinen suola voi olla mikä tahansa mielivaltaisesti valittu merkkijono. Se voidaan esimerkiksi generoida satunnaisesti käyttäjälle tämän rekisteröityessä ja tallentaa tietokantaan sellaisenaan tiivisteen lisäksi. Se miten suolaa käytetään riippuu käytössä olevan tiivistefunktion toteutuksesta. Eräs mahdollinen tapa on konkatenoida käyttäjän syöttämä salasana ja suola, ja laskea näin saadusta merkkijonosta tiiviste. Koska myös suola tallennetaan tietokantaan, voidaan käyttäjän kirjautuessa sisään operaatio toistaa samalla kaavalla ja toteuttaa todennus normaalisti vertaamalla tiivisteitä.

Ollakseen riittävän turvallinen, on suolan oltava tarpeeksi pitkä. Mikäli suola on liian lyhyt, on tiiviste edelleen haavoittuvainen sateenkaaritaulukolle. Riittävän pitkä suola takaa sen, että jokaisen suolan ja salasanan yhdistelmän kerääminen sateenkaaritaulukkoon muuttuu laskennallisesti liian vaativaksi. Toinen suolan turvalliselle käytölle asetettava vaatimus on, että jokaiselle käyttäjälle valitaan uniikki suola. Mikäli samaa suolaa käytetään kaikille käyttäjille, muodostuu identtisille salasanoille jälleen identtiset tiivisteet. Samalla myös sateenkaaritaulukon käyttö on edelleen vaivatonta, sillä hyökkääjä joutuu vain laskemaan taulukon arvot kertaalleen uudelleen.

Idea kryptografisesta suolasta voidaan myös viedä vielä pidemmälle paikkaamaan sen mahdollisia heikkouksia. Mikäli suola päätetään pitää kokonaan salaisena ja säilyttää erillään tiivisteestä, sitä kutsutaan nimellä pippuri.

Pippurin käyttöön pätevät samat säännöt kuin suolaankin, mutta pippuri voi olla sama kaikille käyttäjille. Tällöin pippurin tulisi kuitenkin olla riittävän pitkä, jotta sen arvoa ei pystytä selvittämään väsytyshyökkäyksellä. Ero kryptografisen suolan ja pippurin välillä syntyy siitä, että salasanatietokannan päädyttyä hyökkääjän käsiin, voidaan tiivisteitä

lähteä murtamaan suolan avulla, mutta pippurin kanssa tämä ei onnistu jos tiiviste ja pippuri on oikeaoppisesti säilytetty eri paikoissa. Suolaa ja pippuria voidaan myös käyttää yhdessä lisäsuojauksen saamiseksi.

4.3 Avaimen venytys

Hyökkääjän toimintaa voidaan hidastaa avaimen venytyksen (engl. *key stretching*) avulla. Venytyksen toteuttamiseen on olemassa erilaisia menetelmiä, jotka kaikki pyrkivät tekemään tiivisteen murtamisesta laskennallisesti vaativampaa. Venytystä käytetään usein yhdessä kryptografisen suolan kanssa.

Eräs tapa toteuttaa avaimen venytys on laskea tiiviste uudelleen ja uudelleen sisäkkäisillä funktiokutsuilla. Tällöin venytys voidaan toteuttaa jo käytössä olevilla työkaluilla. Olkoon h jokin tiivistefunktio. Nyt tiiviste voidaan laskea käyttäjän syöttämästä salasanasta sekä mielivaltaisesta suolasta, eli tiiviste $= h(h(h(h(h(\text{salasana}, \text{suola})))))$.

Lopullinen tiiviste saadaan siis laskemalla tiivisteen tiiviste viisi kertaa sisäkkäisillä funktiokutsuilla. Iteraatioiden määrä on ohjelmistokehittäjän päätettävissä. Vaikka avaimen venyttäminen tekee yksittäisen tiivisteen laskemisesta hitaampaa, sillä ei ole kirjautuvan käyttäjän kannalta huomattavaa vaikutusta. Venytyksen merkitys paljastuu kun tiivisteitä yritetään murtaa: yhden tiivisteen murtaminen on nyt laskennallisesti ja ajallisesti vaativampaa ja hyökkäyksen teho hidastuu riippumatta käytetystä hyökkäysmenetelmästä.

4.4 Vahva salasana

Käyttäjän näkökulmasta merkityksellisin tapa suojautua on mahdollisimman vahvan salasanan valitseminen. Kuitenkin myös sovelluskehittäjän on tunnettava salasanan vahvuuden mittarit sekä vahvan salasanan piirteet. Käyttäjän syöttämä salasana tavallisesti validoidaan ennen sen hyväksymistä ja täten salasanan hyväksyminen tai hylkääminen on lopulta sovelluskehittäjän päätäntävallassa. Sovelluskehittäjä voi pyrkiä ohjaamaan käyttäjää vahvemman salasanan valitsemiseen salasanankäytäntöjen avulla.

4.4.1 Entropia

Salasanan vahvuudelle on kehitetty erilaisia mittareita. Eräs näistä mittareista on salasanan entropia, joka mittaa hyökkääjän kohtaamaa epävarmuutta salasanojen murtamisessa (Burr et al., 2013, s. 9). Entropian yksikkö on bitti.

Satunnaisesti valitun salasanan entropia H voidaan laskea kaavan $\mathrm{H} = \log_2(b^l)$ avulla (Burr et al., 2013, s. 104). Kaavassa b merkitsee käytetyn aakkoston kokoa eli uniikkien merkkien lukumäärää ja l valitun salasanan pituutta merkkeinä. Jos salasana muodostetaan esimerkiksi valitsemalla satunnaisesti seitsemän merkkiä suomen kielen 29 aakkosen joukosta, voidaan salasanan entropiaksi laskea $\log_2(29^7) \approx 34$ eli 34 bittiä. Kaavassa b^l siis kuvaa mahdollisten kombinaatioiden määrää. Koska entropian määrää mitataan kaksikantaisen logaritmin avulla laskettuina bitteinä, jokainen lisäbitti kaksinkertaistaa salasanan murtamiseksi tarvittavien arvausten määrän. Mitä korkeampi määrä bittejä entropiaa, sitä vahvempi salasana.

Koska entropia kuvaa tiivisteen murtamiseksi tarvittavien arvausten määrää, voidaan sitä käyttää arvioimaan murtamiseen kuluvaa aikaa. Salasanalle laskettu entropia voidaan jakaa arvausten määrällä per sekunti. Se montaako tiivistettä voidaan testata joka sekunti riippuu monesta tekijästä kuten tiivisteiden laskemiseen käytetystä tiivistefunktiosta, mahdollisista hidasteista kuten avaimen venytyksestä sekä hyökkääjän käytössä olevasta laitteistosta.

Entropiaa salasanan vahvuuden mittarina on myös kritisoitu. Koska salasanat eivät noudata mitään tilastollista jakaumaa, entropiaa ei voida pitää sopivana käyttötarkoitukseen (Ma et al., 2010). Entropian laskeminen ei ole myöskään aivan yhtä yksinkertaista käyttäjien itse valitsemille salasanoille, sillä ne eivät ole satunnaisia ja täten salasanan eri merkkien esiintymistiheys ei jakaudu tasaisesti (Burr et al., 2013, s. 105).

Tarkastellaan entropian ongelmallisuutta vielä esimerkin kautta. Valitaan suomen kielen 29 aakkosen joukosta kaksi salasanaa: "aarrearkku" ja "xdöfujpvif". Salasanoista ensimmäinen on käyttäjän valitsema, suomen kielen sana ja toinen taas on satunnaisesti generoitu merkkijono samassa aakkostossa. On selvää, että ensimmäinen salasana on murrettavissa erittäin nopeasti sanakirjahyökkäyksellä, kun taas toinen on näennäisesti satunnainen ja vaatii todennäköisesti väsytyshyökkäyksen murtuakseen. Kuitenkin entropian laskukaavan perusteella molemmilla on $\log_2(29^{10}) \approx 46$ bittiä entropiaa.

4.4.2 Salasanan valitseminen

Käyttäjän vastuulla on mahdollisimman vahvan salasanan valitseminen. Tarkastellaan seuraavaksi turvallisen salasanan piirteitä pitäen mielessä yleisimmät salasanojen murtamiseen käytetyt hyökkäysmenetelmät.

On selvää, että sanakirjahyökkäystä vastaan suojautuakseen käyttäjän valitseman salasanan ei tule löytyä sanakirjasta. Valittu salasana ei siis saisi olla mikään tavallinen jonkin luonnollisen kielen sana, vaikka se olisikin kirjoitettu poikkeavalla tavalla (esim. "aarrearkku" → "aArrEarKku"). Käyttäjän ei myöskään kannata yrittää olla nokkela ja korvata kirjaimia erikoismerkeillä tai numeroilla, sillä myös yleiset muunnokset kuten A-kirjaimen muuttaminen sitä muistuttavaksi @-merkiksi tai E-kirjaimen korvaaminen numerolla 3 ovat varmasti hyökkääjien tiedossa (esim. "aarrearkku" → "@@rr3@rkku"). Tällaiset muutokset ovat helposti murrettavissa sääntöjen avulla (ks. luku 3.3).

Valitun salasanan on oltava uniikki jokaista palvelua kohden. Käyttäjän ei siis tule käyttää samaa salasanaa useammin kuin kerran. Mikäli käyttäjä on rekisteröitynyt palveluun A ja B samalla salasanalla ja palvelussa A tapahtuu tietomurto, on käyttäjän tili vaarassa myös palvelussa B. Muuten ominaisuuksiltaan turvallinen, mutta kertaalleen vuotanut ja murrettu salasana saattaa myös päätyä osaksi sateenkaaritaulukkoa.

Luvussa 4.4.1 esiteltiin entropian käsite salasanan vahvuuden mittarina. Entropiaa eli salasanan vahvuutta voidaan kasvattaa parhaiten valitsemalla mahdollisimman pitkä salasana. Kuten edellisessä luvussa todettiin, jokainen lisämerkki kaksinkertaistaa murtamiseen vaadittavien arvausten määrän. Toinen entropian laskukaavassa esiintyvä muuttuja b, tarkoittaa käytetyn aakkoston kokoa. Salasanan vahvuutta voi siis kasvattaa myös lisäämällä siihen erikoismerkkejä.

Käyttäjän tulisi valita salasana, joka on mahdollisimman pitkä, satunnaisesti muodostettu merkkijono, joka sisältää merkkejä mahdollisimman suuresta aakkostosta (esim. Unicode). Tällainen salasana tarjoaa suojan tunnettuja hyökkäysmenetelmiä vastaan: se on riittävän pitkä väsytyshyökkäystä vastaan eikä sitä löydy sanakirjasta tai sateenkaaritaulukosta. Käyttäjän ei myöskään tarvitse muistaa salasanojaan, jos hän käyttää hyödyksi salasanan hallintaohjelmistoa.

Salasanan sijaan voidaan käyttää myös salalausetta (engl. *passphrase*). Eräs tapa luoda salalause on noppaware (engl. *diceware*), jossa 7776 sanan joukosta arvotaan noppaa heittämällä haluttu määrä sanoja, jotka yhdessä muodostavat salalauseen (Reinhold, 2022). Noppawaren soveltuvuutta on kuitenkin myös kritisoitu. Antonov ja Georgieva (2020) esit-

tävät, että käytettävän sanalistan koon tulisi olla suurempi, jotta noppawaren avulla luotu salalause olisi riittävän turvallinen.

4.5 Salasanakäytännöt

Usein salasana ei ole kuitenkaan täysin vapaasti käyttäjän päätettävissä, vaan palvelut asettavat salasanoille erilaisia vaatimuksia. Näistä vaatimuksista käytetään myös nimeä salasanakäytäntö (engl. *password policy*).

Salasanakäytäntöjen avulla palvelut pyrkivät ohjaamaan käyttäjiään valitsemaan vahvempia salasanoja. Esimerkiksi vaatimus numeroiden ja erikoismerkkien käyttämisestä kasvattaa suoraan aakkoston kokoa, jolloin salasanan entropia kasvaa. Pelkkien kirjainten käyttämisen estäminen ehkäisee myös sanakirjahyökkäyksen tehokkuutta, kun salasana ei voi löytyä sellaisenaan sanakirjasta, kun siihen on sisällytetty tietty määrä erikoismerkkejä.

OWASP (2023b) määrittelee joukon salasanalle asetettavia vaatimuksia, jotka sovelluskehittäjien olisi hyvä ottaa huomioon salasanakäytännöissään. OWASP suosittelee salasanan vähimmäispituudeksi kahdeksaa merkkiä, joskin myös 12 merkkiä voidaan pitää sopivana salasanan minimipituutena (Bosnjak et al., 2018). OWASP:n mukaan salasanan maksimipituutta ei tulisi asettaa liian alhaiseksi, mutta yli 64-merkkisten salasanojen mainitaan saattavan aiheuttaa ongelmia tiettyjen tiivistefunktioiden kohdalla (OWASP, 2023b). Salasanan maksimipituuden rajoittaminen saattaa estää salalauseen käyttämisen.

Lisäksi OWASP suosittelee, että kaikki mahdolliset Unicode-merkistön symbolit olisivat sallittuja. Kehittäjän ei siis tulisi rajoittaa sallittua merkistöä pelkästään kirjaimiin, numeroihin ja pieneen määrään erikoismerkkejä. Sallimalla valtava määrä merkkejä pelkkä väsytyshyökkäys muuttuu hyvin hitaaksi, kun hyökkääjän on käytävä läpi entistä suurempi määrä erilaisia yhdistelmiä.

Salasanan koostumuksen määrittelyn lisäksi salasanakäytäntöihin voi kuulua salasanan käyttöajan rajaaminen, jonka jälkeen käyttäjä ei enää pysty kirjautumaan tililleen ennen uuden salasanan asettamista. Käyttäjän syöttämää salasanaa voidaan ennen hyväksymistä myös verrata olemassa oleviin vuodettuihin salasanalistoihin, jolloin vuotaneita salasanoja ei ole mahdollista käyttää palvelussa.

4.6 Monivaiheinen tunnistautuminen

Mikäli hyökkääjä on kuitenkin onnistunut selvittämään palvelun käyttäjän salasanan, voidaan luvaton sisäänkirjautuminen vielä estää käyttämällä monivaiheista tunnistautumista (engl. *multi-factor authentication* eli MFA). Monivaiheinen tunnistautuminen on yleensä toteutettu kaksivaiheisena tunnistautumisena (engl. *two-factor authentication* eli 2FA), jossa oikean salasanan syöttämisen jälkeen käyttäjää pyydetään vielä vahvistamaan identiteettinsä ennalta asetetun kanavan kautta. Siinä missä salasana on jotakin mitä käyttäjä tietää, monivaiheinen tunnistautuminen on yleensä jotakin mitä käyttäjällä on.

Tyypillisesti monivaiheinen tunnistautuminen toteutetaan lähettämällä käyttäjän sähköpostiosoitteeseen tai matkapuhelinnumeroon kertakäyttöinen kirjautumiskoodi tai käyttämällä käyttäjän omistamalle laitteelle asennettua erillistä sovellusta. Erillinen MFA-sovellus on turvallisin vaihtoehto, sillä esimerkiksi sähköpostiosoite ei ole sidottu vain yhteen laitteeseen kerrallaan. Päätelaitteen tulee olla ainoastaan käyttäjän hallussa, jotta hänet voidaan tunnistaa luotettavasti. On sovelluskehittäjän vastuulla toteuttaa tuki monivaiheiselle tunnistautumiselle, mutta käyttäjän on osattava ottaa se käyttöön onnistuneesti sekä ymmärrettävä sen toimintamekanismi. Tämä lisää kirjautumisen kognitiivista kuormittavuutta.

Monivaiheisen tunnistautumisen voidaan ajatella olevan viimeinen puolustuslinja, johon hyökkääjä voidaan vielä pysäyttää salasanan päädyttyä vääriin käsiin. MFA:ta hyödyntävälle käyttäjälle saapunut ilmoitus epäilyttävästä kirjautumisyrityksestä saattaa myös paljastaa salasanan päätyneen vääriin käsiin, jolloin se on vaihdettava välittömästi.

Sovelluskehittäjä voi halutessaan tehdä monivaiheisen tunnistautumisen käyttämisestä pakollista, kuten esimerkiksi GitHub (2023). Vaikka salasanan valitsee lopulta käyttäjä itse, kehittäjä voi kuitenkin yrittää ohjata ja opastaa käyttäjää.

Eduistaan huolimatta myös kaksivaiheinen tunnistautuminen on altis hyökkäyksille. Salasanan haltuunsa saanut hyökkääjä saattaa pyrkiä manipuloimaan käyttäjää luovuttamaan MFA-kirjautumiskoodinsa esiintymällä palvelun ylläpitäjänä. Jos MFA:n kirjautumiskoodit lähetetään käyttäjälle SMS-viestinä, hyökkääjä saattaa pyrkiä saamaan haltuunsa SIM-kortin, jolla on sama puhelinnumero kuin hyökkäyksen kohteena olevalla käyttäjällä (Hess et al., 2021).

5 Yhteenveto

Tutkielmassa käsiteltiin salasanojen tiivisteiden murtamista sekä siihen käytettyjä menetelmiä. Havaittiin, että salasanojen tiivisteitä voidaan murtaa varsin tehokkaasti. Jos käytössä oleva laskentatehon määrä jatkaa kasvuaan Mooren lain mukaisesti, tulee salasanojen jatkossa olla yhä pidempiä ollakseen turvallisia.

Toinen keskeinen tutkimuskysymys liittyi murtamisen menetelmiltä suojautumiseen. Havaittiin, että hyökkäyksiltä suojautuminen on itse asiassa varsin monimutkaista. Ohjelmistokehittäjän on osattava ottaa huomioon erilaiset hyökkäysmenetelmät ja käyttäjän on ymmärrettävä vahvan salasanan piirteet sekä salasanojensa oikeaoppinen säilyttäminen. Puutteet salasanojen säilytyksessä tai huonosti valittu salasana mahdollistavat tiivisteiden nopeamman murtumisen.

Salasanojen tietoturva on monimutkainen, mutta myös kriittinen kysymys. Koska salasanat ovat edelleen yleisin todentamiseen käytetty menetelmä, on aihepiirin tutkiminen myös jatkossa erityisen tärkeää. Aihe koskettaa käytännössä jokaista Internetin käyttäjää. Etenkin suojautumiseen on kiinnitettävä huomiota, sillä tiivisteiden murtaminen ei ole teknisesti edes kovin haastavaa. Vuotaneita salasanoja ja niiden tiivisteitä on miljoonittain saatavilla ja ohjelmistot kuten Hashcat mahdollistavat hyökkäyksen suorittamisen omalla tietokoneella jopa ilman erityistä ohjelmointitaitoa.

Tutkielman keskeinen havainto on, että salasanojen tiivisteiden murtaminen on mahdollista lähinnä siksi, että ihmiset valitsevat heikkoja salasanoja. Uniikkeja, riittävän pitkiä ja satunnaisesti generoituja salasanoja ei pystytä nykyisillä menetelmillä murtamaan kohtuullisessa ajassa. Jos jokainen käyttäjä valitsisi riittävän vahvan salasanan eikä myöskään joutuisi sosiaalisen manipuloinnin tai haittaohjelmien uhriksi, eivät hyökkääjät saisi ensimmäistäkään salasanaa murrettua. Tämä tosin lienee toiveajattelua ja käytännössä mahdotonta.

Voidaan myös argumentoida, että salasanatietokannan vuotaminen ulkopuolisille ja murtamisen mahdollistaminen on sovelluskehittäjän syytä. Tämän takia myös kehittäjällä on vastuu salasanojen turvallisesta säilyttämisestä, sopivan tiivistefunktion valitsemista ja kryptografisen suolan käytöstä. Sovelluskehittäjän vastuulla on myös sopivien salasanakäytäntöjen määritteleminen sekä tuen lisääminen monivaiheiselle tunnistautumiselle.

Salasanojen ilmeisistä ongelmista johtuen myös vaihtoehtoisia keinoja käyttäjän todentamiseen on syytä kehittää. Erilaisia ratkaisuja kuten biometrinen tunnistus tai kertakäyttöiset salasanat on jo laajalti käytössä. Verkkopankkeihin kirjaudutaan usein erillisillä pankin myöntämillä tunnuksilla sekä erillisellä tunnuslukulistalla tai sovelluksella sen sijaan, että pankin asiakas päättäisi itse käyttäjätunnuksensa ja salasanansa.

Yhteenvetona voidaan todeta, että salasanojen tiivisteihin liittyviä ongelmia voidaan ehkäistä tiedon avulla. Asiaan perehtyneet sovelluskehittäjät luovat turvallisempia sovelluksia ja valveutuneet käyttäjät valitsevat parempia salasanoja.

Tekoälyn käyttö tutkielmassa

Tutkielmaprosessin tukena on käytetty tekoälyä. Tutkielman aiheen löytämiseen ja tarkempaan rajaamiseen käytettiin ChatGPT-palvelua (OpenAI, 2023), joka mahdollistaa keskustelujen käymisen tekoälyn kanssa. Samaa palvelua käytettiin myös LaTeX-syntaksin kertaamisen ja käytön tukena.

Sopivien tutkimusartikkelien löytämisen apuna on käytetty verkossa toimivaa tekoälypohjaista Keenious-sovellusta (Keenious, 2023).

Tekoälyä ei käytetty tekstin tuottamiseen.

Lähteet

Andersson, M. (2023). ”Optimizing the computation of password hashes”. Tutkielma, s. 54. URL: https://helka.helsinki.fi/permalink/358UOH_INST/q5v72t/alma9934509034906253.

Antonov, P. ja Georgieva, N. (2020). ”Security Analysis of Diceware Passphrases”. *Information & security* 47.2, s. 276–282. ISSN: 0861-5160. DOI: 10.11610/isij.4719.

Bonneau, J. (2012). ”The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords”. Teoksessa: *2012 IEEE Symposium on Security and Privacy*. IEEE, s. 538–552. DOI: 10.1109/SP.2012.49.

Bosnjak, L., Sres, J. ja Brumen, B. (2018). ”Brute-force and dictionary attack on hashed real-world passwords”. Teoksessa: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, s. 1161–1166. DOI: 10.23919/MIPRO.2018.8400211.

Burr, W. E., Dodson, D. F., Newton, E. M., Perlner, R. A., Polk, W. T., Gupta, S. ja Nabbus, E. A. (2013). *Electronic Authentication Guideline*. Tekninen raportti NIST SP 800-63-2. National Institute of Standards ja Technology, s. 104. DOI: 10.6028/NIST.SP.800-63-2.

Dang, Q. (2012). *Recommendation for Applications Using Approved Hash Algorithms*. DOI: 10.6028/NIST.SP.800-107r1.

Finlex (2015). *Rikoslaki, 38. luku, 8. pykälä*. URL: https://www.finlex.fi/fi/laki/ajantasa/1889/18890039001#L38 (viitattu 24. 12. 2023).

GitHub (2023). *About mandatory two-factor authentication*. URL: https://docs.github.com/en/authentication/securing-your-account-with-two-factor-authentication-2fa/about-mandatory-two-factor-authentication (viitattu 27. 12. 2023).

Grassi, P. A., Garcia, M. E. ja Fenton, J. L. (2017). *Digital identity guidelines: revision 3*. Tekninen raportti NIST SP 800-63-3. National Institute of Standards ja Technology, s. 49. DOI: 10.6028/NIST.SP.800-63-3.

Hashcat (2023). *Rule-based attack, Implemented compatible functions*. URL: https://hashcat.net/wiki/doku.php?id=rule_based_attack#implemented_compatible_functions (viitattu 24. 12. 2023).

Heen, O. ja Neumann, C. (2017). ”On the Privacy Impacts of Publicly Leaked Password Databases”. Teoksessa: *Detection of Intrusions and Malware, and Vulnerability Assess-*

ment. Toim. M. Polychronakis ja M. Meier. Vol. 10327. Series Title: Lecture Notes in Computer Science. Springer International Publishing, s. 347–365. DOI: 10.1007/978-3-319-60876-1_16.

Hess, E. M., Tolbert, M. M. ja Nascimento, M. C. (2021). *Vulnerabilities of Multi-factor Authentication in Modern Computer Networks*. Tekninen raportti. Worcester Polytechnic Institute, s. 11.

Keenious (2023). *Keenious*. URL: https://keenious.com (viitattu 24. 12. 2023).

Ma, W., Campbell, J., Tran, D. ja Kleeman, D. (2010). "Password Entropy and Password Quality". Teoksessa: *2010 Fourth International Conference on Network and System Security*. Melbourne, Australia: IEEE, s. 583–587. DOI: 10.1109/NSS.2010.18.

Menezes, A. J., Van Oorschot, P. C. ja Vanstone, S. A. (1997). *Handbook of applied cryptography*. CRC Press, s. 33. ISBN: 0-8493-8523-7.

Nordpass (2023). *Top 200 Most Common Passowords*. URL: https://nordpass.com/most-common-passwords-list (viitattu 24. 12. 2023).

Oechslin, P. (2003). "Making a Faster Cryptanalytic Time-Memory Trade-Off". Teoksessa: *Advances in Cryptology - CRYPTO 2003*. Toim. G. Goos, J. Hartmanis, J. van Leeuwen ja D. Boneh. Vol. 2729. Series Title: Lecture Notes in Computer Science. Springer Berlin Heidelberg, s. 617–630. DOI: 10.1007/978-3-540-45146-4_36.

OpenAI (2023). *ChatGPT*. URL: https://chat.openai.com (viitattu 24. 12. 2023).

OWASP (2023a). *10k worst passwords*. URL: https://github.com/OWASP/passfault/blob/master/wordlists/wordlists/10k-worst-passwords.txt (viitattu 24. 12. 2023).

– (2023b). *Authentication Cheat Sheet*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (viitattu 24. 12. 2023).

– (2023c). *How attackers crack password hashes*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#how-attackers-crack-password-hashes (viitattu 24. 12. 2023).

Reinhold, A. G. (2022). *The Diceware Passphrase Home Page*. URL: https://theworld.com/~reinhold/diceware.html (viitattu 27. 12. 2023).

Shen, C., Yu, T., Xu, H., Yang, G. ja Guan, X. (2016). "User practice in password security: An empirical study of real-life passwords in the wild". *Computers & Security* 61, s. 130–141. DOI: 10.1016/j.cose.2016.05.007.

Zimmermann, V. ja Gerber, N. (2020). "The password is dead, long live the password – A laboratory study on user perceptions of authentication schemes". *International Journal of Human-Computer Studies* 133, s. 26–44. DOI: 10.1016/j.ijhcs.2019.08.006.