Monivaiheinen tunnistautuminen (MFA)

Ryhmä 13

Leevi Kauranen, AC7750

Samir Benjenna, AD1437

Eelis Suhonen, AA3910

Juho Eräjärvi, AD1276

Mikke Kuula, AC7806

Koventaminen TTC6050-3007

6.12.2024

Tieto- ja viestintätekniikka

Sisältö

[1 Johdanto 4](#_Toc184378814)

[2 Teoria 4](#_Toc184378815)

[2.1 Google Authenticator 5](#_Toc184378816)

[3 Työn kulku 6](#_Toc184378817)

[3.1 WordPress 6](#_Toc184378818)

[3.2 WWW-palvelin 11](#_Toc184378819)

[4 Pohdinta 18](#_Toc184378820)

[Lähteet 20](#_Toc184378821)

Kuviot

[Kuvio 1. VLE-ympäristö 4](#_Toc184378822)

[Kuvio 2. WP 2FA asennus 6](#_Toc184378823)

[Kuvio 3. Setup Wizard 7](#_Toc184378824)

[Kuvio 4.One-timeCode 7](#_Toc184378825)

[Kuvio 5. Monivaiheisen tunnistautumisen vaatimisen rajaus 8](#_Toc184378826)

[Kuvio 6. Siirtymäajan valitseminen 8](#_Toc184378827)

[Kuvio 7. QR-koodi sovelluksen käyttöönottoa varten 9](#_Toc184378828)

[Kuvio 8. Backup koodien generoinnin valinta 10](#_Toc184378829)

[Kuvio 9. Autentikointi koodi 10](#_Toc184378830)

[Kuvio 10. Komennot Google Authenticatorin asentamiseksi 11](#_Toc184378831)

[Kuvio 11. Aikaperusteiset tokenit 11](#_Toc184378832)

[Kuvio 12. QR-koodi WWW-palvelimella 12](#_Toc184378833)

[Kuvio 13. Hätäkoodit 12](#_Toc184378834)

[Kuvio 14. Turvallisuusasetukset 13](#_Toc184378835)

[Kuvio 15. ssh-keygen -komento 14](#_Toc184378836)

[Kuvio 16. id\_rsa yksityinen avain 15](#_Toc184378837)

[Kuvio 17. Puttygen 16](#_Toc184378838)

[Kuvio 18. Sshd\_config -tiedosto 16](#_Toc184378839)

[Kuvio 19. UsePAM 17](#_Toc184378840)

[Kuvio 20. ChallengeResponseAuthentication 17](#_Toc184378841)

[Kuvio 21. AuthenticationMethods 17](#_Toc184378842)

[Kuvio 22. pam.d/sshd -tiedoston muokkaus 17](#_Toc184378843)

[Kuvio 23. Monivaiheinen tunnistautuminen SSH-yhteyttä avattaessa 18](#_Toc184378844)

# Johdanto

Tämän harjoitustyön tarkoituksena on tutustua monivaiheiseen tunnistautumiseen (Multi-Factor Authentication, MFA) ja konfiguroida se käyttöön WordPressiin sekä WWW-palvelimen SSH-kirjautumiseen. Harjoitustyö toteutetaan VLE-ympäristössä, joka on esitetty kuviossa 1.

Kuva, joka sisältää kohteen teksti, diagrammi, kuvakaappaus, Suunnitelma

Kuvaus luotu automaattisesti

Kuvio 1. VLE-ympäristö

# Teoria

Monivaiheinen tunnistautuminen on turvallisuustoimenpide, jossa käyttäjän henkilöllisyys varmistetaan kahdella tai useammalla todennusmenetelmällä. Tavallisesti tämä tarkoittaa käyttäjätunnuksen ja salasanan lisäksi kolmatta tunnistautumistapaa, kuten puhelimeen lähetettävää koodia tai mobiilisovelluksen kautta tehtävää vahvistusta. (Monivaiheinen tunnistautuminen (MFA) 2024).

MFA:n päätarkoitus on estää käyttäjätunnusten väärinkäyttö ja parantaa tietoturvaa. Vaikka salasana joutuisi vääriin käsiin, hyökkääjä ei pysty kirjautumaan palveluun ilman toista tunnistautumismenetelmää. Tämä tekee tietojenkalastelusta ja tunnusten murtamisesta huomattavasti vaikeampaa. (Mikä on MFA ja miksi se tulisi ottaa käyttöön? 2019).

Monivaiheinen tunnistautuminen on erityisen tärkeä arkaluontoista tietoa sisältävissä palveluissa, kuten sähköposteissa ja yritysten sisäisissä järjestelmissä. Se on tehokas keino suojata käyttäjätilejä ja vähentää niiden alttiutta hyökkäyksille. (Monivaiheinen tunnistautuminen suojaa käyttäjätilejäsi. 2024)

## Google Authenticator

Käytämme labran molemmissa vaiheissa Google Authenticator sovellusta, jonka avulla kirjautumisesta saadaan monivaiheinen. Kun se on konfiguroituna Wordpressiin tai SSH:lle, kirjautumiseen vaaditaan salasanan tai avainten lisäksi koodi, jonka saa sovelluksesta esimerkiksi mobiililaitteeseen. Google Authenticator luo kertakäyttöisiä salasanoja (OTP) käyttämällä kahden algoritmin avulla: **HOTP** (Event-based) ja **TOTP** (Time-based):

1. **Salainen avain:**

* Sekä palvelin että asiakas (Google Authenticator) jakavat yhteisen salaisen avaimen, joka tallenne-taan luotettavasti molemmille osapuolille. Avain toimitetaan usein QR-koodina.

1. **HOTP (Event-based):**

* Käytetään salaisen avaimen lisäksi laskuria, jonka arvo nousee, kun salasana luodaan ja sitä käyte-tään. Asiakas ja palvelin pysyvät synkronoituna laskurin käydessä.

1. **TOTP (Time-based):**

* Käyttää samaa salaista avainta kuin HOTP, mutta laskurin sijaan perustuu nykyiseen aikaan.Aika synkronoidaan molemmille osapuolille esimerkiksi Network Time Protocolin (NTP) avulla.

1. **Koodin muodostuminen:**

* Google Authenticator laskee kertakäyttöisen salasanan (yleensä 6-numeroinen) yhdistämällä salai-sen avaimen ja ajan/laskurin algoritmiin. Asiakas syöttää koodin, ja palvelin tarkistaa sen laskemalla saman koodin.

1. **Turvallisuus:**

* Salainen avain ja laskuri/aika varmistavat, että molemmat osapuolet voivat tuottaa saman salasanan ilman suoraa yhteyttä, mikä tekee järjestelmästä turvallisen.

(How does Google Authenticator work? 2013.)

# Työn kulku

## WordPress

Aloitimme monivaiheisen tunnistautumisen lisäämisen WordPressiin kirjautumalla WordPressin hallintapaneeliin osoitteessa <http://www.ttc60z.vle.fi/wp-admin/>. Plugins välilehdeltä etsimme ”WP 2fa” ja painoimme Install Now. (Kuvio 2).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Verkkosivusto

Kuvaus luotu automaattisesti

Kuvio 2. WP 2FA asennus

Ohjeen mukaan seuraavaksi olisi pitänyt tulla virhe, että kansiota ei voida luoda. Olimme aiemmassa harjoitustyössä jo tehneet tämän vaiheen pluginien päivityksen ohessa.

Pluginin asennuksen jälkeen klikkasimme activate ja siirryimme automaattisesti Setup Wizardiin. (Kuvio 3).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti, logo

Kuvaus luotu automaattisesti

Kuvio 3. Setup Wizard

Emme ottaneet käyttöön sähköpostitodennusta, vaan ainoastaan kertakäyttökoodit applikaation avulla. (Kuvio 4)

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti, dokumentti

Kuvaus luotu automaattisesti

Kuvio 4.One-timeCode

Otimme käyttöön vaihtoehdon, että jokaisen käyttäjän on käytettävä monivaiheista tunnistautumista. (Kuvio 5).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Kuvio 5. Monivaiheisen tunnistautumisen vaatimisen rajaus

Seuraavaksi valitsimme vaihtoehdon, että käyttäjien on otettava monivaiheinen tunnistautuminen käyttöön heti ilman siirtymäaikaa. (Kuvio 6).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Kuvio 6. Siirtymäajan valitseminen

Klikkasimme All done ja valitsimme seuraavaksi Configure 2FA now.

Valitsimme tunnistautumiseen käytettäväksi applikaatioksi Google Autheticator -sovelluksen.

Skannasimme QR koodin, joka näkyy kuviossa 7 ja saimme 6 numeroisen koodin puhelimessa olevaan authenticator sovellukseen. Teimme ryhmällemme myös oman sähköposti osoitteen tunnistautumista varten.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Verkkosivusto

Kuvaus luotu automaattisesti

Kuvio 7. QR-koodi sovelluksen käyttöönottoa varten

Syötettyämme koodin puhelimen sovelluksesta, saimme valita, generoimmeko backup koodeja. Valitsimme, että generoimme ne myöhemmin.

Kuva, joka sisältää kohteen teksti, ohjelmisto, Multimediaohjelmisto, multimedia

Kuvaus luotu automaattisesti

Kuvio 8. Backup koodien generoinnin valinta

Seuraavaksi testasimme kirjautumista käyttäen kaksivaiheista tunnistautumista. Kirjautuessamme WordPressin hallintasivulle, meiltä kysyttiin tunnistautumiskoodia. (Kuvio 9).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti, logo

Kuvaus luotu automaattisesti

Kuvio 9. Autentikointi koodi

Syötimme Google Authenticatorissa näkyvän 6 numeroisen koodin pääsimme hallintapaneeliin.

## WWW-palvelin

Seuraavana tehtävänä oli konfiguroida monivaiheinen tunnistautuminen toimimaan WWW-palvelimelle SSH-kirjautumista varten. Asensimme Google Authenticatorin palvelimelle kuvion 10 mukaisilla komennoilla.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Kuvio 10. Komennot Google Authenticatorin asentamiseksi

Kuvion 10 alimmainen komento luo salaisen avaimen tunnistautumista varten .ssh kansioon nimellä google\_authenticator. Seuraavaksi meiltä kysyttiin, tahdommeko tunnistustokenien olevan aikaperusteisia. Vastaamme tähän kyllä. (Kuvio 11)



Kuvio 11. Aikaperusteiset tokenit

Vastauksen jälkeen komentoriville tuli QR-koodi, jonka luimme puhelimen Google Authenticator sovelluksella. (Kuvio 12).

Kuva, joka sisältää kohteen teksti, kuvio, kuvakaappaus, neliö

Kuvaus luotu automaattisesti

Kuvio 12. QR-koodi WWW-palvelimella

Syötimme sovelluksessa näkyvän koodin komentoriville. Saimme hätäkoodeja, jotka tallensimme itsellemme turvalliseen paikkaan. (Kuvio 13)

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Kuvio 13. Hätäkoodit

Seuraavaksi vastasimme kyllä kaikkiin kuvion 14 kysymyksiin. Kysymykset liittyivät turvallisuus asetuksiin. Esimerkiksi viimeisenä oli kysymys, että konfiguroidaanko rate-limiting joka suojaa meitä brute-force hyökkäyksiltä.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Kuvio 14. Turvallisuusasetukset

Seuraavaksi asetimme SSH daemonin käyttämään google authentikaattoria.

Ryhmällämme oli aiemmin käytössä SSH-avaimet käyttäjien SSH-kirjautumisille, mutta jouduimme edellisen harjoitustyön loppuvaiheilla resetoimaan WWW-palvelimen. Kofigroimme ensin SSH-avaimen käyttöön. Tässä esimerkkinä luodaan SSH-avain root-käyttäjälle.

Generoimme SSH-avaimen komennolla ssh-keygen. (Kuvio 15).

Kuva, joka sisältää kohteen teksti, kuvakaappaus

Kuvaus luotu automaattisesti

Kuvio 15. ssh-keygen -komento

Kopiomme julkisen avaimen authorized keys kansioon komennolla dp id\_rsa.pub authorized\_keys. Kopioimme myös avain id\_rsa WS01:lle talteen Putty- kirjautumista varten. (Kuvio 16).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti, kuvio

Kuvaus luotu automaattisesti

Kuvio 16. id\_rsa yksityinen avain

Loimme avaimen tekstitiedostosta Puttygen ohjelmalla ja tallensimme sen painamalla save private key. (Kuvio 17).

Kuva, joka sisältää kohteen teksti, elektroniikka, kuvakaappaus, ohjelmisto

Kuvaus luotu automaattisesti

Kuvio 17. Puttygen

Muokkasimme /etc/ssh/sshd\_config tiedostoa siten, että kirjautuessa vaaditaan avain. (Kuvio 18).

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti, muotoilu

Kuvaus luotu automaattisesti

Kuvio 18. Sshd\_config -tiedosto

Käynnistimme sshd:n uudelleen komennolla systemctl restart sshd, jotta asetukset tulevat voimaan.

Ottaaksemme käyttöön monivaiheisen todennuksen muokkasimme sshd\_config tiedostoon UsePam yes (Kuvio 19) ja ChallengeResponseAuthetication yes (Kuvio 20). Lisäsimme myös kuvion 21 mukaisen rivin.



Kuvio 19. UsePAM

Kuva, joka sisältää kohteen teksti, Fontti, kuvakaappaus

Kuvaus luotu automaattisesti

Kuvio 20. ChallengeResponseAuthentication

Kuva, joka sisältää kohteen kuvakaappaus, Fontti, viiva

Kuvaus luotu automaattisesti

Kuvio 21. AuthenticationMethods

Muokkasimme /etc/pam.d/sshd tiedostoon kuviossa 22 mukaiset rivit.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Verkkosivusto

Kuvaus luotu automaattisesti

Kuvio 22. pam.d/sshd -tiedoston muokkaus

Käynnistimme sshd:n uudelleen komennolla systemctl restart sshd, jotta asetukset tulevat voimaan.

Kun otimme SSH-yhteyden uudelleen WWW-palvelimelle puttyllä, meiltä kysyttiin 2 vaiheista tunnistautumista. Tähän syötettiin puhelimesta Google Authenticatorin koodi. (Kuvio 23).

Kuva, joka sisältää kohteen teksti, elektroniikka, kuvakaappaus, ohjelmisto

Kuvaus luotu automaattisesti

Kuvio 23. Monivaiheinen tunnistautuminen SSH-yhteyttä avattaessa

# Pohdinta

Harjoitustyössä pääsimme tutustumaan kaksivaiheisen tunnistautumisen konfigurointiin ja käyttöön. Tämä oli helpompi toteuttaa kuin kuvittelimme ja homma olikin ohi alle 30 minuutissa. Iloksemme huomasimme myös palvelimelle konfiguroinnissa, että samalla käyttöön otetaan rate-limiting, joka rajoittaa palvelimelle tulevien pyyntöjen määrää ja näin torjuu DDoS-hyökkäyksiltä.

Ohjeet WordPressin monivaiheisen tunnistautumisen konfigurointiin olivat selkeät ja asennus oli lisäosan avulla helppoa. Myös SSH-yhteydelle konfigurointi oli suoraviivaista ja se ei vaatinut paljoakaan komentoja, lähinnä vain tiedostojen muokkaamista, joka helpotti siinä, että tajusi hyvin mitä missäkin vaiheessa tapahtuu.

Toteutuksen helppous korostaa sitä, kuinka kaksivaiheisen tunnistautumisen käyttöä tulisi lisätä yritysympäristöissä ja miksei omissakin ympäristöissä. Moni yritys laiminlyö tietoturvallisuuden toteutusta, ja näen että kaksivaiheinen tunnistautuminen järjestelmiin olisi melkein vähimmäisvaatimus.

Lähteet

How does Google Authenticator work? Stack Exchange vastaus. 2013. Viitattu 4.12.2024. <https://security.stackexchange.com/questions/35157/how-does-google-authenticator-work>.

Mikä on MFA ja miksi se tulisi ottaa käyttöön? Magiccloud.fi artikkeli. 31.12.2019. Viitattu 4.12.2024. <https://magiccloud.fi/mika-on-mfa-ja-miksi-se-tulisi-ottaa-kayttoon/>.

Monivaiheinen tunnistautuminen (MFA). Helsingin Yliopiston Opiskelijan Digitaidot opintojakso. 2024. Viitattu 4.12.2024 <https://blogs.helsinki.fi/opiskelijan-digitaidot/4-tietoturva/>.

Monivaiheinen tunnistautuminen suojaa käyttäjätilejäsi. Kyberturvallisuuskeskuksen artikkeli. 7.5.2024. Viitattu 4.12.2024. <https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/ohjeet-ja-oppaat/monivaiheinen-tunnistautuminen-suojaa-kayttajatilejasi>.