jamk.fi

Organisaatiokohtaisen autentikaatiojärjestelmän ja kaksivaiheisen todennuksen toteuttaminen RGCE-ympäristöön

Henrik Saari

Opinnäytetyö
Joulukuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), Tietoverkkotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Saari, Henrik	Opinnäytetyö, AMK	Joulukuu 2016
	Sivumäärä 94	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Organisaatiokohtaisen autentikaatiojärjestelmän ja kaksivaiheisen todennuksen toteuttaminen RGCE-ympäristöön

Tutkinto-ohjelma

Tietotekniikan (Tietoverkkotekniikan) koulutusohjelma

Työn ohjaaja(t)

Häkkinen, Antti

Rantonen, Mika

Toimeksiantaja(t)

JYVSECTEC

Vatanen, Marko

Tiivistelmä

Opinnäytetyö toteutettiin Jyväskylän ammattikorkeakoulun tiloissa toimivalle kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskukselle, JYVSECTEC:lle. JYVSECTEC tarjoaa monipuolista harjoitus- ja koulutustoimintaa RGCE-kehitysympäristössä.

Tavoitteena työlle oli tutkia ja kehittää RGCE-kehitysympäristössä sijaitseviin organisaatioympäristöihin organisaatiokohtainen autentikaatiojärjestelmä, jonka avulla pystyttäisiin toteuttamaan kertakirjautuminen ympäristön palveluihin. Lisäksi työssä oli tarkoitus tutkia ja toteuttaa kaksivaiheinen todennus Palo Alto palomuurien GlobalProtect-palveluun.

Työssä tutkittiin yleisiä tapoja toteuttaa keskitetty käyttäjän tunnistus, kertakirjautuminen ja kaksivaiheinen todennus. Taustatietojen pohjalta valittiin käytetyt palvelut ja tekniikat.

Toteutus suoritettiin testiympäristössä, joka mallinsi palveluiltaan RGCE-ympäristöä. Kertakirjautumiskomponentti toteutettiin käyttäen avoimeen lähdekoodiin pohjautuvaa LemonLDAP WebSSO –palvelua, joka asennettiin CentOS 7 –palvelimelle. Kertakirjautuminen toteutettiin pääosin käyttäen HTTP-todennusta ja Kerberos-protokollaa. Kaksivaiheinen todennus toteutettiin käyttäen sekä avoimen lähdekoodin tuotetta että maksullisen palvelun testiversiota. Avoimen lähdekoodin palveluna toimi FreeIPA ja maksullisen palvelun tarjosi Duo Security.

Lopputuloksena saatiin valmis komponentti käytettäväksi RGCE-ympäristöön ja kertakirjautuminen saatiin toteutettua valittuihin palveluihin. Kaksivaiheinen todennus Palo Alton GlobalProtect-palveluun saatiin toteutettua ja todennettua.

Avainsanat (asiasanat)

Kertakirjautuminen, kaksivaiheinen todennus, vahva todennus

Muut tiedot

-

jamk.fi

Description

Author(s)	Type of publication	Date
Saari, Henrik	Bachelor's thesis	December 2016
		Language of publication: Finnish
	Number of pages 94	Permission for web publication: x

Title of publication

Implementing organization-specific authentication system and two-factor authentication to RGCE-environment

Degree programme

Information Technology

Supervisor(s)

Häkkinen, Antti

Rantonen, Mika

Assigned by

JYVSECTEC

Vatanen, Marko

Abstract

The bachelor's thesis was assigned by JYVSECTEC, which operates on the premises of Jyväskylä University of Applied Sciences. JYVSECTEC is a cyber security center focusing on research, development and training and offers diverse possibilities for hands on practice in its RGCE –development environment.

The purpose of the thesis was to research and implement a centralized authentication system and single sign-on for the services offered in the RGCE environment. In addition, two-factor authentication was to be researched and implemented to Palo Alto firewall's GlobalProtect-service.

Common techniques and services for centralized authentication, single sign-on and two-factor authentication were researched and based on this, the used services and techniques were selected.

The authentication system was implemented to a test environment, which modelled the services used in the actual RGCE organization environments. The authentication component selected was an open source based LemonLDAP WebSSO –service which was installed on a CentOS 7 server. Single sign-on was accomplished using mainly HTTP-authentication and Kerberos. Two-factor authentication was implemented using both open source and commercial services. The open source service used was FreeIPA and the commercial service used was provided by Duo Security.

As a result, a working authentication component was produced and single sign-on was accomplished to all chosen services. Two-factor authentication for Palo Alto GlobalProtect was successfully implemented and verified.

Keywords/tags (subjects)

Single sign-on, two-factor authentication, strong authentication

Miscellaneous

-

Sisältö

Kuviot

Taulukot

Lyhenteet

AD	Active Directory
ADFS	Active Directory Federation Services
AS	Authentication Server
CAS	Central Authentication Service
HOTP	HMAC-based One-Time Password
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
IdP	Identity Provider
KDC	Key Distribution Center
LDAP	Lightweight Directory Access Protocol
NTLM	NT Lan Manager
NTP	Network Time Protocol
OaaS	Organization as a Service
OP	OpenID Provider
OTP	One Time Password
RGCE	Realistic Global Cyber Environment
RP	Relaying Party
SAML	Security Assertion Markup Language
SP	Service Provider
SPN	Service Principal Name
SSL	Secure Sockets Layer
SSO	Single sign-on
TGS	Ticket Granting Server
TGT	Ticket Granting Ticket
TLS	Transport Layer Security
TOTP	Time-based One-Time Password
VPN	Virtual Private Network
XML	Extensible Markup Language

1 Lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulun JYVSECTEC-hanke. JYVSECTEC aloitti toimintansa projektimuotoisena syyskuussa 2011 tavoitteenaan luoda Keski-Suomeen kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus sekä kehittää turvallisuusalan kansallista ja kansainvälistä yritysten ja toimijoiden yhteistyöverkostoa. Tätä kaikkea mahdollistamaan luotiin RGCE-kehitysympäristö (Realistic Global Cyber Environment), jonka myötä JYVSECTEC on voinut tarjota harjoitus-, tutkimus- ja kehitystyöhön ajankohtaisen infrastruktuurin. Hankkeen onnistumisesta kertoo myös keväällä 2015 saatu kahden miljoonan euron jatkorahoitus toiminnan ja ympäristöjen laajentamiseen ja kyberturvallisuuden tilannekuvan tutkimiseen ja kehittämiseen. (JYVSECTEC – Tietoa meistä 2016.)

RGCE-kybertoimintaympäristö on JYVSECTEC:n asiantuntijoiden kehittämä kyberturvallisuuden tutkimus-, kehitys- ja koulutusympäristö. Ympäristö on toteutettu nykyaikaista pilvipalveluarkkitehtuuria käyttäen ja virtualisoinnin lisäksi ympäristöön voidaan liittää myös fyysisiä elementtejä, kuten verkon aktiivilaitteita eli palomuurituotteita, kytkimiä ja reitittimiä. RGCE tarjoaa täysin eristetyn ja kontrolloidun ympäristön, joka on suunniteltu ja rakennettu vastaamaan Internetin todellisia rakenteita ja toiminnollisuuksia. Nämä pitävät sisällään oikeaa Internetiä mallintavat julkiset IP-osoitteistukset maantieteellisesti sidottuina ja lukuisia Internetin ydinpalveluja, kuten nimi- ja aikapalvelut sekä sertifikaatti-infrastruktuuri. RGCE:n infrastruktuuri tarjoaa mahdollisuuden luoda monipuolisia organisaatioympäristöjä mallintamaan todellisen maailman yrityksiä palveluineen ja verkkoineen. (JYVSECTEC – RGCE 2016.)

1.2 Toimeksianto ja tavoitteet

Työn tavoitteena oli suunnitella ja toteuttaa SSO-autentikaatiojärjestelmä (Single sign-on) RGCE-kehitysympäristössä käytettyihin organisaatiomalleihin. Toimeksiantajan toiveena oli, että järjestelmä voidaan toteuttaa yhtenä komponenttina haluttuun organisaatioympäristöön. Lisäksi työssä oli tarkoitus tutkia ja toteuttaa Palo Alto – palomuurien VPN-palveluun (Virtual Private Network) kaksivaiheinen todennus.

RGCE-ympäristössä on käytössä Palo Alton virtuaalipalomuureja, joilla kyseinen palvelu voidaan toteuttaa haluttuun ympäristöön. Toteutustapa eli menetelmät ja palvelut jätettiin työn tekijän harkinnan varaan, mutta jos mahdollista työ tulisi toteuttaa käyttäen avoimeen lähdekoodiin pohjautuvia ilmaisia ratkaisuja. Toteutus tehtiin omassa testiympäristössä, joka karkeasti mallinsi RGCE-organisaatioympäristöissä tarjottavia palveluita.

1.3 Organization as a Service

RGCE-organisaatioympäristöihin tarjotaan valmista pohjamallia, josta voidaan tarpeen ja tilanteen mukaisesti valita halutut komponentit toteutettavaan organisaatioon. OaaS (Organization as a Service) mahdollistaa kustannustehokkaan tavan luoda eri tarkoitusperiä palvelevia ympäristöjä. Valmis jaottelu verkkoihin ja komponentteihin mahdollistaa ympäristöjen tuottamisen hyvin pitkälti automatisoidusti. Yleisluontoinen organisaationmalli esitetään liitteessä 1.

Luotava ympäristö voi siis vaihdella palveluiltaan, mutta itse toteutettavat palvelut ovat samoista mallipohjista ja täten toimivat melkein identtisesti ympäristöstä riippumatta. Lisäksi palveluiden nimet ja osoitteistukset on pyritty vakioimaan, joten opinnäytetyön lopputuloksen tulisi sopia ilman suurempia muutoksia moneen eri ympäristöön.

2 Single sign-on

2.1 Yleistä

Yrityksen sisäisten ja ulkoisten palveluiden siirtyessä yhä enemmän tietoverkkoihin, tavallinen yritysverkon käyttäjä, eli työntekijä kohtaa yleensä yhden työpäivän aikana lukuisia eri järjestelmiä ja sovelluksia, joita hänen täytyy käyttää suoriutuakseen työstään. Nämä palvelut sijaitsevat yleensä omien käyttäjätunnustensa takana ja työntekijällä voi nopeasti olla muistettavanaan useita eri käyttäjätunnus-salasana –yhdistelmiä. Käyttäjäkokemus jää näin ollen todella heikoksi ja työn tehokkuus kärsii. (Tipton & Krause 2007.)

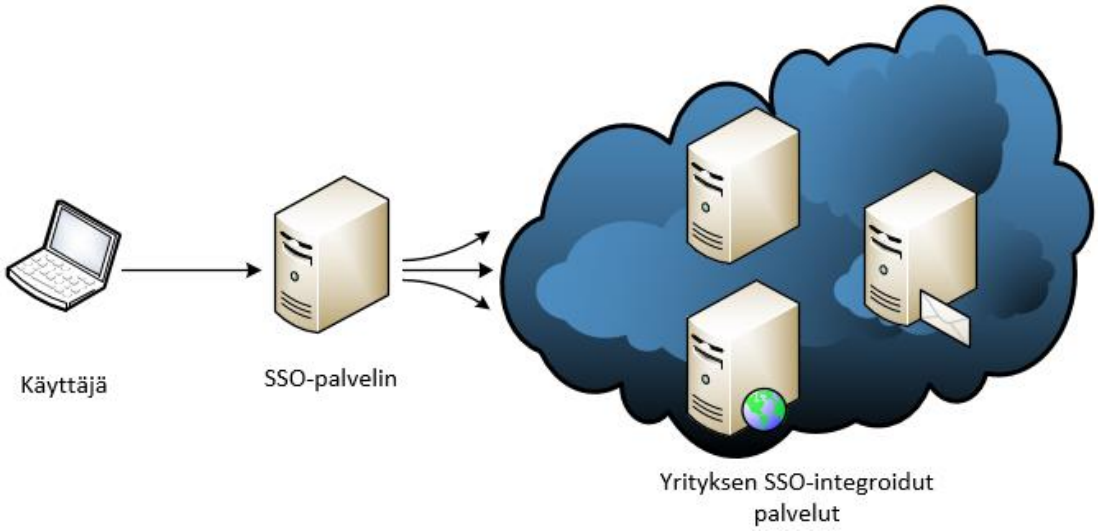
Yrityksen sisäisten palveluiden sitominen yrityksen omaan käyttäjähakemistoon tuo monesti helpotusta ongelmaan. Kuitenkin varsinaisia järjestelmiin ja palveluihin kirjautumisia tulee silti suuri määrä jo yhden työpäivän aikana ja edelleen työn tehokkuus ei ole maksimissaan. Suorasukaisempi ratkaisu ongelmaan on SSO eli kertakirjautuminen. (Tipton & Krause 2007.)

Nykypäivän tietoverkkojärjestelmät ovat normaalisti hyvin hajautettuja, niin fyysisesti kuin loogisestikin. Tämä tarkoittaa, että yrityksen palvelut tarjotaan mahdollisesti lukuisista eri sijainneista ja sovellusten taustalla pyörivä rauta ja ohjelmisto voi vaihdella runsaastikin. Tämä johtaa väistämättä siihen, että SSO:n toteuttaminen on yleensä todella haastavaa ja syö paljon resursseja. (Tipton & Krause 2007.)

SSO-ympäristössä käyttäjän täytyy todentaa itsensä, eli kirjautua, vain kerran ja tämän jälkeen SSO-järjestelmä on vastuussa käyttäjän kirjaamisesta järjestelmän piirissä oleviin palveluihin. Järjestelmällä saavutetaan lukuisia etuja, kuten:

1. Vähennetään käyttäjätunnus ja salasana –yhdistelmiä: Käyttäjän ei tarvitse muistaa ja hallita suurta määrää eri kirjautumistietoja
2. Salasanojen hallinnan helpottaminen: Käyttäjä ei yhtä todennäköisesti käytä heikkoja salasanoja ja/tai kirjoita niitä muistiin muistilapuille/muistiinpanoihinsa
3. Käyttäjäkokemuksen parantuminen: Käyttäjän tarvitsee käyttää huomattavasti vähemmän aikaa eri palveluihin kirjautumiseen
4. IT-ylläpidon työn helpottuminen: Vähemmän käyttäjien yhteenottoja salasanojen resetointien osalta
5. Keskitetty komponentti käyttäjien hallintaan: Esimerkiksi pääsynhallinta ja käyttäjän salasanojen hallinta onnistuu yhdellä komponentilla

Kuviossa 1 on esitelty yrityksen SSO-järjestelmän pääperiaate. Käyttäjä kirjautuu sisään SSO-järjestelmään, joka on vastuussa käyttäjän kirjaamisesta yritysverkon palveluihin. Tämä automaattinen kirjautuminen voidaan toteuttaa esimerkiksi tiketin tai evästeen avulla, jonka SSO-palvelin jakaa käyttäjälle todennuksen yhteydessä. (Jamsa 2013.)



Kuvio 1. Yrityksen SSO-järjestelmän periaate (Jamsa 2013.)

2.2 Ratkaisumallit

SSO:n implementointiin ei ole yhtä oikeaa toteutustapaa vaan ratkaisumalli voi käyttää tiettyä SSO-arkkitehtuuria, tai yhdistelmää eri arkkitehtuureista. Arkkitehtuurit voidaan karkeasti jakaa neljään eri kategoriaan: Broker-, Agent-, Token- ja Proxy-based. (Hursti 1997.)

Broker-mallissa verkossa on keskitetty todennuspalvelin. Tämä palvelin jakaa loppukäyttäjälle esimerkiksi tiketin tai evästeen, jonka avulla loppukäyttäjä voi pyytää pääsyä muihin järjestelmiin ilman erillistä kirjautumista. Kerberos on yksi esimerkki Broker-mallin SSO:sta. (Hursti 1997.)

Agentteihin pohjautuvassa mallissa jokaisen SSO-järjestelmään integroidun palvelun front endistä löytyy todennuksen tekevä agentti. Tämä agentti toimii tulkkina sovelluksen oman todennusjärjestelmän ja SSO-järjestelmän välillä. Esimerkkinä tästä on

agentti (Authentication Agent), joka tunnistaa käyttäjän esimerkiksi X.509-sertifikaatin pohjalta ja kirjaa käyttäjän sisään suojattuun sovellukseen, johon todennustapa on käyttäjätunnus ja salasana. (Hursti 1997.)

Token-mallissa voidaan käyttää hyväksi rauta- tai ohjelmistopohjaisia tokeneita. Token on eräänlainen tunniste, jonka avulla käyttäjä todistaa identiteettinsä. Loppukäyttäjä pyytää identiteetin todistavan tunnisteen (ID Token), jonka avulla voidaan pyytää pääsytunniste (Access Token). Näiden avulla käyttäjät voidaan taas tunnistaa ja kirjata palveluihin. Esimerkkinä nykyteknologiasta Token-pohjaiseen toteutukseen on OpenID Connect, jossa pääsynhallinnasta vastaa OAuth 2.0 –protokolla. Vaihtoehtoisesti token voi olla käyttäjään sidottu kertakäyttösalasanoja luova fyysinen tai ohjelmistopohjainen komponentti, kuten esimerkiksi RSA SecurID. (Hursti 1997.)

Proxyyn pohjautuvassa mallissa käytetään hieman vastaavaa ideologiaa kuin Broker-toteutuksessa. Broker-toteutuksessa keskitetty todennuspalvelin toimii eräänlaisena vahtikoirana, mutta Proxy-toteutuksessa dedikoitu välityspalvelin toimii eräänlaisena ovena muihin verkon palveluihin. (Hursti 1997.)

2.3 SSO vai federointi

SSO-tekniikoita ja palveluntarjoajia on lukuisia ja eri tekniikat soveltuvat paikoin eri tarkoitukseen. Toimialueen sisäinen SSO auttaa käyttäjää kirjautumaan palveluihin yhdellä todennuksella, mutta käyttäjällä on silti edelleen käyttäjätunnus ja salasana jokaiseen erilliseen palveluun. SSO-järjestelmä vain huolehtii näiden lähettämisestä sovellukselle käyttäjän puolesta. (Killeen 2012.)

Toimialueiden välinen SSO taas edellyttää käyttäjän identiteetin federointia, eli edelleen lähettämistä toiselle osapuolelle, ja tämä taas toteutetaan mahdollisesti eri tekniikoilla kuin yrityksen sisäinen SSO. Federoinnissa käyttäjän identiteetti voidaan varmistaa IdP-palvelimelta (Identity Provider) ja IdP-palvelin myöntää tunnisteen (Token) käyttäjälle, jota SSO-järjestelmään sidottu SP (Service Provider) käyttää pääsyn myöntämiseen. (Killeen 2012.)

2.4 Vaihtoehtoja

Seuraavaksi käydään läpi muutamia yleisesti käytössä olevia SSO- ja federointitekniikoita, sekä palveluntarjoajia kyseisille tekniikoille. Itse työssä käytetyt tekniikat käydään lopulta läpi tarkemmin.

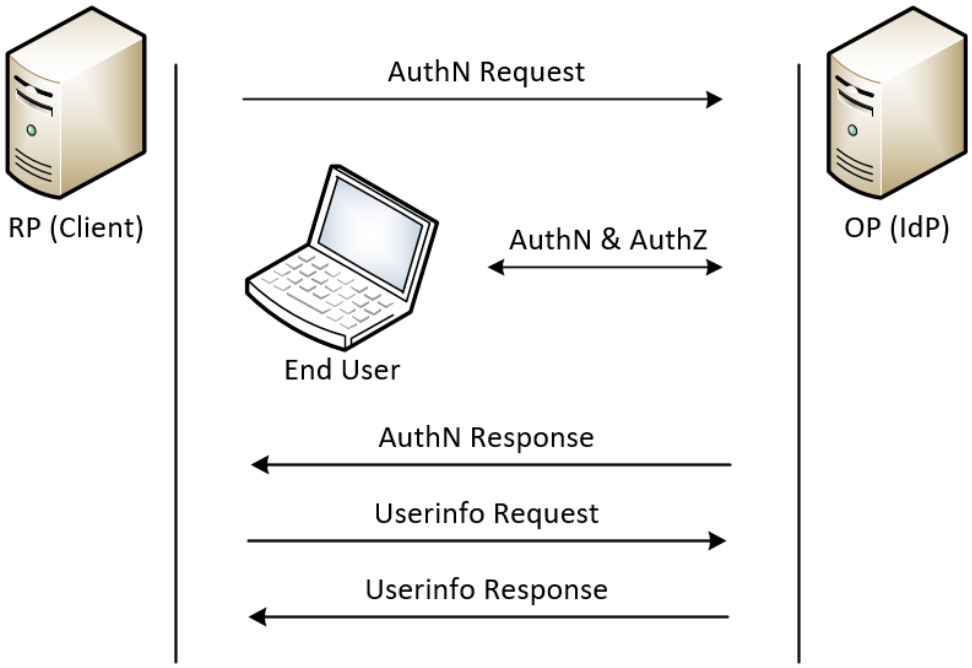
SAML (Security Assertion Markup Language) on XML-pohjainen (Extensible Markup Language) standardi todennus- (autentikaatio) ja valtuuttamistietojen (autorisointi) vaihtoon. SAML pyrkii ratkaisemaan SSO-ongelman ulottuen myös intraverkkojen ulkopuolelle. SAML-järjestelmässä käyttäjän tunnistamisen ja todennuksen hoitaa IdP-palvelin, joka jakaa palveluntarjoajalle, SP:lle, SAML assertion. Assertio on eräänlainen väite, joka sisältää tietoa käyttäjästä ja käyttäjän todennuksesta. Väitteellä käyttäjän identiteetti välitetään (federoidaan) palvelulle ja SP käyttää väitteen tietoja hyväkseen päättäessään pääsystä palveluun. Nykyinen versio, SAML 2.0, on OASIS Security Services Committeen ylläpitämä. (Tarkoma & Kangasharju 2009.)

Shibboleth on standardeihin pohjautuva avoimen lähdekoodin ohjelmistopaketti SSO:n toteuttamiseen. Shibboleth toteutti alun perin omaa versiotaan SAML-standardista, mutta SAML 2.0 versio yhdisti sekä Shibbolethin että OASIS:n ja Liberty Alliancen itsenäisesti kehittämät viitekehykset. Nykypäivän Shibboleth tarjoaa SSO:n ja federoinnin nimenomaan SAML 2.0:n avulla. Shibboleth IdP mahdollistaa käyttäjän tunnistamisen ja todentamisen muun muassa Lightweight Directory Access Protocol eli LDAP-protokollalla, Kerberoksella ja X.509-sertifikaateilla. Shibboleth SP tukee monipuolisesti eri käyttöjärjestelmiä, kuten Windows Server –käyttöjärjestelmiä, Linux-käyttöjärjestelmiä (RedHat, OpenSUSE), ja tuki löytyy jopa OS X- ja Solaris-käyttöjärjestelmille. (Shibboleth N.d.)

CAS (Central Authentication Service) viittaa sekä CAS-protokollaan että tätä protokollaa toteuttavaan palveluun. CAS on tikettipohjainen protokolla ja palvelu itsessään jakautuu kahteen osaan: CAS-palvelimeen ja CAS-asiakkaaseen (client). Palvelin on vastuussa käyttäjän tunnistamisesta ja todentamisesta sekä pääsyn myöntämisestä sovelluksiin. Asiakas on CAS-sovelluksia suojaava komponentti ja se on vastuussa käyttäjän identiteetin noutamisesta CAS-palvelimelta. Yksinkertaisuudessaan protokolla toimii hieman vastaavanlaisella logiikalla kuin Kerberos, jossa käyttäjälle myön-

netään Ticket Granting Ticket, jonka avulla käyttäjä voi noutaa palvelukohtaisia tikettejä. Kerberoksesta poiketen Ticket Granting Ticket tallennetaan evästeeseen, joka toimii olemassa olevan SSO-istunnon tunnisteena. Palvelutiketit lähetetään GET-parametreina pyydettyihin resursseihin ja toimivat todisteena pääsystä CAS-sovelluksille. Tästä voidaan päätellä hyvin helposti, että CAS pohjautuu HTTP-protokollaan (Hypertext Transfer Protocol) ja toiminta palvelimen ja asiakkaiden välillä hoituu määrättyjen URI-päätepisteiden (Uniform Resource Identifier) kautta. Oman CAS-protokollansa lisäksi CAS-palvelu tukee myös OpenID-, OAuth- ja SAML-protokollia. Paikallisena käyttäjän todentaminen hoituu Shibbolethin tavoin esimerkiksi LDAP-protokollalla tai X.509-sertifikaateilla. (CAS N.d.)

OIDC (OpenID Connect) on uudempi token-pohjainen federaatiotekniikka. OIDC-todennusprotokolla toimii OAuth 2.0 –protokollan päällä tarjoten sovelluksille mahdollisuuden todentaa käyttäjä luotettua IdP:tä vasten. IdP on mahdollista toteuttaa itse ja tarjota organisaation SSO tätä kautta. On myös mahdollista käyttää kolmannen osapuolen IdP:tä, joita tarjoavat muun muassa Google ja Microsoft. OpenID Connectin pohjalla toimiva OAuth 2.0 –protokolla määrittää todennuksessa ja valtuuttamisessa käytettävän viestien vaihdon. Itse viestit ovat JSON-muotoisia (JavaScript Object Notation) ja kulkevat HTTP:n yli. Tämän vuoksi OIDC-toteutusten tulee aina käyttää TLS-suojattua (Transport Layer Security) yhteyttä. Normaali käyttäjän todennus noudattaa seuraavan sivun kuvion 2 periaatetta. (OpenID Connect 2014.)



Kuvio 2. OpenID Connect –todennus (OpenID Connect 2014.)

Relaying Party, RP, on jokin palvelu, jonka käyttäjien todennus tapahtuu OIDC:n avulla. Käyttäjän ottaessa RP:n yhteyttä, RP lähettää todennuspyynnön (AuthN) OP:lle (OpenID Connect Provider). OP todentaa (AuthN) ja valtuuttaa (AuthZ) loppukäyttäjän käyttäen Authorization päätepistettä (endpoint). Tämän jälkeen OP vastaa alkuperäiseen todennuspyyntöön lähettämällä asiakkaalle ID Tokenin ja yleensä myös Access Tokenin. RP voi käyttää Access Tokenia lähettääkseen pyyntöjä UserInfo päätepisteelle, joka palauttaa tietoja (Claims) loppukäyttäjästä. (OpenID Connect 2014.)

ADFS (Active Directory Federation Services) on Microsoftin ohjelmistokomponentti kertakirjautumiseen ja identiteetin federointiin. Microsoft ADFS käyttää federointiin WS-Federation-, SAML- ja nykyisin myös OpenID Connect-protokollia. Etu ADFS:n käytössä on sen suora yhteensopivuus Active Directory Domain Services –palveluun. Uusin versio ADFS-palvelusta tukee myös muita LDAPv3-yhteensopivia hakemistoja, joten komponentin käyttö ei ole rajattu Microsoft-ympäristöön. Työssä testattiin Windows Server 2016 –käyttöjärjestelmällä varustettu palvelin OpenID Connect Provider -roolissa. (Mathers 2016.)

3 Kerberos

3.1 Yleistä

Kerberos sai alkunsa tutkimusprojektina Massachusetts Institute of Technology –yliopistossa 1980-luvulla. Ajanjakson vallitsevat etäkäyttöprotokollat, kuten Telnet, lähettivät käyttäjän tunnukset selväkielisinä verkon yli. Lisäksi hallittavien koneiden ja palvelujen määrä kasvoi huimaa tahtia ja eri laitteet ja palvelut vaativat aina omat käyttäjätunnukset salasanoineen. Ongelmien ratkaisemiseksi kehitettiin uusi käyttäjäntodennusprotokolla: Kerberos. (Garman 2003, 1-5.)

Ensimmäinen MIT:n ulkopuolelle jaettu versio Kerberoksesta oli Kerberos 4. Laajempi levinneisyys jäi kuitenkin saavuttamatta, sillä salausohjelmistojen vientiä kontrolloitiin ja rajoitettiin Yhdysvaltain valtion toimesta. Kerberos 4 toteutuksia on edelleen olemassa, mutta käytännössä protokollan kyseistä versiota pidetään epäturvallisena, ja Kerberoksen uusin versio, Kerberos 5, on muutenkin huomattavasti paranneltu ja myös saanut lukuisia uusia ominaisuuksia. Näihin lukeutuvat muun muassa laajempi tuki salausmenetelmille, tunnuksien delegointi ja edelleen lähettäminen sekä tuen parantaminen toimialueiden välisessä todennustransaktiossa. (Garman 2003, 8-9.)

Viimeisin virallinen parhaiden käytäntöjen ohjeistus on julkaistu RFC 6649 –dokumentissa ja se käsittelee heikkojen kryptografisten algoritmien poistamista käytöstä. RFC 6649 ehdottaa, että DES eli Data Encryption Standardiin pohjautuvat algoritmit tulisi jättää kokonaan pois tuen piiristä, kuten myös RC4-HMAC-EXP. On kuitenkin huomioitava, että vanhat versiot Microsoft Windows -käyttöjärjestelmistä eivät tue uudempia algoritmeja kuin RC4-HMAC, joten vielä ei välttämättä ole mahdollista aktiivisesti ajaa kyseistä kryptoalgoritmia pois protokollatoteutuksista. (RFC 6649 2012.)

3.2 Toiminta

Kerberos on protokolla, jolla tarjotaan turvallinen kertakirjautuminen käyttäen hyväksi luotettua kolmatta osapuolta. Turvallisuus taataan käyttämällä salasanojen sijaan tikettejä, eräänlaisia aikaleimoihin nojautuvia kryptografisia viestejä, joilla käyt-

täjän identiteetti voidaan varmistaa ilman salasanan lähettämistä. Tästä syystä Kerberos sopii hyvin käytettäväksi suojaamattomissa verkoissa. Kertakirjautuminen toteutetaan todentamalla käyttäjä kerran, esimerkiksi työasemalle kirjautuessa. Tämän jälkeen Kerberos-tiketti huolehtii käyttäjän kirjaamisesta muihin järjestelmään sidottuihin palveluihin. Palvelun kerberoinnista puhutaan silloin, kun jonkin palvelun käyttäjien todennukseen sovelletaan Kerberos-protokollaa. (Garman 2003, 6-7.)

Kerberos toimii molempiin suuntiin. Asiakas eli loppukäyttäjä varmennetaan ja lisäksi myös palvelin varmentaa loppukäyttäjälle itsensä. Näin molemmat osapuolet varmistuvat siitä, että kumpikin kommunikaation osapuoli on juuri se, joka väittääkin olevansa. Turvalliseksi tämän transaktion tekee salauksen, eheyden tarkistuksen ja aikaleimojen käyttäminen. Verkon ja verkon laitteiden, joissa Kerberosta käytetään, täytyykin olla synkronoitu suurin piirtein samaan aikaan. (RFC 4120 2005.)

3.3 Key Distribution Center

Järjestelmän kulmakivenä toimii KDC (Key Distribution Center), joka koostuu kolmesta loogisesta komponentista. Nämä ovat järjestelmän sisäinen tietokanta, johon on koottu kaikkien käyttäjien päänimet (Principal Name) ja niihin sidotut kryptoavaimet, todennuspalvelin ja TGS (Ticket Granting Server). KDC voidaan varmentaa, ja myös tulisi varmentaa, eli KDC palvelimia voi olla useita. KDC:n sisäinen tietokanta voidaan säilyttää Active Directoryn LDAP (Lightweight Directory Access Protocol) rakenteessa, mutta avoimen lähdekoodin ratkaisut tukevat myös tiedon säilyttämistä erillisessä kevyessä tietokannassa. (Garman 2003, 20-21.)

Todennuspalvelin, AS (Authentication Server), jakaa kirjautumisessa asiakkaille TGT-tiketin (Ticket Granting Ticket). TGT-tiketti varmistaa, että loppukäyttäjän ei erikseen tarvitse todentaa identiteettiään KDC:lle, sillä TGT-tiketti salataan loppukäyttäjän salasanalla, joka on tiedossa vain kyseisellä käyttäjällä itsellään ja KDC-palvelimella. TGT:llä pyydetään yksittäiset palvelutiketit ja tämä poistaa tarpeen salasanan uudelleensyöttämiselle jokaiseen erilliseen palveluun. (Garman 2003, 20-21.)

Ticket Granting Server, eli TGS, on vastuussa yksittäisten palvelutikettien jakamisesta asiakkaille pyyntöä vastaan. Pyynnössä käytetään TGT-tikettiä ja halutun palvelun päänimeä. Päänimi on käyttäjä- (User Principal Name) ja palvelukohtainen (Service

Principal Name) nimi, joka on ainutlaatuinen Kerberos-toimialueella. TGS validoi TGT-tiketin varmistamalla, että se on salattu Kerberos palvelimen TGT-avaimella, ja luo tämän jälkeen asiakkaalle palvelutiketin. (Garman 2003, 20-21.)

Jokaisella palveluun sidotulla päänimellä on lisäksi oma salausavaimensa, kuten jokaisella päänimellä muutenkin. Windows-koneilla nämä luodaan lennosta tarpeen vaatiessa, eli käytännössä kerberoituja palveluja asennettaessa. Linux-palvelimilla nämä palveluavaimet on sidottu omaan tiedostoonsa, jota kutsutaan nimellä keytab. Jokainen järjestelmään integroitu palvelu tulisi sitoa omaan keytab-tiedostoonsa ja palvelua tulisi ajaa omalla Linux-käyttäjätilillä. Tämä mahdollistaa sen, että keytab-tiedoston lukuoikeus voidaan määrittää vain palvelun ajossa käytettävälle tilille. (Garman 2003, 138).

3.4 Tiketit

Kerberos-tiketti on KDC:n jakama kryptattu datarakenne ja sisältää jaetun salausavaimen, joka on istuntokohtaisesti ainutlaatuinen. Tiketti sisältää lippuja ilmoittamaan, mihin tikettiä voidaan käyttää. Liput kertovat esimerkiksi, voidaanko tiketti välittää toiselle palvelulle. Tiketti varmistaa istunnon molempien osapuolten identiteetin ja muodostaa osapuolten välille salausavaimen nimeltään istuntoavain (Session Key). Kuviossa 3 esitetään Kerberos-tiketin rakenne. (Garman 2003, 21-22.)

```
Client: h3334 @ AD.JAMK.FI
Server: krbtgt/AD.JAMK.FI @ AD.JAMK.FI
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 10/11/2016 16:52:40 (local)
End Time:   10/12/2016 2:52:40 (local)
Renew Time: 10/18/2016 16:52:40 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: DC2
```

Kuvio 3. Kerberos-tiketin rakenne

Jokainen tiketti sisältää vähintään seuraavat kentät: Käyttäjän päänimi (h3334@AD.JAMK.FI), palvelun päänimi (krbtgt/AD.JAMK.FI@AD.JAMK.FI), voimassaoloaika (Start, End, Renew time) ja istuntoavain (Session Key). Lisäksi lipuista (Ticket flags) käy ilmi kyseisen tiketin ominaisuudet. Esimerkiksi forwardable-lippu kertoo,

että kyseinen TGT-tiketti voidaan lähettää edelleen toiselle isäntäkoneelle kirjautumisen yhteydessä. Tällöin alkuperäistä TGT-tikettiä voidaan käyttää uudessa kohteessa ja kertakirjautuminen toteutuu jopa sisäkkäisissä sovelluksissa. Esimerkkinä tästä voidaan mainita tilanne, jossa käyttäjä kirjautuu Windows-työasemaltaan kerberoidun SSH-palvelun kautta Linux-palvelimelle ja edelleen Linux-palvelimelta toiseen kerberoituun palveluun. (Garman 2003, 42).

Koska tiketit ovat palvelukohtaisia, on mahdollista, että yhtä aikaa on voimassa useita tikettejä. Nämä säilötään Ticket Cache –nimellä kulkevaan rakenteeseen, joka voi olla esimerkiksi tiedosto. Tiedoston sijaan Microsoftin Active Directory implementoi muistipohjaisen Ticket Cachen, jolloin voidaan varmistua siitä, että olemassa olevat tiketit terminoidaan istunnon päättyessä. (Garman 2003, 21-22.)

3.5 Transaktio

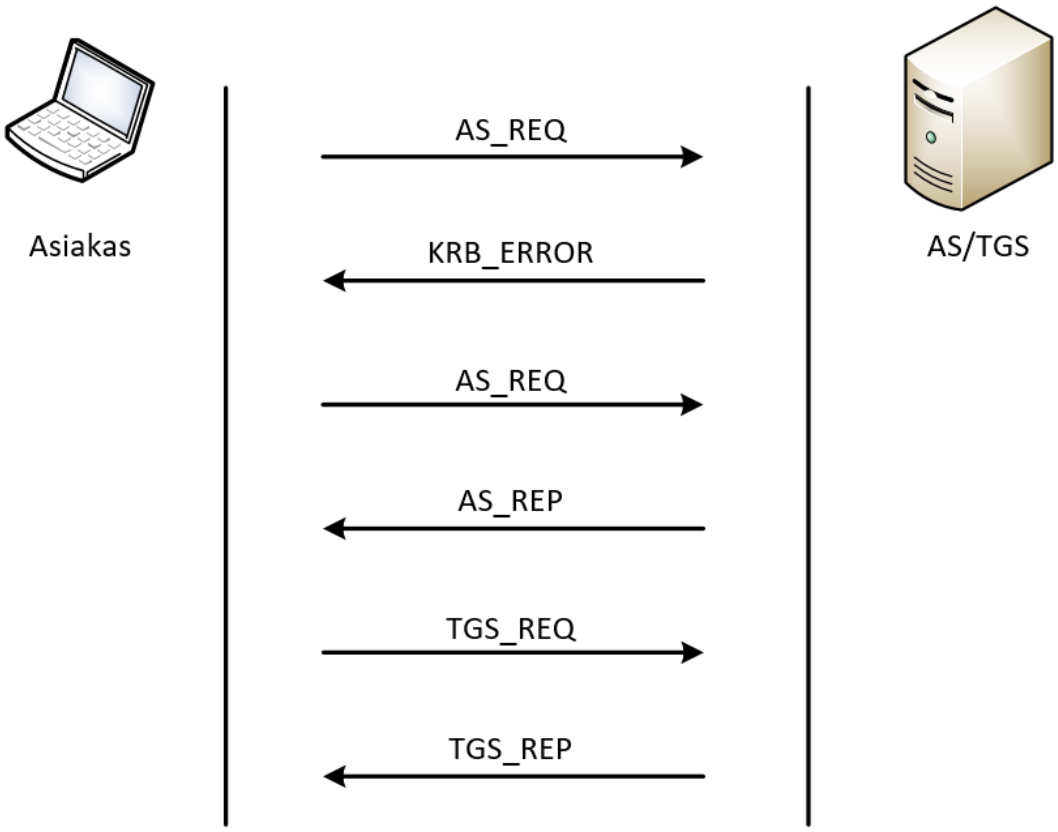
Normaali Kerberos-transaktio alkaa AS_REQ-pyynnöllä AS-palvelimelle TGT-tiketin saamiseksi. Palvelin vastaa AS_REP-viestillä, joka sisältää asiakkaan TGT-tiketin. Tämän jälkeen pyydetään palvelutiketti TGS_REQ-viestillä TGS-palvelimelta käyttäen TGT-tikettiä ja halutun palvelun päänimeä. TGS-palvelin kuittaa pyynnön validoinnin jälkeen TGS_REP-viestillä. Tämän jälkeen loppukäyttäjä voi tehdä kerberoiduille palveluille pyynnön päästä käsiksi resurssiin. Onnistunut pyyntö AS-palvelimelle ei siis automaattisesti tarkoita, että käyttäjällä on tämän jälkeen pääsy kaikkiin kerberoituihin palveluihin, sillä valtuuttaminen voi olla palvelukohtaista. (Kerberos 2009; Garman 2003, 34-41).

Transaktio alkuperäisenä Kerberos 4 –protokollatoteutuksena oli kuitenkin haavoittuvainen. Koska KDC-palvelin antaa käyttäjän salausavaimella salatun TGT-tiketin kenelle tahansa sitä pyytävälle, oli mahdollista pyytää kyseinen tiketti, ja sitä kautta yrittää murtaa salausavain brute force –tekniikalla. Monesti käyttäjät valitsivat kehnoja salasanoja, joka tietenkin edesauttoi hyökkääjää. (Garman 2003, 43).

Kerberos 5 implementoi esitodennuksen (Preauthentication) korjaamaan tämän haavoittuvaisuuden. Ensimmäinen AS_REQ pyyntö AS-palvelimelle epäonnistuu ja palauttaa KRB_ERROR-viestin, jolloin asiakkaan täytyy todentaa henkilöllisyytensä pal-

velimelle ennen kuin TGT-tiketti myönnetään. Käytännössä asiakaslaite todentaa itsensä lähettämällä uuden AS_REQ viestin, joka sisältää aikaleiman, joka salataan asiakkaan avaimella. (Garman 2003, 43).

Kuviossa 4 esitetään vielä koko edellä mainittu transaktio sekä asiakkaan että palvelimen näkökulmasta. Kuviosta käy myös selkeästi ilmi, mitä viestejä transaktioon kuuluu ja kumpi osapuoli toimii kunkin viestin lähettäjänä.



Kuvio 4. Kerberos-todennus (Garman 2003, 35.)

4 HTTP-todennus

4.1 Yleistä

Hypertext Transfer Protocol on tilaton sovellustason protokolla, jolla mahdollistetaan tiedonsiirto hypertekstiin perustuvien tietojärjestelmien välillä. HTTP on suunniteltu hyvin yleisluontoiseksi protokollaksi, jonka tarkoitus on muodostaa rajapinta eri tietojärjestelmiin ja se toimii pyyntö-vastaus periaatteella. Asiakas kohdistaa palvelimeen pyynnön, johon sisällytetään tietty metodi, tietty kohderesurssi eli Uniform Resource Identifier, käytetty HTTP-protokollan versio ja muita lisätietoja HTTP-otsikoiden muodossa. Palvelin palauttaa pyyntöön vastauksen, joka sisältää tiedot pyydetyn URI:n sijainnista, HTTP-protokollan versiosta ja tilakoodin (Status Code), joka kertoo lisätietoa pyynnön onnistumisesta. Lisäksi vastaukseen sidotaan lisäinformaatiota antavia HTTP-vastausotsikoita ja lopulta myös mahdollinen hyötykuorma. (RFC 7230 2014.)

Pyyntöihin sidottuja metodeja on useita, mutta tämän työn kannalta keskitytään vain oleellisimpaan. Näistä tärkein on GET-metodi, joka on ensisijainen tapa noutaa informaatiota HTTP-protokollan avulla. Yksinkertaisimmillaan transaktio tapahtuu esimerkiksi asiakkaana toimivan selaimen ja web-palvelimen välillä. Käyttäjä haluaa siirtyä selaimellaan selaamaan HTTP-protokollan toiminnasta kertovaa dokumentaatiota ja kirjoittaa selaimen osoiteriville: (RFC 7230 2014.)

https://tools.ietf.org/html/rfc7230

Selain tekee tällöin seuraavan pyynnön käyttäen GET-metodia:

GET /html/rfc7230 HTTP/1.1
Host: tools.ietf.org

Palvelin palauttaa vastauksen seuraavanlaisesti:

HTTP/1.1 200 OK
Content-Location: rfc7230.html
Content-Type: text/html; charset=UTF-8

Tilakoodi, 200 OK, kertoo pyynnön onnistumisesta ja vastauksessa kerrotaan pyydetyn resurssin sijainti ja hyötykuorman tyyppi. Lisäksi HTTP-otsikoilla voidaan neuvotella informaation esitystapa ja antaa tietoa esimerkiksi yhteyttä ottavasta asiakkaasta, User-Agentista. Pyyntöön vastaavasta palvelimesta voidaan palauttaa tietoa käyttäen Server-otsikkoa. Kuviossa 5 on esitetty yksinkertainen pyyntö-vastaus –transaktio edellä esitetystä tilanteesta.

Kuvio 5. HTTP-pyyntö ja –vastaus

Palvelimen palauttamat tilakoodit voidaan jakaa karkeasti viiteen kategoriaan numeroinnin perusteella. Tilakoodi muodostuu kolmesta numerosta ja aloittava numero määrittää kategorian. Tilakoodien luokat voidaan määritellä seuraavanlaisesti: (RFC7231 2014.)

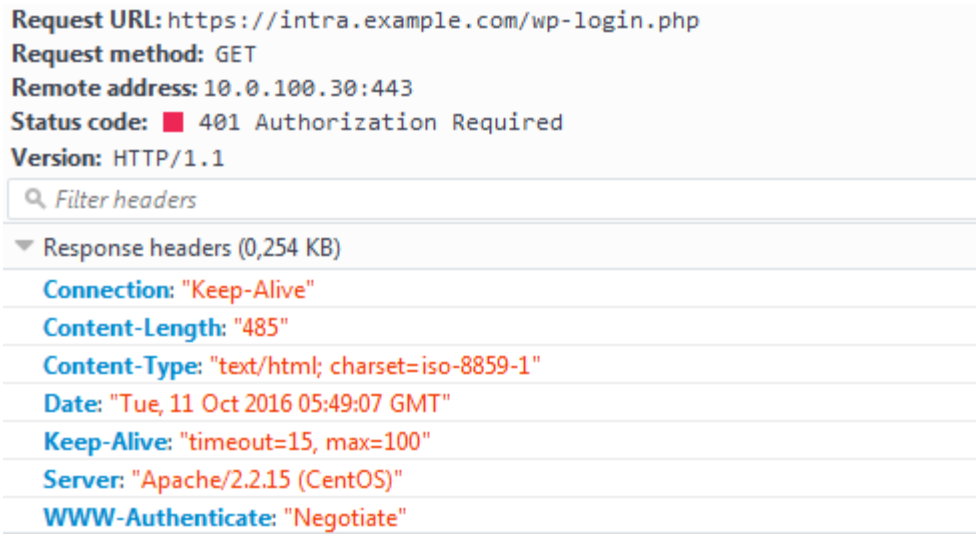
1. Informational 1xx: Pyyntö otettiin vastaan ja prosessointi jatkuu.
2. Succesful 2xx: Pyyntö otettiin onnistuneesti vastaan, ymmärrettiin ja hyväksyttiin
3. Redirection 3xx: Pyynnön suorittamiseksi täytyy tehdä lisätoimenpiteitä
4. Client Error 4xx: Pyyntö on virheellinen tai sitä ei voida suorittaa
5. Server Error 5xx: Palvelin epäonnistuu suorittamaan pyynnön, vaikka pyyntö on validi.

Näistä monelle käyttäjälle tuttuja ovat esimerkiksi tilakoodi 404, joka kertoo, että pyydettyä resurssia ei löydy ja tilakoodi 500, Internal Server Error, joka kertoo palvelinpään asetusvirheestä (RFC7231 2014). HTTP-otsikot mahdollistavat myös käyttäjän todentamisen, jolloin asiakkaalle ilmoitetaan vaaditusta todennuksesta tietyllä tilakoodilla ja palvelimen vastaukseen sisällytetyissä otsikoissa käytetään todennukseen liittyvää lisäotsikkoa. (RFC 7235 2014.)

Muita metodeja ovat POST, HEAD, PUT, DELETE, CONNECT, OPTIONS ja TRACE. POST-metodia käytetään yleisesti esimerkiksi HTML-lomakkeiden (Hypertext Markup Language) tietojen lähettämisessä prosessoitavaksi. Monien web-sovellusten kirjautumissivu hyödyntääkin HTML-lomaketta ja POST-metodia. (RFC7231 2014).

4.2 Toiminta

HTTP-todennuksen (HTTP-authentication) vaativa palvelu vastaa pyyntöön 401 Unauthorized -tilakoodilla. Kyseisen palvelimen täytyy tällöin lisätä vastaukseen WWW-Authenticate –otsikko. Kuviossa 6 esitetään kyseinen toiminne. Resurssia pyydettäessä palvelin vastaa 401-tilakoodilla ja lisää vastausotsikoihin WWW-Authenticate –kentän. (RFC 7235 2014.)

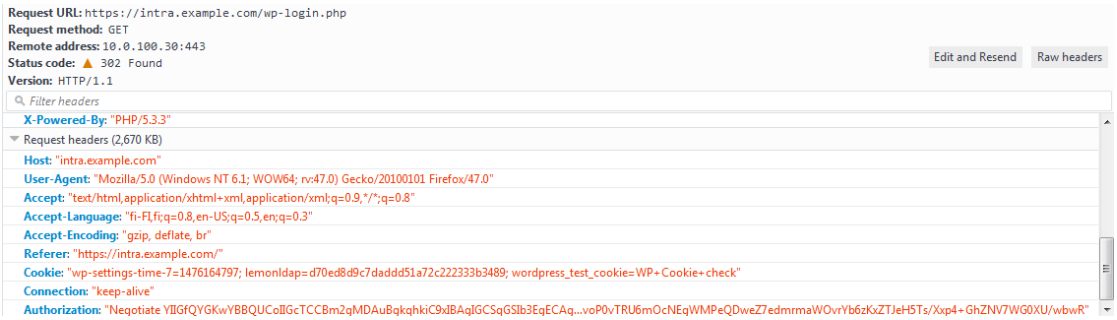


Kuvio 6. HTTP-todennus

Seuraavaan pyyntöön asiakas lisää Authorization-otsikon, jonka sisällöllä neuvotellaan käyttäjän todennus. Yksinkertaisimmillaan Authorization-otsikko voi olla BASE64

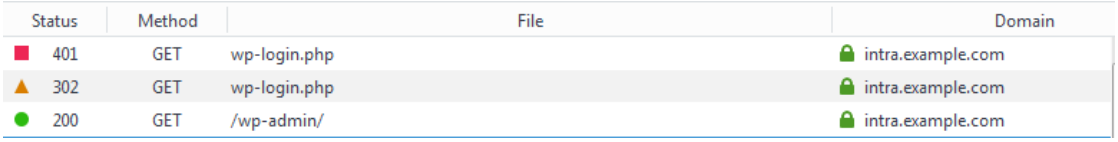
enkoodaus käyttäjän tunnuksesta ja salasanasta, muodossa ”Authorization: Basic . base64($username . ’:’ . $password)”. (RFC 7235 2014.)

Kuviossa 7 esitetään Windows-asiakkaan selaimen ja Linux web-palvelimen välillä tapahtuva todennusneuvottelu, jossa valitaan Kerberos- tai NTLM-protokollan (NT Lan Manager) välillä. Kerberosta käytetään ensisijaisesti, jos kyseinen palvelu on saatavilla (Understanding HTTP Authentication N.d.)



Kuvio 7. HTTP-pyyntö sisältäen Authorization-otsikon

Kuviossa 8 esitetään vielä koko todennuksen eteneminen. Palvelin vastaa pyyntöön ensin edellä mainitun mukaisesti statuskoodilla 401 ja pyytää todennusta. Todennus suoritetaan lisäämällä pyyntöön Authorization-otsikko, ja jos käyttäjälle on sallittu pääsy resurssiin, vastaa palvelin pyyntöön statuskoodilla 200 OK.



Kuvio 8. HTTP-todennuksen kulku

Todennuksen epäonnistuessa palvelin vastaa tilakoodilla 403 Forbidden. Tämä statuskoodi kertoo, että palvelin on ymmärtänyt pyynnön, mutta kieltäytyy antamasta pääsyä resurssiin. Palvelin voi myös halutessaan palauttaa vastauksen höytykuormassa tiedon siitä, miksi pyyntö evättiin. (RFC 7231 2014.)

Moni työssä käytetyistä palveluista tarjoaa mahdollisuuden käyttää HTTP-todennusta käyttäjän tunnistamiseen. Tämä toteutetaan sovelluksissa niin, että HTTP-todennuksen tekevä liitännäinen tai ominaisuus ei varsinaisesti tee itse todennusta,

vaan tarkistaa onko todennus jo tehty. Jos on, niin käyttäjä kirjataan palveluun sisään. Hyvien käytänteiden mukaisesti olemassa oleva HTTP-todennus tarkistetaan yleensä etsimällä tiettyä muuttujaa sovellusta pyörittävän web-palvelimen tiedoista. Usein, kuten esimerkiksi Apache web-palvelimella, tämä muuttuja on $SERVER[’REMOTE_USER’]. Muuttuja asetetaan onnistuneen HTTP-todennuksen avulla vastaamaan loppukäyttäjän käyttäjätunnusta. Huomioitavaa tässä on se, että jos sovelluksen HTTP-todennuksen toteuttava liitännäinen ei ota kantaa kuin olemassa olevan muuttujan sisältöön, voidaan itse muuttujan asettamiseen käyttää lukuisia eri keinoja. (RFC 3875 2004.)

5 Kaksivaiheinen todennus

5.1 Yleistä

Autentikointi eli todennus on prosessi, jonka avulla käyttäjä todistaa identiteettinsä. Karkeasti jaoteltuna käyttäjän identiteetin voi todistaa kolmella eri tavalla. Näistä ensimmäinen on käyttäjän todentaminen jonkin käyttäjän omistaman asian avulla. Tällainen asia voi olla esimerkiksi henkilökortti. Toinen tapa todentaa käyttäjä perustuu käyttäjän itsensä ominaisuuksiin. Tällainen ominaisuus voi olla esimerkiksi sormenjälki tai verkkokalvo, eli jokin ainutlaatuinen ominaisuus, joka on monesti liitettävissä biometriikkaan. Kolmas tapa käyttäjien todentamiseen on monelle perinteisesti tuttu. Tämä perustuu siihen, mitä käyttäjä tietää ja tämä voi olla esimerkiksi salasana tai PIN-numero. Kolme eri tapaa todentaa käyttäjä voidaan siis tiivistää seuraavasti: (Stanislav 2015.)

1. Jotain mitä omistat
2. Jotain mitä olet
3. Jotain mitä tiedät

Yleisesti käytetty todennustapa, käyttäjätunnus ja salasana, on riskialtis monestakin syystä. Käyttäjät ovat laiskoja ja käyttävät ennalta-arvattavia tai lyhyitä salasanoja tai kirjoittavat salasanansa muistilapuille. Phishing-kampanjat ja keylogger-ohjelmistot ovat myös suosittuja ja tehokkaita tapoja saada käyttäjän kirjautumistiedot selville. Ratkaisu ongelmaan ei ole pidempien ja hankalampien salasanojen käyttö ja käyttäjän pakottaminen vaihtamaan salasanansa usein. Tämä vain johtaa heikompiin salasanoihin. Tehokkaampi keino on ottaa käyttöön toinen tai useampi edellä mainituista todennustavoista. (Stanislav 2015.)

Vahva todennus tarkoittaa käyttäjän tunnistamista käyttäen jotain pelkän salasanan lisäksi. Tämä saavutetaan ottamalla rinnalle esimerkiksi toinen edellä mainituista kategorioista. Kaksivaiheiseksi todennukseksi (Two-factor authentication eli 2FA) kutsutaan käyttäjän todennusta, jossa käytetään kahta yllä mainituista todennustavoista. Tuttu esimerkki tästä on pankkiautomaatilla asiointi, jossa käyttäjä tarvitsee sekä pankkikortin (jotain mitä omistat) että PIN-koodin (jotain mitä tiedät). Monivaiheinen

todentaminen (Multi-factor authentication eli MFA) sen sijaan käyttää kahta tai useampaa yllä mainituista todennustavoista. Lisäksi on vielä erotettavissa kaksivaiheinen vahvistus (Two-step verification eli 2SV), jossa käytetään kahta erillistä, toisistaan riippumatonta tunnistusvaihetta. Nämä eivät välttämättä ole eri kategorioista, vaan käyttäjältä voidaan vaatia esimerkiksi kaksi erillistä salasanaa ennen järjestelmään pääsyä. (Stanislav 2015.)

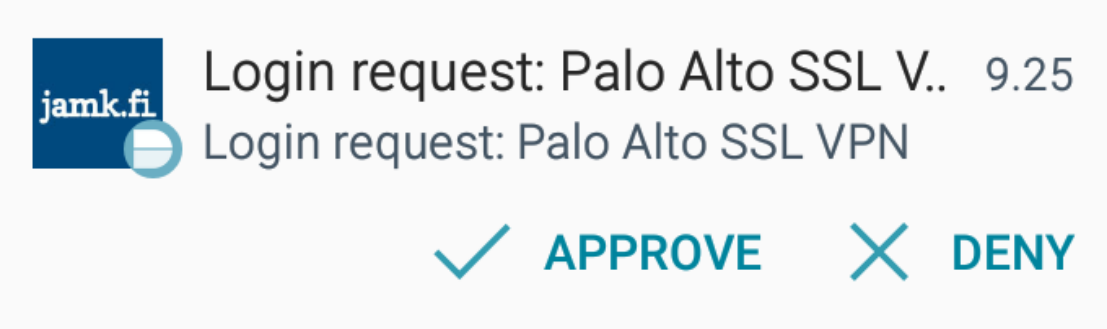
5.2 Vaihtoehtoja

Vaihtoehdot kaksivaiheisen todennuksen toteuttamiseen ovat lukuisat. Tunnetuimpia lienevät kuitenkin rauta- ja ohjelmistopohjaiset tokenit, joilla generoidaan satunnaisia kertakäyttösalasanoja. Toteutustapa, eli käytännössä algoritmi, vaihtelee hieman eri valmistajan tai palveluntarjoajan mukaan, mutta myös avoimiin standardeihin perustuvia toteutuksia on runsaasti tarjolla. Näitä ovat esimerkiksi HOTP- (HMAC-Based One-Time Password) ja TOTP- (Time-based One-Time Password) algoritmit, joita voidaan käyttää hyväksi esimerkiksi Googlen autentikaattoripalvelulla tai sen avoimeen lähdekoodiin perustuvalla versiolla. (Google Authenticator 2016; Stanislav 2015.)

HOTP- ja TOTP-algoritmit perustuvat samankaltaiseen perusideaan. OTP-salasana (One-Time Password) muodostetaan käyttämällä asiakkaan ja palvelimen välistä jaettua avainta sekä jonkinlaista laskuria. HOTP-toteutuksessa laskurina toimii 8-tavun mittainen counter, jonka täytyy olla synkronoitu asiakkaan ja palvelimen välillä. Jokaisella salasanan luomiskerralla laskuriin lisätään vakioitu arvo. TOTP-toteutuksessa laskurina toimii aika. HOTP-salasana vaihtuu siis joka generoinnilla, kun taas TOTP-salasana vaihtuu asetetun aikavälin jälkeen. Huomioitavaa on kuitenkin, että vaikka TOTP-salasanan luonnin aikaväli voi olla esimerkiksi kolmekymmentä sekuntia, on salasana silti kertakäyttöinen. (RFC 4226 2005; RFC 6238 2011.)

Nykymaailman älylaiteähkyssä voidaan valjastaa käyttäjän tunnistukseen myös muut älylaitteiden toiminnot. Eräs esimerkki tästä on push-viestien käyttö kaksivaiheisessa todennuksessa. Push-viestillä on mahdollista siirtää pieni määrä dataa palvelusta asiakkaan laitteelle. Viesti tulee suoraan näkyviin loppukäyttäjän laitteelle, jolloin se herättää käyttäjän huomion välittömästä. Push-viesti voi sisältää esimerkiksi tarvittavan

toimenpiteen, jolla aktivoidaan kyseisen palvelun mobiilisovellus ja soveltuu täten erinomaisesti sellaisille käyttäjille, jotka eivät ehkä ole teknisesti kovin perehtyneitä tai haluavat vähän vaivaa vaativia ominaisuuksia. Kaksivaiheisessa todennuksessa push-viesti voidaan lähettää esimerkiksi onnistuneen käyttäjätunnus/salasana –varmennuksen jälkeen, ja se voi sisältää esimerkiksi Hyväksy/Hylkää –toiminteet. Lisäksi viestistä on hyvä löytyä kohdepalvelun nimi ja aikaleima. Kaksivaiheinen todennus push-viestejä käyttämällä on todella vaivatonta loppukäyttäjän näkökulmasta, sillä käyttäjän ei tarvitse syöttää kuin käyttäjätunnuksensa ja salasanansa, ja tämän jälkeen hyväksyä vahvistusviesti. (Stanislav 2015.) Kuviossa 9 esitetään esimerkki tällaisesta push-viestistä.



Kuvio 9. Push-viesti

Muita yleisesti käytössä olevia tunnistusmenetelmiä ovat muun muassa tekstiviesteillä toimitettavat OTP-salasanat, puhelinsoitolla toimitettavat OTP-salasanat, äly- ja sirukortit sekä erilaiset biometriset tunnisteet. Tekstiviestit eli SMS-viestit (Short Message Service) ovat olleet käytössä jo yli kolmekymmentä vuotta ja täten ovat luonnollisesti tuettuja myös niissä mobiililaitteissa, joissa ei varsinaisia älypuhelimen toiminteita ole. Sama koskee puhelinsoitolla toimitettavia tunnistetietoja. (Stanislav 2015.)

Biometriset tunnisteet sisältävät esimerkiksi sormenjäljet, silmän iiriksen muodon tai vaikkapa käyttäjän äänen. Siinä missä OTP-salasanan täytyy olla identtinen odotettuun arvoon verrattuna, täytyy biometrisissa tunnisteissa turvautua kynnysarvoihin. Sormenjäljen tilastollisen analyysin täytyy tuottaa tarpeeksi lähellä vertailuarvoa oleva tulos, jotta todennus onnistuu. Täysin identtiseen tulokseen vertailuarvon kanssa on lähes mahdotonta päästä. Tämä myös asettaa biometristen tunnisteiden

käytölle rajoituksia, sillä liian löysät raja-arvot mahdollistavat järjestelmän väärinkäytön ja liian kireät raja-arvot saattavat johtaa virheisiin todennuksessa. (Stanislav 2015.)

5.3 Palo Alto

Palo Alton SSL-VPN -yhteydelle löytyy tuki kolmelle eri kaksivaiheisen todennuksen mahdollistavalle tavalle. Ensimmäisessä käyttäjä on mahdollista todentaa käyttäjätunnuksen, salasanan ja asiakassertifikaatin yhdistelmällä. Tässä tapauksessa käyttäjälle luodaan sertifikaattiprofiili (Certificate Profile) ja todennusprofiili (Authentication Profile). Käyttäjän täytyy todentaa itsensä molempiin onnistuneesti, jotta VPN-yhteys muodostuu. (Palo Alto – 2FA. N.d.)

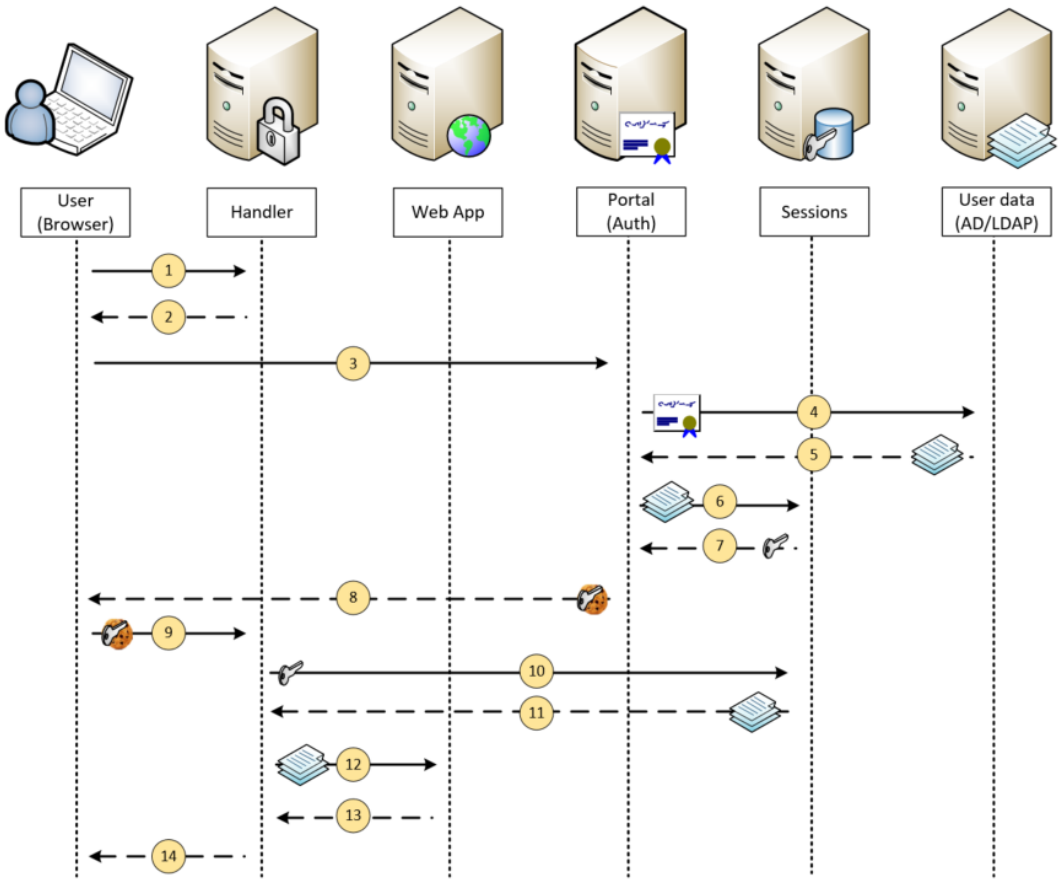
Toinen tapa toteuttaa kaksivaiheinen todennus on sitoa käyttäjän tunnistus Radius-profiiliin. Itse Radius-palveluun voidaan tällöin sitoa jokin OTP-palvelu. Käyttäjä kirjautuu käyttäen käyttäjätunnusta, salasanaa ja OTP-salasanaa. Radiuksen kautta voidaan sitoa myös mikä tahansa muu 2FA-palvelu. (Palo Alto – 2FA. N.d.)

Kolmas tapa toteuttaa 2FA Palo Alton palomuuriin on käyttää älykortteja. Virtualisoidun ympäristön puolesta tämä hylättiin heti vaihtoehtona toteutukselle. Itse älykortti sidottaisiin VPN-palveluun myös sertifikaattiprofiilin avulla. (Palo Alto – 2FA. N.d.)

6 LemonLDAP

6.1 Toiminta

LemonLDAP:n toiminta perustuu Perl-ohjelmointikielellä toteutettuihin moduuleihin joista koostetaan kolme pääkomponenttia: konfiguraatiokantaa käsittelevä Manager, käyttäjän todentamisesta ja istunnoista vastaava Portal ja sovellusten suojelusta ja käyttäjän uudelleenohjauksesta vastaava Handler. Nämä komponentit voivat sijaita samassa palvelimessa, mutta se ei ole pakollista, ja varsinkin Handler on usein etäkomponentti. Kuviossa 10 esitetään LemonLDAP-järjestelmän toiminta. (LemonLDAP – Presentation N.d.)



Kuvio 10. LemonLDAP:n toiminta (LemonLDAP – Presentation N.d.)

Käyttäjän pyyntö järjestelmään integroidulle web-sivulle jää kiinni Handler-komponenttiin. Handler-komponentti tutkii pyynnön HTTP-otsikoita ja havaitsee, että pyynnöstä puuttuu järjestelmän oma eväste (1). Handler-komponentti

uudelleenohjaa käyttäjän kirjautumaan Portal-komponentille (2-3). Kirjautuminen Portalissa tapahtuu käyttäjätietokantaa vasten, joka voi olla esimerkiksi Active Directory. Kirjautumisen onnistuessa Portal-komponentti kerää käyttäjätiedot käyttäjäkannasta (4-5). Käyttäjätiedoilla luodaan istunto ja istuntoa varten luodaan istuntoavain, joka sidotaan kyseiseen istuntoon (6-7). Eväste SSO:ta varten luodaan käyttäen evästeen arvona istuntoavainta ja käyttäjä uudelleenohjataan takaisin alkuperäiseen sovellukseen (8-9). Handler-komponentti huomaa uudessa pyynnössä evästeen, johon on sidottu istuntoavain. Istuntoavaimella Handler-komponentti pystyy noutamaan istunnon ja siihen sidotut käyttäjätiedot, joita käytetään kirjautumisissa ja pääsynhallinnassa (10-11). Haettuja käyttäjätietoja verrataan sovellukselle asetettuihin pääsysääntöihin ja suojatulle sovellukselle viedään sovellukselle määritetyt käyttäjätiedot. Tämän jälkeen käyttäjä pääsee suojattuun sovellukseen (12-14). Handler-komponentti tarkistaa evästeen jokaisella HTTP-pyynnöllä, joten järjestelmällä voidaan myös estää asiattomilta pääsy sovelluksiin. Käyttäjien onnistuneet ja epäonnistuneet todennukset sekä luodut istunnot voidaan myös lokittaa Syslog-viestien avulla keskitetylle lokipalvelimelle. (LemonLDAP – Presentation N.d.)

Alaluvussa 2.2 läpikäytyjen mallien avulla voidaan selittää LemonLDAP:n luomaa SSO-kokonaisuutta. Itse Portal-komponentti toimii käytännössä keskitettynä tunnistuspalvelimena, eli Brokerina. Handler-komponentti sen sijaan toimii käytännössä Proxy-mallin mukaisesti ja on vastuussa käyttäjän pyyntöjen filtteröinnistä ja uudelleenohjauksista tarvittaessa. Sovelluksesta riippuen on mahdollisesti tarve lisätä myös agentti-komponentti, kuten Kerberos- tai CAS-moduuli. Myös ohjelmistopohjaisten tokenien käyttö on mahdollista OpenID Connectin kautta.

6.2 Käyttäjän todennus

LemonLDAP tukee lukuisia eri käyttäjäkantoja. Eroavaisuuksia käyttäjänhallintaan muodostuu lähinnä eri kantojen ominaisuuksien vuoksi. Esimerkiksi Active Directory –käyttäjien salasanoja ei voida hallinnoida LemonLDAP:ssa, sillä salasanakäytänteet ovat hyvin käyttäjäkohtaisia ja jokaisella käyttäjällä voisi teoriassa olla eri salasanan voimassaoloaika. Käyttäjiä taas ei voida hallinnoida, jos käyttäjän tunnistus tapahtuu

Radius-palvelinta vasten, sillä käyttäjät voidaan tuoda Radiukselle jostain muusta järjestelmästä. Taulukossa 1 esitetään LemonLDAP:n tukemat käyttäjäkannat. Taulukosta on jätetty pois oletuksena päällä oleva Demonstration-kanta, joka koostuu tekstitiedostosta. Lisäksi on mahdollista käyttää esimerkiksi Null-kantaa, jolla voidaan ohittaa käyttäjän tunnistus testaustilanteessa. (LemonLDAP – Start N.d.)

Taulukko 1. LemonLDAP:n käyttäjäkannat (LemonLDAP – Start N.d.)

Käyttäjäkanta	Todennus	Käyttäjät	Salasanat
Active Directory	✓	✓	
Apache	✓		
BrowserID	✓		
CAS	✓		
Databases	✓	✓	✓
Facebook	✓	✓	
LDAP	✓	✓	✓
Radius	✓		
SAML 2.0 / Shibboleth	✓	✓	
Twitter	✓		
OpenID Connect	✓	✓	

Mahdollisuus käyttää eri käyttäjäkantoja palvelee JYVSECTEC:n harjoitustoimintaa. SSO-järjestelmää voidaan tarjota harjoitusympäristöihin, joissa ei ole käyttäjähakemistoa, kuten Active Directory tai OpenLDAP. Lisäksi esimerkiksi Radiuksen käyttäminen mahdollistaa kaksivaiheisen todennuksen käyttöönoton, kuten myöhemmin tässä työssä tullaan havaitsemaan.

6.3 Konfiguraatio- ja istuntokanta

LemonLDAP vaatii Manager-komponentin hallitsemalle konfiguraatiolle tallennussijainnin. Tallennustapaa valittaessa tulee ottaa huomioon, onko etäkomponenteille tarvetta. Kaikki tallennustavat eivät ole automaattisesti jaettavissa verkon yli. Toinen huomionarvoinen asia on, että LDAP-tuki on vain OpenLDAP-toteutuksissa, sillä Microsoft ei tue tarvittavan pitkiä attribuutteja omassa hakemistopalvelussaan. (LemonLDAP – AD 2016.)

Konfiguraatio voidaan esittää palvelulle käyttäen JSON-muotoiltua konfiguraatiotiedostoa tai eri tietokantoja. Tuettuja tietokantoja ovat esimerkiksi MySQL ja MongoDB. Tarvittaessa konfiguraatio voidaan jakaa myös SOAP-rajapinnan (Simple Object Access Protocol) kautta. Tällöin SOAP-rajapintaa toimii eräänlaisena välityspalvelimena etäkomponenttien ja konfiguraation sisältävän levyjärjestelmän välillä. Tekstitiedostomuotoa käyttäessä voidaan myös hyödyntää mahdollisesti verkossa olevia NFS-jakoja (Network File System), jolloin SOAP-rajapintaa ei tarvita. (LemonLDAP – Start N.d.)

Istuntokantana voidaan niin ikään käyttää paikallista tiedostoa. Muita vaihtoehtoja ovat eri tietokannat kuten MySQL, MongoDB ja Redis. Myös istuntotiedot voidaan kirjoittaa LDAP-hakemistoon OpenLDAP:a käyttävissä toteutuksissa. Paikallinen istuntokanta voidaan konfiguraation tapaan jakaa SOAP-rajapinnan yli. (LemonLDAP – Start N.d.)

Työssä käytettiin aluksi paikallisia konfiguraatio- ja istuntokantoja, jotka jaettiin SOAP-rajapinnan yli etäkomponenteille. SOAP-rajapinnan käyttö HTTPS-yhteyden (Hypertext Transfer Protocol Secure) yli tuotti hieman vaikeuksia, sillä salauksen mahdollistavat Perl-moduulit puuttuivat riippuvuuslistoista ja nämä piti käsin etsiä ja pakottaa toimimaan itse allekirjoitettujen sertifikaattien kanssa. Lopulta SOAP-toteutuksesta luovuttiin, sillä etäkomponenttien istuntomuutokset eivät tallentuneet SOAP:n päätepistettä käyttäessä ja lopullinen toteutus käyttää MySQL-tietokantaa.

6.4 Single sign-on –tuki

LemonLDAP tukee riittävän monipuolisesti sekä yleisiä SSO- ja federointitekniikoita että omia, integrointia helpottavia tekniikoitaan. Tuettuna on esimerkiksi SAML 2.0, jossa LemonLDAP voi toimia IdP:n tai palveluntarjoajan roolissa. CAS-protokolla ja –palvelu ovat tuettuja versiosta 1.0 asti ja lisäksi LemonLDAP:n uusin versio, 1.9, tarjoaa myös OpenID Connect 1.0 –palvelun. Tarvittaessa LemonLDAP voi toimia pelkästään OpenID Connect -välityspalvelimena ja tällöin on mahdollista käyttää kolmannen osapuolen palveluntarjoajaa, jonka palvelussa käyttäjä varsinaisesti tunnistetaan. Huomioitavaa on, että kaikki yllä mainitut palvelut voivat olla yhtä aikaa käytössä ja lisäksi ne voivat tarvittaessa toimia protokollatulkkeina toisilleen. (LemonLDAP – Documentation N.d.)

LemonLDAP:n omat tekniikat, kuten HTTP-otsikoiden käyttö integroitavien palveluiden omien HTTP-todennustoteutusten hyödyntämiseksi, helpottavat muuten hankalaa SSO-toteutusta ja lisäksi on mahdollista käyttää hyväksi LemonLDAP:n automaattista lomakkeentäyttöä. Automaattinen lomakkeentäyttö, Form Replay, mahdollistaa myös sellaisten palveluiden integroinnin järjestelmään, joihin ei SSO-kirjautumista ole mahdollistettu muulla tavalla. Työssä pyrittiin hyödyntämään ja testaamaan mahdollisimman monta tekniikkaa. Taulukossa 2 on esitetty yhteenveto käytettävissä olevista SSO-tekniikoista. (LemonLDAP – Documentation N.d.)

Taulukko 2. LemonLDAP:n SSO-tekniikat (LemonLDAP – Documentation N.d.)

Protokolla/tekniikka	**Versio**	**Toimintamalli**
SAML	2.0	IdP, Service Provider
CAS	1.0, 2.0, 3.0	Server
OpenID Connect	1.0	Provider, Relaying Party
HTTP-otsikot	-	HTTP-todennukseen
Form Replay	-	Lomakkeen automaattinen täyttö

6.5 Perustelut valinnalle

LemonLDAP:n toimintafilosofia, tuki usealle SSO-tekniikalle ja ennen kaikkea tuki monelle eri käyttäjäkannalle olivat lopulta ratkaisevat tekijät SSO-komponentin valinnassa. Useat organisaatioympäristössä käytössä olevista palveluista tukivat suoraan HTTP-todennusta jollain tasolla ja tämä yksinkertaisti toteutussuunnitelmaa. Lisäksi LemonLDAP:sta löytyvä Form Replay –toiminne mahdollistaa myös sellaisten sovellusten integroinnin, joihin ei SSO-ratkaisua ole toteutettu. Perl asettaa rajoituksen Windows-pohjaisten palveluiden suoralle integroinnille, mutta toteutus pysyy kuitenkin melko siistinä, sillä Windows-palvelimet voidaan sijoittaa saman välityspalvelimen taakse, sillä Handler-komponentit ovat Virtual Host –kohtaisia. Tämä tarkoittaa, että pyyntö välityspalvelimen suojaamaan sovellukseen osuu aina oikeaan palveluun, kunhan huolehditaan, että välityspalvelimena toimiva web-palvelin käyttää nimiin pohjautuvia Virtual Hosteja. Tällöin pyyntö kohdistetaan sellaiseen Virtual Hostiin, jonka ServerName vastaa pyydettyä host-nimeä.

Mahdollisuus tuoda käyttäjät esimerkiksi tietokannasta taas tarkoittaa, että SSO-komponentti voidaan toteuttaa tarvittaessa myös ympäristöön, jossa ei ole LDAP-palvelinta. Samasta syystä Kerberosta ei ole työssä käytetty kuin pakon edestä, sillä KDC-palvelinta ei myöskään löydy jokaisesta ympäristöstä. Jos taas toteutuksessa olisi keskitytty yksinomaan esimerkiksi OAuth-protokollan käyttämiseen, olisi salasanaa vaativien sovellusten integroinnissa tullut vaikeuksia. Esimerkiksi web-pohjaisena sähköpostipalveluna toimiva Roundcube todentaa käyttäjän sähköpostipalvelimen IMAP-palvelua vastaan, eikä suoraan haluttua OpenID Connect Provideria vastaan. Tämä tarkoittaa, että vaikka Roundcubemail itse saisi käyttäjän identiteetille vahvistuksen, vaatisi IMAP-palveluna toimiva Dovecot silti salasanan käyttäjältä.

Myös LemonLDAP:n referenssit vaikuttivat ja järjestelmän tulisi olla erittäin suorituskykyinen, sillä esimerkiksi Ranskan kansallinen poliisi käyttää SSO-toteutuksessaan LemonLDAP:a ja järjestelmään kohdistuu jopa 25 miljoonaa pyyntöä päivittäin (LemonLDAP – References). Ohjelmiston kehitys on myös jatkunut jo pitkään ja lähitulevaisuuden kehityksen piirissä on esimerkiksi OATH-tuki, jolloin järjestelmään saa suoraan tuen kaksivaiheiselle todennukselle (LemonLDAP – OATH 2016).

7 Testiympäristö

7.1 Yleistä

Testiympäristössä pyrittiin mallintamaan RGCE-organisaatioympäristöjä käyttäen vastaavaa jaottelua verkkoihin ja keskittyen vain yleisen organisaatiomallin palveluihin. Virtualisointialustana palveluille toimivat Oracle VirtualBox ja VMWare Workstation 12, joista ensimmäistä käytettiin toteutettaessa SSO-järjestelmää ja jälkimmäistä VPN-palvelun kaksivaiheisen todennuksen toteutukseen. Liitteessä 2 esitetään VirtualBoxissa käytetty ympäristö palveluineen ja osoitteistuksineen.

Testiympäristö oli jaettu kolmeen sisäiseen verkkoon. Hallintaverkossa eli liitteen 2 WS/MGMT-verkossa toimi lokipalvelin ja valvontaohjelmisto sekä itse SSO-komponentti. Organisaatiomallin mukaisesti lokipalvelin käsittää kokonaisuutena palvelut Logstash, Elasticsearch ja Kibana. Valvontapalvelimella itse palvelusta vastaa Paesslerin PRTG.

Sisäisten palvelujen verkossa eli liitteen 2 SERVERS-verkossa toimi ympäristön käyttäjäkanta- ja hakemistopalveluna Active Directory Domain Services. Ympäristön aikasynkronoinnista vastasi yksinkertainen NTP-palvelin. Ympäristössä tarjotaan intrapalvelu, jota mallinnetaan Wordpress-sivustolla. Lisäksi toteutuksessa testattiin myös Active Directory Federation Services –palvelua käyttäen Windows Server 2016 –käyttöjärjestelmän testiversiota.

Julkisten palveluiden verkkosegmentissä eli liitteen 2 DMZ-verkossa toimi yrityksen verkosta ulospäin näkyviä palveluja, kuten sähköposti sisältäen webmailin ja yrityksen palvelupiste. Sähköposti toteutettiin käyttäen RGCE-mallin mukaisesti palveluina Postfix, Dovecot ja Roundcubemail –yhdistelmää. Palvelupisteenä toimi Open-source Ticket Request System, eli OTRS.

Yhteistä palveluille valvontaa lukuun ottamatta on niiden Open Source –pohja ja käyttöjärjestelmänä toimiva Linux. SSO-komponenttia lukuun ottamatta testiympäristön Linux-jakelupakettina käytettiin CentOS 6:sta. Testiympäristön toimialueena käytetään nimeä example.com.

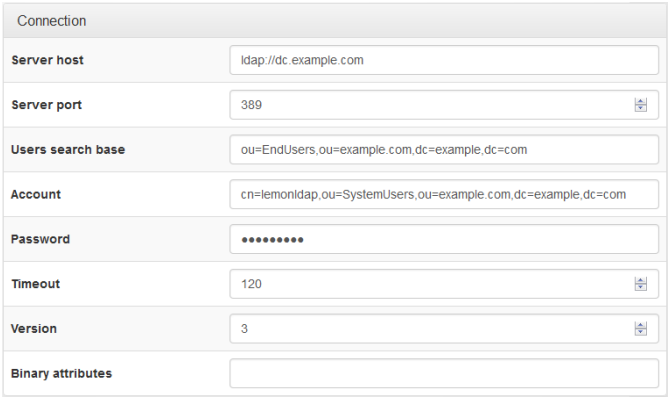
7.2 LemonLDAP

Single sign-on –komponentti konfiguroitiin ympäristöön käsin. Sen jälkeen, kun järjestelmä oli todettu toimivaksi, komponentista tehtiin mallipohja, johon lisättiin asennusta helpottavat skriptit ja tehtiin valmiiksi palomuurisäännöt, SELinux-säännöt ja asennettiin tarvittavat paketit. Järjestelmä toimii yhdellä palvelimella, jonka käyttöjärjestelmänä toimii CentOS 7. Lisäksi järjestelmän integrointia eri organisaatioympäristöihin helpotettiin luomalla RCGE-ympäristöön oma Linux-repository LemonLDAP:n tarvitsemia paketteja varten. SSO-komponentin asennusskripti esitetään liitteessä 3.

Skriptin avulla on mahdollista määrittää järjestelmä käyttämään joko paikallista tai toisella palvelimella sijaitsevaa tietokantaa. Lisäksi skriptiin on toteutettu jonkin verran käyttäjäsyötteen testausta mahdollisten ongelmatilanteiden välttämiseksi. Esimerkkinä tästä on syötetyn IP-osoitteen tarkistaminen ja validoiminen. Asennus keskeytetään, jos IP-osoite ei ole muodoltaan oikea tai osoittaa localhost-osoitteeseen, sillä Handler-komponentit eivät voi ottaa yhteyttä toisen palvelimen localhostiin. Käyttäen hyväksi LemonLDAP:n komentorivityökalua, saadaan asennusskriptillä luotua Active Directory –integrointi, salasanan sitominen istuntoon ja konfiguraatio- sekä istuntokantojen käyttöönotto.

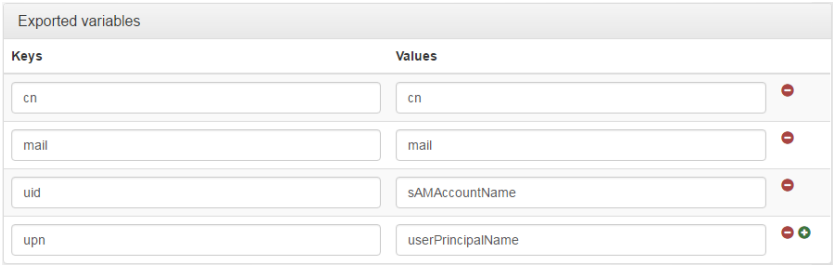
Ympäristön muutoksiin voidaan reagoida myös vaihtamalla tietokantapalvelin paikallisesta etäpalvelimeen ja toisin päin. Näihin muutoksiin on myös luotu omat skriptinsä, jotka käytännössä vain luovat uuden asetustiedoston ja siirtävät konfiguraatio- ja istuntokannat halutulle palvelimelle vastaavalla tavalla kuin itse asennusskriptissäkin. Näin ympäristössä toimivat henkilöt voivat itse tehdä päätöksiä järjestelmään liittyen.

Järjestelmä toteutetaan tuomalla käyttäjäkanta Active Directorysta. Yhteys vaatii Active Directoryyn käyttäjätilin, jolla on lukuoikeus hakemistorakenteeseen. Mikä tahansa käyttäjätili käytännössä kävisi tähän, mutta hyödyllisempää on tehdä oma käyttäjätili palvelulle, jolloin palvelun toiminta on sidottavissa helposti löydettävään tiliin sen sijaan, että muun muassa lokiviestit hukkuisivat jonkin sisäänrakennetun admin-tilin tapahtumiin. Seuraavalla sivulla, kuviossa 11, on esitetty tehdyt Active Directory –integroinnin mahdollistavat muutokset hallintasovellukseen.



Kuvio 11. LemonLDAP:n Active Directory –integrointi

Samalla määritellään, mitä attribuutteja hakemistosta tuodaan LemonLDAP:n käytettäväksi. Avain (Key) on tämän jälkeen käytettävissä, kun halutaan viedä suojatuille sovelluksille käyttäjätietoja ja avaimen arvo määritetään hakemistosta noudetun attribuutin arvolla. Esimerkiksi avain ”uid” voidaan lähettää suojatulle sovellukselle muuttujana ”$uid” ja arvo muuttujalle on hakemistosta noudettu käyttäjän ”sAMAccountName”, eli käyttäjätunnus. Kuviossa 12 esitetään noudetut attribuutit.



Kuvio 12. Active Directorysta noudettavat LDAP-attribuutit

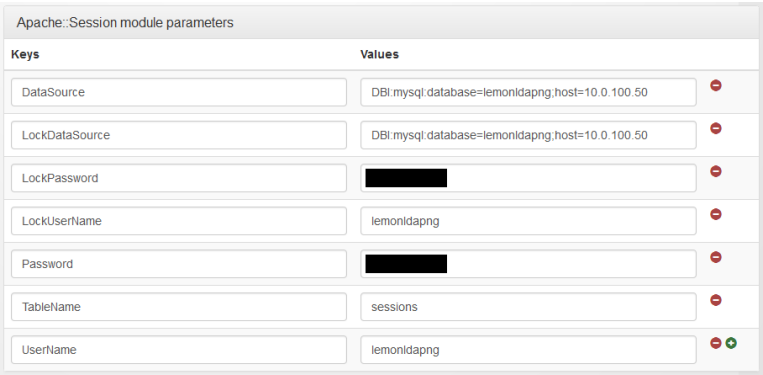
Pitkällisen testauksen jälkeen todettiin, että konfiguraation jakaminen tietokannan kautta toimii ympäristössä parhaiten. Konfiguraation jakaminen SOAP-rajapinnan yli on myös mahdollista, mutta tuntemattomasta syystä istuntoja ei onnistuttu sitomaan Handler-komponentteihin.

Tietokantapalvelimella konfiguraatio- ja istuntotietojen jakaminen onnistui kuitenkin helposti. Hallintasovelluksen kautta muutoksia tehtäessä tulee kuitenkin olla tark-

kana, että asetukset ovat kerralla oikein ennen konfiguraation tallentamista. Asennusskripteillä näistä vaikeuksista päästiin kuitenkin eroon, sillä muutoksia ei tarvitse tehdä hallintasovelluksen kautta.

Erheen sattuessa LemonLDAP tarjoaa kaksi korjausvaihtoehtoa, komentorivityökalun ja vi-pohjaisen tekstieditorityökalun konfiguraatiolle. Komentorivityökalu on omiaan automatisoidessa toimintoja, mutta tekstieditorityökalu konfiguraation editoimiselle toimii huomattavasti luotettavammin, sillä etukäteen ei tarvitse tietää, mikä avain tai avaimen arvo on kyseessä, vaan kaikki konfiguraatio on kerralla nähtävissä ja muokattavissa.

Huono puoli konfiguraation jakamisessa tietokannan kautta on tietokannan itsensä rakenne. LemonLDAP:n asetukset ovat JSON-muodossa, joten jokainen avain ja avaimen arvo ovat oma rivinsä tietokannassa. Yksi konfiguraatiotiedosto tuottaa jo monta sataa riviä tietokantaan. Testiympäristössä käytetty SSO-komponentti sisälsi lopulta noin 160 konfiguraatioversiota, joista muodostuu jo yli kuusikymmentätuhatta riviä tietokantaan. Kuviossa 13 esitetään tarvittavat hallintasovelluksen muutokset tietokannan käyttöönottamiseksi.



Kuvio 13. LemonLDAP:n MySQL-istuntoasetukset

Itse konfiguraatiotiedoston muoto ja sijainti määritetään järjestelmän yleisissä asetuksissa. Muokataan tiedostoon /etc/lemonldap-ng/lemonldap-ng.ini seuraavan sivun kuvion 14 esittämät tiedot.

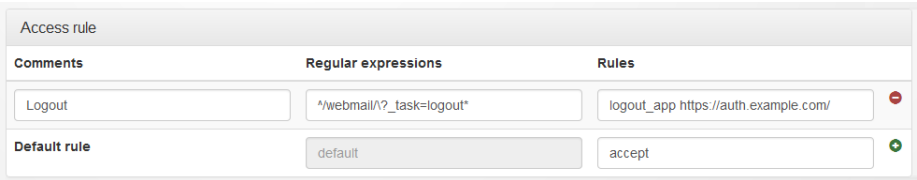
```
* RDBI/CDBI : you have to set 'dbiChain' (required) and 'dbiUser' and 'dbiPassword'
        if needed. Example:

          type        = RDBI
          dbiChain    = DBI:mysql:database=lemonldapng;host=10.0.100.50
          dbiUser     = lemonldapng
          dbiPassword = [redacted]
```

Kuvio 14. LemonLDAP:n MySQL-konfiguraatioasetukset

Konfiguraatiolle määritellään näin tietokannan nimi ja tietokannan sijainti. Lisäksi asetuksiin määritetään tietokannan käyttäjä ja käyttäjän salasana. Jos lemonldap-ng.ini-tiedostoon ei tehdä muita yliajavia muutoksia, on tiedosto helppo kopioida suoraan etäkomponenteille.

7.3 Roundcubemail

Organisaatioympäristöjen sähköpostipalvelimessa selainpohjaisena sähköpostiohjelmistona käytetään Roundcubemailia. Tämä on käytetyistä palveluista ainoa, johon LemonLDAP tarjoaa suoraan toimivan ratkaisun kertakirjautumisen toteuttamiseen. Single sign-on toteutetaan käyttämällä hyväksi Roundcubemailin omaa HTTP-todennuksen mahdollistavaa liitännäistä. Lisäksi sähköpostipalvelimelle asennetaan LemonLDAP:n Handler-komponentti, joka on vastuussa HTTP-pyyntöjen kiinniottamisesta ja tarvittavien HTTP-otsikoiden välittämisestä. Ensimmäiseksi lisätään LemonLDAP-hallintasovelluksella uusi Virtual Host, mail.example.com. Virtual Hostille määritetään pääsysäännöt kuvion 15 mukaisesti.

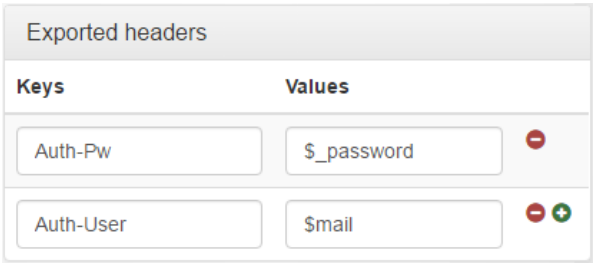


Kuvio 15. Mail.example.com Virtual Hostin pääsysäännöt

Sähköpostipalvelimen Handler-komponentti tarkistaa jokaisen sivustolle saapuvan HTTP-pyynnön. Jos HTTP-pyynnön GET-parametri on _task=logout, LemonLDAP:n

Handler-komponentti huomaa tämän ja purkaa paikallisen istunnon. Uloskirjautumissäännöllä voidaan määritellä käyttäjälle uudelleenohjaus esimerkiksi Portal-komponenttiin tai purkaa koko SSO-istunto. Oletussääntö ”Accept” määrittää, että kaikki onnistuneesti todennetut käyttäjät päästetään sovellukseen.

Koska Virtual Host on TLS-suojattu, määritetään asetuksiin pakotus portille 443, jonka yli HTTPS yhdistyy. Itse kertakirjautumisen mahdollistaa lähetettävät HTTP-otsikot, jotka määritetään kuvion 16 mukaisesti.



Kuvio 16. Mail.example.com-sivustolle lähetetyt HTTP-otsikot

Salasana käyttäjälle poimitaan Portaaliin kirjautumisesta, sillä AD-käyttäjän salasana on tallennettu hakemistorakenteeseen vain tiivistesummana ja täten kyseistä attribuuttia ei voida noutaa käyttöön suoraan Active Directorysta. Auth-User nimellä lähetettävä käyttäjänimi taas on sidottava sähköpostipalvelimen käyttämään käyttäjänimeen, joka tässä tapauksessa on Active Directory –käyttäjän sähköposti-kentän arvo.

Roundcubemailin Virtual Host -tiedostoon määritellään uudelleenohjaus HTTP-yhteyksistä HTTPS-yhteyteen ja Handler-komponentti asetetaan kuuntelemaan *:443 Virtual Hostia. Lisätään tiedostoon /etc/httpd/conf.d/zzz-mail.conf rivit:

```
<VirtualHost *:80>
   ServerName mail.example.com
   RedirectMatch ^/$ /webmail/
   RewriteEngine  on
   RewriteCond    %{HTTPS} !=on
   RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
</VirtualHost>
<VirtualHost mail.example.com:443>
   ServerName mail.example.com
   SSLEngine on
   SSLCertificateFile /etc/ssl/certs/mail.crt
```

```
   SSLCertificateKeyFile /etc/ssl/certs/mail.key
   Alias /webmail /usr/share/roundcubemail
   PerlHeaderParserHandler Lemonldap::NG::Handler
   *Roundcuben omat konfiguraatiot*
</VirtualHost>
```

Lisäksi LemonLDAP:n konfiguraation pitämiseksi ajantasaisena luodaan toinen Virtual Host –tiedosto. Lisätään tiedostoon /etc/httpd/conf.d/00-handler.conf rivit:

```
NameVirtualHost "*:80"
PerlOptions +GlobalRequest
PerlModule Lemonldap::NG::Handler
ErrorDocument 403 http://auth.example.com/?lmError=403
ErrorDocument 500 http://auth.example.com/?lmError=500
ErrorDocument 503 http://auth.example.com/?lmError=503
<VirtualHost "*:80">
   ServerName reload.example.com
   <Location /reload>
      Require ip 10.99.0.250 127 ::1
      PerlHeaderParserHandler Lemonldap::NG::Handler->reload
   </Location>
</VirtualHost>
```

Näin mahdollistetaan Handler-komponentin välimuistin päivittäminen etänä LemonLDAP:n Manager-hallintasovelluksesta. Lisäksi päivittämisen voi tehdä käsin myös paikallisesti, sillä localhost-osoitteet ovat sallittujen listalla.

Itse HTTP-todennus toteutetaan valmiilla liitännäisellä, joka otetaan käyttöön lisäämällä /etc/roundcubemail/confic.inc.php konfiguraatiotiedostoon rivi:

```
$config['plugins'] = array('http_authentication');
```

Itse liitännäiseen tarvitsee myös tehdä muutoksia, sillä liitännäinen etsii PHP_AUTH_*-nimisiä muuttujia, ja tarkoitus on käyttää HTTP_AUTH_*-muuttujia. Tämä onnistuu helposti vaihtamalla kaikki esiintyvät PHP_AUTH_*-muuttujat HTTP_AUTH_*-muuttujiin vapaavalintaisen tekstieditorin korvaustoiminnolla tiedostosta /usr/share/roundcubemail/plugins/http_authentication/http_authentication.php.

Tämän jälkeen pyyntö mail.example.com osoitteella jää kiinni Virtual Hostissa määritettyyn Handler-komponenttiin, joka tarkistaa onko pyyntöön lisätty LemonLDAP:n eväste. Jos ei ole, käyttäjä ohjataan Portaaliin ja kirjautumisen jälkeen käyttäjä ohjataan takaisin alkuperäisen resurssipyynnön osoitteeseen, jossa Handler-komponentti luo määritetyt HTTP-otsikot ja kirjautuminen onnistuu automaattisesti. Jos eväste löytyy alkuperäisestä pyynnöstä, niin käyttäjä kirjataan suoraan omalle postitililleen.

7.4 OTRS

Ympäristön helpdesk-sovelluksena toimii vastaavien organisaatioympäristöjen tapaan OTRS. OTRS on kirjoitettu Perl-ohjelmointikieltä käyttäen, kuten myös LemonLDAP:n Handler-komponenttikin, ja tämä aiheuttaa ristiriitoja. Käytetyt Perl-moduulit eivät lataudu käyttöön oikein Handler-komponenttia kutsuttaessa. Tästä johtuen kertakirjautuminen toteutettiin käyttämällä pelkästään Kerberosta ja sovelluksen omaa HTTP Basic –todennusta.

Palvelu täytyy sitoa AD-ympäristöön (Active Directory) palvelukohtaisella päänimellä (Service Principal Name), jotta Kerberos-tikettien pyyntö onnistuu. Service Principal Name luodaan ympäristön ohjainpalvelimella käyttämällä Powershell-komentotulkin työkaluja. Lisäksi palvelu halutaan sitoa käyttäjään, jotta useampi yhtäaikainen Kerberos-tiketti voidaan neuvotella käyttöön. Lisätään siis käyttäjätili kerberos_otrs ja tämän jälkeen luodaan elevoidulla komentorivillä SPN:t käyttäen komentoja:

```
setspn -A host/helpdesk.example.com kerberos_otrs
setspn -A HTTP/helpdesk.example.com kerberos_otrs
```

SPN asetetaan sekä isäntäpalvelimelle itse että sen tarjoamalle palvelulle. Seuraavaksi sidotaan luodut päänimet käyttäjään komennoilla:

```
ktpass -princ host/helpdesk.example.com@EXAMPLE.COM -pass
SALASANA -mapuser kerberos_otrs -pType KRB5_NT_PRINCIPAL -out
c:\temp\otrs.host.keytab
ktpass -princ HTTP/helpdesk.example.com@EXAMPLE.COM -pass
SALASANA -mapuser kerberos_otrs -pType KRB5_NT_PRINCIPAL -out
c:\temp\otrs.HTTP.keytab
```

Nimi esitetään KerberosV5-protokollan mukaisesti ja sidotaan äsken luotuun käyttäjään. Oletuksena keytab-tiedostot suojataan varsin heikolla salausalgoritmilla, joten

lisäksi kannattaa käyttää parametria –crypto ja määrittää vahvempi algoritmi, kuten esimerkiksi AES256-SHA1. Tehdyt keytab-tiedostot viedään helpdesk-palvelimelle ja sidotaan yhteen tiedostoon. Linux-palvelimelle täytyy lisäksi asentaa seuraavat paketit: krb5-workstation, mod_auth_kerb, ja jos halutaan suorittaa pääsynhallintaa, niin lisäksi asennetaan mod_auth_ldapz. Krb5-workstation –paketti tuo tarvittavat kirjastot Kerberos-yhteyksien luomiseksi ja mod_auth_kerb tuo Apache-webpalvelimeen tuen Kerberos-protokollalle. Pääsynhallintaa voidaan tehdä Apachen LDAP-moduulilla, sillä sen avulla voidaan määrittää esimerkiksi tietty LDAP-tili tai –ryhmä vaatimukseksi onnistuneen käyttäjän tunnistuksen lisäksi.

Kerberos-toimialue (Kerberos Realm) täytyy myös määrittää palvelimelle tiedostossa /etc/krb5.conf. Tiedostoon lisätään seuraavat rivit:

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log
[libdefaults]
    default_realm = EXAMPLE.COM
    v4_instance_resolve = false
    v4_name_convert = {
        host = {
            rcmd = host
            ftp = ftp
        }
        plain = {
            something = something-else
        }
    }
    fcc-mit-ticketflags = true
[realms]
    EXAMPLE.COM = {
        kdc = dc.example.com
        admin_server = dc.example.com
        default_domain = example.com
    }
[domain_realm]
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM
[login]
    krb4_convert = true
    krb4_get_tickets = false
```

Krb5.conf on rakennettu tukemaan myös Heimdal-toteutusta Kerberoksesta ja sisältää vielä tuen Kerberoksen neljännelle versiolle. Kerberos TGT-tiketin pyytäminen onnistuu nyt komennolla:

kinit -kt /etc/httpd/HTTP.keytab HTTP/helpdesk.example.com

Tehdyt muutokset ja tikettipyynnön onnistuminen voidaan todentaa kuvion 17 osoittamalla tavalla.

```
[root@helpdesk ~]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: HTTP/helpdesk.example.com@EXAMPLE.COM

Valid starting     Expires            Service principal
10/11/16 09:39:46  10/11/16 19:39:49  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 10/12/16 09:39:46
```

Kuvio 17. TGT-pyyntö helpdesk-palvelimelta

Kun Kerberos-tikettien pyyntö onnistuu, voidaan toteuttaa OTRS:n Active Directory –integrointi, jotta saadaan käyttäjien kirjautuminen onnistumaan. Oletuksena sovelluksella on vain yksi superuser-käyttäjä, joten käyttäjäkanta täytyy noutaa ohjainpalvelimelta. Integraatio tehdään lisäämällä liitteen 4 sisältö tiedostoon /opt/otrs/Kernel/Config.pm. Tiedoilla määritellään tunnistusmoduulit ja niiden järjestys, sekä agentti- eli admin-käyttäjien tietojen synkronointi ohjainpalvelimelta OTRS:n tietokantaan. Huomioitavaa on, että HTTPBasicAuth-moduulin täytyy olla ensisijainen, jotta kertakirjautuminen on mahdollista. OTRS käy määritetyt tunnistusmoduulit järjestyksessä läpi, kunnes käyttäjän tunnistus onnistuu. Kertakirjautumista ei siis yritetä, jos moduulin edelle on määritetty toinen tunnistusmoduuli.

Käytössä olevat selaimet eivät myöskään oletuksena osaa jakaa käyttäjän tietoja web-sovellukselle, joten selaimiin täytyy myös tehdä muutoksia. Mozilla Firefoxin asetukset saadaan auki kirjoittamalla osoiteriville about:config ja käyttäjätietojen jako sovelluksille määritetään etsimällä kohdat:

network.negotiate-auth.trusted-uris
network.negotiate-auth.delegation-uris

Nämä kentät määrittävät sivustot, joihin luotetaan, ja joille käyttäjän käyttäjätiedot voidaan jakaa. Kenttiin käy arvoksi tarkka sivuston osoite tai esimerkiksi korvausmerkki, jolla saadaan koko toimialue luotetuksi. Työssä käytettiin kenttään korvausmerkkiä *.example.com, sillä helpdesk.example.com ei ole ainoa tarvittava URI. Internet Explorer ja Google Chrome toimivat samoilla asetuksilla, sillä Chrome ottaa kyseiset tietoturva-asetukset Internet-asetuksista Explorerin tavoin. Muokataan Internet-asetuksia Explorerin kautta:

Internet Options -> Security -> Trusted sites -> Automatic login using current username and password -> add site

Tähän voidaan Mozillan tavoin käyttää koko toimialueen sallivaa arvoa, joka on *.example.com. Vielä täytyy määrittää Apachen konfiguraatiotiedostoon tarvittavat muutokset, jotta käytössä olevat moduulit pyydystävät pyynnöt resursseihin ja istunnot pysyvät auki. Muokataan tiedostoon /etc/httpd/conf/httpd.conf rivit:

```
KeepAlive On
UseCanonicalName On
```

Lisätään tiedostoon /etc/httpd/conf.d/zzz-otrs.conf rivit käyttäen pohjana liitteen 5 konfiguraatioita:

```
<VirtualHost *:80>
   ServerName helpdesk.example.com
   ScriptAlias /otrs/ "/opt/otrs/bin/cgi-bin/"
   Alias /otrs-web/ "/opt/otrs/var/httpd/htdocs/"
   Alias /agent/ "/opt/otrs/var/httpd/htdocs/"
   RedirectMatch ^/agent$ https://helpdesk.example.com/otrs/index.pl
   RedirectMatch ^/agent/$ https://helpdesk.example.com/otrs/index.pl
   RedirectMatch ^/$ https://helpdesk.example.com/otrs/customer.pl
   RedirectMatch  ^(?!/(otrs|otrs-web)/(js|skins|index|customer|faq|json|rpc)) https://helpdesk.example.com/otrs/customer.pl
</VirtualHost>
<VirtualHost *:443>
   ServerName helpdesk.example.com
   ScriptAlias /otrs/ "/opt/otrs/bin/cgi-bin/"
   Alias /otrs-web/ "/opt/otrs/var/httpd/htdocs/"
   Alias /agent/ "/opt/otrs/var/httpd/htdocs/"
   SSLEngine on
   SSLCertificateFile /etc/ssl/certs/helpdesk.crt
   SSLCertificateKeyFile /etc/ssl/certs/helpdesk.key
```

```
<Directory "/opt/otrs/bin/cgi-bin/">
   AllowOverride None
   Options +ExecCGI -Includes
<Files "index.pl">
# Kerberos SSO
# LDAP auth
</Files>
<Files "customer.pl">
# Customers
   Order deny,allow
   Allow from all
</Files>
```

Jos pyyntö kohdistuu agenttiportaaliin index.pl, niin käyttäjän tunnistus tapahtuu Kerberoksen avulla ja valtuuttamineni LDAP:n avulla. Esimerkkiympäristössä sidottiin kirjautumisen edellytykseksi käyttäjän kuuluminen Active Directoryn Administrators-ryhmään. Itse Kerberos SSO ja LDAP auth –asetukset esitetään liitteessä 5.

Lopuksi integraatiota pyrittiin automatisoimaan mahdollisimman pitkälle, sillä ympäristöstä riippuen esimerkiksi palveluiden sekä palvelimien päänimet ja toimialueen nimet vaihtelevat. Ohjainpalvelimelle luotiin liitteen 6 mukainen Powershell-skripti, jolla saadaan tehtyä palvelun vaatima Active Directory –käyttäjätili, SPN-nimien sitominen käyttäjätiliin ja keytab-tiedostot.

Skripti luo käyttäjän syötetyn nimen perusteella mallin kerberos_NIMI mukaisesti. Lisäksi skriptiin määritetään kerberoitava kohdepalvelin ja palvelun tyyppi, kuten HTTP. KerberosV5 mukaiset SPN-nimet luodaan parsimalla ohjainpalvelimen tiedoista toimialueen nimet. Tällä hetkellä jokaiselle palvelulle luodaan oma käyttäjätilinsä, joten skripti tarkistaa onko syötetty käyttäjätili olemassa ja jos on niin keskeyttää toiminnan ja tulostaa näkyviin olemassa olevan käyttäjän ja mahdolliset sidotut SPN-nimet, jotta ne voidaan halutessa purkaa.

Tämän jälkeen luodaan SPN sekä kohdepalvelimelle, että kohdepalvelulle ja luodaan keytab-tiedostot. Tämän jälkeen keytab-tiedostot voidaan kopioida kohdepalvelimelle. Jos kerberoitu palvelu halutaan poistaa, onnistuu se liitteen 7 skriptillä. Parametreiksi syötetään poistettavan palvelimen ja käyttäjätilin tiedot. Halutessa voidaan poistaa vain SPN-mappaukset ja jättää käyttäjätili ja keytab-tiedostot ennalleen, jos palvelu halutaankin ottaa uudelleen käyttöön kerberoituna.

Kohdepalvelimella prosessi pyritään automatisoimaan Bash-skriptillä. Skriptille annetaan syötteeksi halutun palvelun tyyppi, tässä tapauksessa HTTP, ja polku keytab-tiedostoihin, jotka kopioitiin ohjainpalvelimelta. Lisäksi ennen varsinaista ajoa tarkastetaan, että tarvittavat paketit on varmasti asennettu, jotta vältytään turhilta virheiltä. Itse skripti yhdistää ohjainpalvelimella luodut keytab-tiedostot, jotta palvelin voi käyttää niitä tikettipyynnöissä ja lisäksi täytetään tarvittavat rivit Apachen konfiguraatiotiedostoon ja tehdään mallit joko itse sivuston Virtual Host –tiedostosta tai halutessa voidaan täyttää .htaccess-tiedosto, joka voidaan kopioida suojaamaan haluttua tiedostosijaintia. Bash-skripti on esitetty liitteessä 8.

7.5 Loki- ja flowpalvelimet

Sekä lokipalvelimen että flowpalvelimen web-sovelluksesta vastaa Kibana. Kibana on erittäin helppo integroida SSO-järjestelmään, sillä se ei varsinaisesti sisällä minkäänlaista käyttäjän tunnistusta. Käyttäjän todennuksesta vastaa Kibanan edessä oleva web-palvelin, joka testiympäristössä on Apache. Myös Nginx on tuettu käytetyssä versiossa LemonLDAP:sta. Yksinkertaisimmillaan siis riittää, että asennetaan palvelimelle Handler-komponentti käsittelemään pyyntöjä resurssiin. Apache toimii välityspalvelimena Kibanalle ja suojattava sijainti määritetään Virtual Host –tiedostoon /etc/httpd/conf.d/zzz-kibana.conf vastaavalla tavalla:

```
<VirtualHost *:80>
   ServerName log.example.com
   RewriteEngine  on
   RewriteCond    %{HTTPS} !=on
   RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
</VirtualHost>
<VirtualHost log.example.com:443>
   ServerName log.example.com
   SSLEngine on
   SSLCertificateFile /etc/ssl/certs/log.crt
   SSLCertificateKeyFile /etc/ssl/certs/log.key
      <Location /app/ >
        PerlHeaderParserHandler Lemonldap::NG::Handler
      </Location>
   *Muut konfiguraatiot*
</VirtualHost>
```

Lisäksi, kuten sähköpostisovellukselle, myös loki- ja flowpalvelimille luodaan vastaava tiedosto /etc/httpd/conf.d/00-handler.conf, joka on sisällöltään identtinen sähköpostipalvelimen tiedoston kanssa. Tämän avulla Handler-komponentin välimuisti voidaan päivittää etänä ja myös paikallisesti, eikä Apache-palvelua tarvitse aina käynnistää uudelleen.

Pääsynhallinta tehdään LemonLDAP:n hallintasovelluksen avulla. Ensiksi luodaan uudet Virtual Hostit: log.example.com ja flow.example.com. Myös näille sovelluksille määritellään oletusportiksi 443, jotta uudelleenohjaukset toimisivat oikein HTTPS-yhteyden yli. Pääsysäännöillä määritellään, miten pääsy resurssiin toteutetaan. Tämä mahdollistaa sen, että eri URI:t voidaan avata kaikille, tai vaikka vain tietylle ryhmälle. Ympäristössä luotiin vain oletussääntö ”accept”, joka hyväksyy kaikki SSO-evästeen sisältävät pyynnöt.

Toisena vaihtoehtona tutkittiin ja testattiin myös OpenID Connect -pohjainen todennus ja valtuuttaminen Apache/Kibana yhdistelmälle, jossa varsinainen käyttäjän tunnistaminen tapahtuu OAuth 2.0 –protokollalla, mutta lopullisesta toteutuksesta tämä jätettiin pois. OpenID Connect –palveluntarjoana käytettiin onnistuneesti sekä LemonLDAP:n omaa OpenID Connect –toteutusta että Microsoft Server 2016 –käyttöjärjestelmän ADFS-palvelun toteutusta.

7.6 Intra

Organisaatioympäristöjen intra-sivusto on Wordpress-pohjainen, kuhunkin ympäristöön erikseen kustomoitava palvelu. Tuki kertakirjautumiselle toteutetaan yhdistämällä LemonLDAP:n Handler-komponentti, Kerberos-todennus ja Wordpressin HTTP-todennus- sekä Active Directory Integration –liitännäiset. Käytännössä yhdistelmässä Handler-komponentti toimii pääsynhallintana, sekä sitoo käyttäjän evästeellä koko järjestelmään. Kerberos-todennus toteutetaan mod_auth_kerb-moduulilla, sillä intran web-palveluna toimii myös Apache. HTTP-todennus poimii olemassa olevasta Kerberos-todennuksesta käyttäjän tiedot ja käyttää niitä kirjautuakseen Wordpressiin ja Active Directory Integration –lisäosa takaa, että ympäristön käyttäjätiedot ovat yhtenevät sekä ohjainpalvelimella että Wordpressin tietokannassa. Palvelua integroidessa pystyttiin käyttämään hyväksi edellä esitettyjä skriptejä ja Kerberos-integrointi sujui nopeasti ja vaivattomasti.

Handler-komponentti asennetaan kuten muillekin palvelimille ja aktivoidaan intran konfiguraatiotiedostossa /etc/httpd/conf.d/zzz-intra.conf seuraavilla riveillä käyttäen pohjana liitteen 5 konfiguraatioesimerkkiä:

```
<VirtualHost *:80>
ServerName intra.example.com
RewriteEngine  on
RewriteCond    %{HTTPS} !=on
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
</VirtualHost>
<VirtualHost *:443>
ServerName intra.example.com
SSLEngine on
SSLCertificateFile /etc/ssl/certs/intra.crt
SSLCertificateKeyFile /etc/ssl/certs/intra.key
<Directory /var/www/html>
  <Files wp-sso.php>
  PerlHeaderParserHandler Lemonldap::NG::Handler
# Kerberos SSO
  </Files>
</Directory>
</VirtualHost>
```

Lisäksi määritetään seuraavat yleiset Apachen asetukset, kuten OTRS:n yhteydessäkin:

```
KeepAlive On
UseCanonicalName On
```

Kyseiset konfiguraatiot saadaan lisättyä automaattisesti edellä esitellyn Bash-skriptin avulla. Apachen Virtual Host –tiedostoon lisättävät rivit voidaan suoraan kopioida skriptin luomasta mallista, ja Apachen konfiguraatiotiedoston muutokset toteutetaan automaattisesti Linuxin omalla sed-työkalulla. Lisäksi, kuten edellä, luodaan myös konfiguraatiovälimuistin päivittämiseen oma Virtual Host –tiedosto /etc/httpd/conf.d/00-handler.conf.

Käyttäjätunnus täytyy poimia sopivassa muodossa HTTP-todennuksen tekevälle liitännäiselle. Oletuksena tämä muoto on UserPrincipalName eli käyttäjätunnus@example.com. Lisäämällä rivi KrbLocalUserMapping Off palvelun Virtual Host –tiedostoon pakotetaan tunnusten poiminta tapahtumaan oikeaan muotoon. Vastaavasti jos kirjautumisessa halutaan käyttää sAMAccountName muotoa eli

EXAMPLE\käyttäjätunnus, niin tämä voidaan toteuttaa korjaamalla itse liitännäisestä rivit:

```
if (! empty($_SERVER[$server_key])) {
        $username = $_SERVER[$server_key];
}
```

Muotoon:

```
if (! empty($_SERVER[$server_key])) {
$userparts = explode('@', $_SERVER[$server_key]);
$username = $userparts[0]
}
```

HTTP-todennus –liitännäinen on tehty ulkopuolista käyttäjäntunnistuspalvelua silmällä pitäen. Liitännäinen lisää Wordpressin kirjautumissivulle painikkeen, jolla käyttäjän voi ohjata omaan tunnistuspalveluunsa. Luodaan Wordpressiin kirjautumispyyntöjä käsittelevästä php-tiedostosta kopio komennolla:

```
cp /var/www/html/wp-login.php /var/www/html/wp-sso.php
```

Luotu kopio voidaan lisätä nyt Login URI –arvoksi liitännäisen asetuksiin Wordpressin Admin-paneelista. Tämä myös mahdollistaa sen, että LemonLDAP:n Handler-komponentti sidotaan vain luotuun kopioon Apachen <files> direktiivillä, joten kirjautumisessa voi hyödyntää Wordpressin omaa kirjautumistoiminnetta tai käyttää kertakirjautumistoimintoa liitännäisen painikkeella. Tämä mahdollistaa esimerkiksi oletuksena luotavan Admin-tilin käyttämisen samanaikaisesti kertakirjautumisjärjestelmän kanssa. Kuviossa 18 esitetään muutokset liitännäisen asetuksiin.

Kuvio 18. Wordpress HTTP-Authentication –asetukset

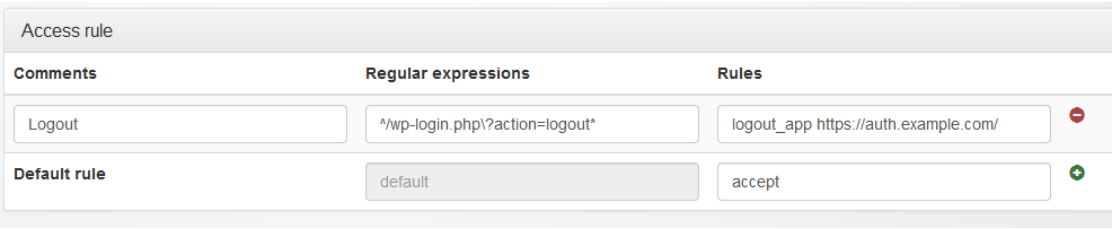
Koska uloskirjaus ei toimi Handler-komponenttiin sidotun php-tiedoston kautta, korjataan uloskirjaus suoraan liitännäisen kirjastoihin. Muokataan ~/http-authentication.php tiedoston rivi:

wp_redirect($logout_uri);

Muotoon:

wp_redirect('https://auth.example.com');

LemonLDAP:n hallintasovellukseen määritetään Virtual Host intra.example.com, jolle määritetään pääsysäännöt ja pakotus HTTPS-yhteyden käyttöön. Jos Handler-komponentti sidotaan Virtual Hostiin muulla tavoin, voidaan käyttää Virtual Hostin omia pääsääntöjä uloskirjautumisen uudelleenohjaukseen. Kuviossa 19 on esitetty tässä tapauksessa luotava sääntö.



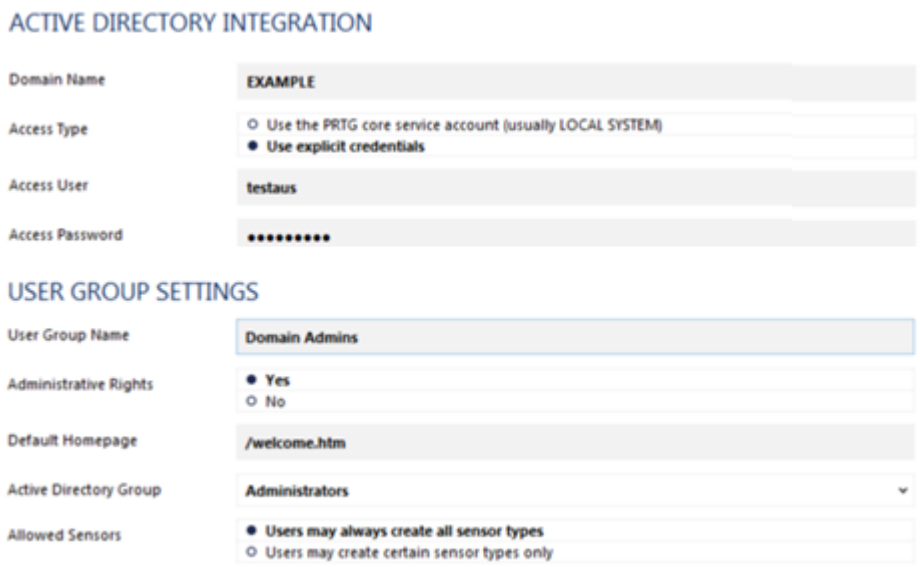
Kuvio 19. Intra-sivuston pääsysäännöt

7.7 Monitorointi

Ympäristön monitoroinnista vastaa Paesslerin PRTG monitorointipalvelu. Muista palveluista poiketen PRTG toimii vain ja ainoastaan Windows-käyttöjärjestelmän päällä. Sovellukseen ei myöskään ole olemasta mitään valmista, tuettua tapaa toteuttaa kertakirjautuminen. Single sign-on palveluun toteutetaan näistä syistä johtuen käyttämällä hyväksi Linux-palvelinta välityspalvelimena PRTG:lle. Välityspalvelin sisältää Apache web-palvelun, johon sidotaan LemonLDAP:n Handler-komponentti ohjaamaan pyyntöjä.

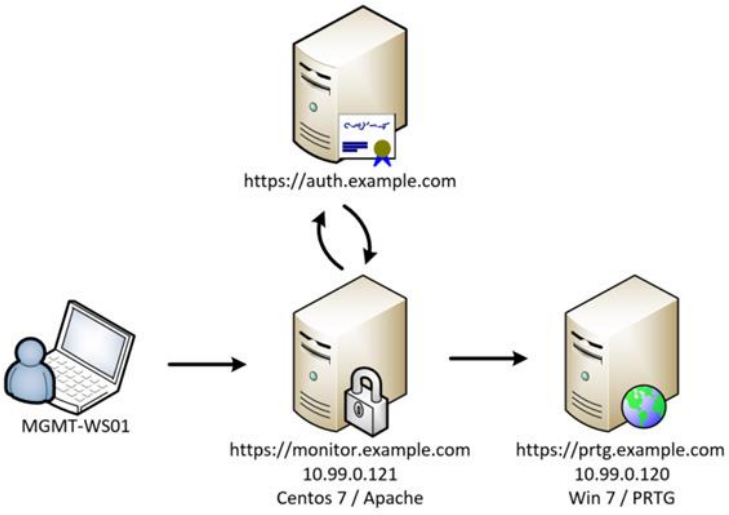
Ensimmäiseksi monitorointisovellus asetetaan hakemaan käyttäjät Active Directorysta. Tämä on hyvin nopea ja helppo toimenpide ja vaatii vain käyttäjätilin Active Directoryyn käyttäjähakua varten ja tämän jälkeen määritellään esimerkiksi valmis AD-

ryhmä Administrator-oikeuksille PRTG-palvelussa. Asetukset on esitetty seuraavalla sivulla kuviossa 20.



Kuvio 20. PRTG AD-integrointi

Näin LemonLDAP:n AD-käyttäjäkantaa voidaan suoraan käyttää kirjautumisessa. Single sign-on toteutetaan LemonLDAP:n Form Replay –toiminnolla. Handler-komponentti tarkistaa ensin onko käyttäjä jo kirjautunut ja jos on, niin Handler-komponentti tekee jQueryn avulla automaattisen kirjautumisen kohdesivuston kirjautumislomakkeeseen. Järjestelmän toiminta esitetään vielä kuviossa 21.



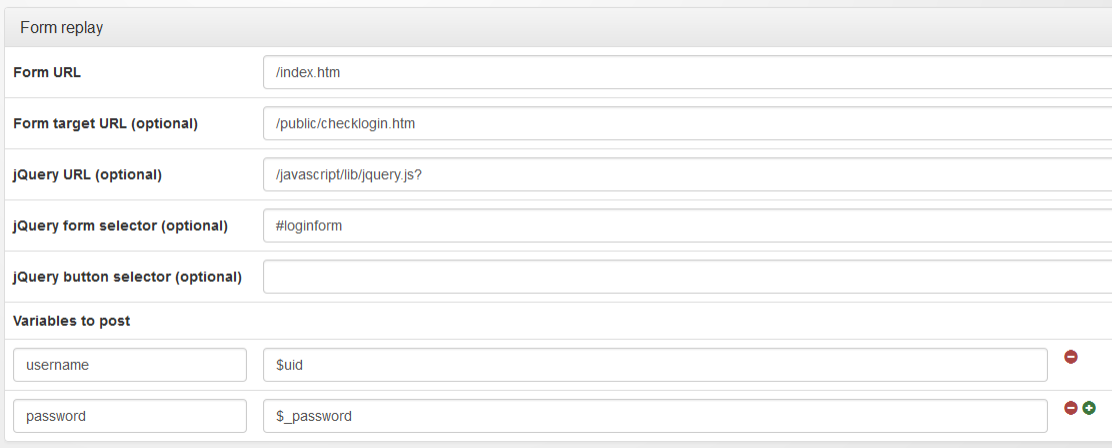
Kuvio 21. Kertakirjautuminen välityspalvelimen avulla

HTTPS on ainoa suoja käyttäjän kirjautumistiedoille, joten välityspalvelimen täytyy olla SSL-suojattu. Luodaan välityspalvelimelle tiedosto /etc/httpd/conf.d/zzz-prtgproxy.conf ja lisätään sinne rivit:

```
<VirtualHost *:80>
    ServerName monitor.example.com
    RewriteEngine on
    RewriteCond %{HTTPS} !=on
    RewriteRule ^/?(.*)$ https://%{SERVER_NAME}/$1 [R,L]
</VirtualHost>
<IfModule mod_ssl.c>
<VirtualHost monitor.example.com:443>
    ServerName monitor.example.com
    RewriteEngine on
    RewriteRule ^/(.*)$ https://prtg.example.com/$1 [P,L]
    PerlHeaderParserHandler Lemonldap::NG::Handler
    SSLEngine on
    SSLProxyEngine on
    SSLProxyCheckPeerName off
    SSLProxyCheckPeerCN off
    SSLCertificateFile    /etc/ssl/certs/prtg.crt
    SSLCertificateKeyFile /etc/ssl/certs/prtg.key
</VirtualHost>
</IfModule>
```

Koska suojauksen täytyy yltää myös välityspalvelimen ja itse monitorointipalvelimen välille, täytyy välityspalvelimen Virtual Host –asetuksiin määrittää, että sertifikaattitietoja ei erikseen varmenneta. Nämä muutokset tehdään siksi, että PRTG ja ympäristöt ylipäänsä pitävät sisällään itse allekirjoitettuja sertifikaatteja ja normaalikäyttäytymisellä Apache ei luota sertifikaatteihin, joiden tiedot eivät täsmää palvelimen nimen kanssa tai joilla ei ole luotettua varmentajaa. Tämä johtaa epäonnistuneeseen yhteyteen ja palvelu ei ole täten saatavilla.

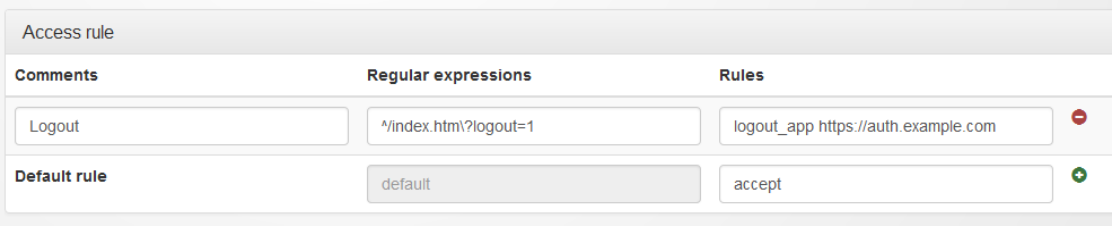
LemonLDAP:n hallintasovellukseen lisätään Virtual Host monitor.example.com ja määritellään tälle HTTPS-pakotus ja Form Replay –tiedot. Seuraavalla sivulla, kuviossa 22, esitetään Form Replay –asetukset PRTG:n web-palvelun kirjautumislomakkeeseen.



Kuvio 22. Monitor-palvelun Form Replay –asetukset

Asetuksiin syötetään lomakkeen sijainti ja halutut muuttujat sidotaan lomakkeessa esiintyviin kenttiin. Toiminta perustuu Handler-komponentin tekemään jQueryyn, joka tekee nolladatalla testin lomakkeeseen ja päättelee vastauksesta oikeat kentät, joihin sitten täydennetään asetetut muuttujat. Näiden tietojen pohjalta LemonLDAP lähettää POST-pyynnön, jonka avulla kirjautuminen suoritetaan.

Lisäksi tehdään muiden sivustojen tavoin logout-sääntö. Samoilla määrityksillä on mahdollista myös toteuttaa pääsynhallintaa yhdessä PRTG:n omien valtuutusmääritysten ohella, sillä Handler-komponentti ei ole niistä tietoinen. Kuviossa 23 esitetään luotu logout-sääntö.

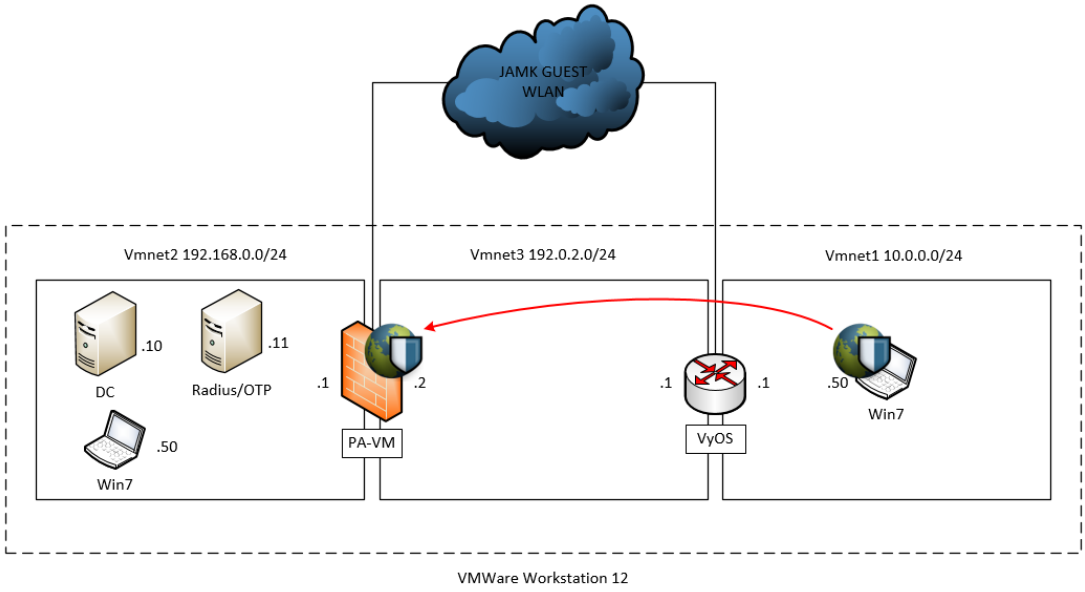


Kuvio 23. Monitor.example.com-sivuston pääsysäännöt

7.8 Kaksivaiheinen todennus

Kaksivaiheinen todennus tehtiin Palo Alton palomuurin GlobalProtect SSL-VPN palveluun. Muu testiympäristö käytti hypervisorina VirtualBoxia, mutta Palo Alton virtuaa-

lipalomuurien tuen takia kaksivaiheinen todennus toteutettiin VMWare Workstation-nin päälle. Kuviossa 24 on esitetty tätä varten luotu ympäristö käytettyine verkkoineen ja virtuaalikoneineen.



Kuvio 24. VPN-yhteyden topologia

VPN-palvelu toimii Palo Alto VM100 palomuurissa ja asiakkaana toimii Windows 7 työpöytäkone. Asiakas- ja kohdeverkko eristettiin toisistaan kolmannella verkolla, jolloin Vmnet1 ja Vmnet2 eivät voi kommunikoida keskenään ilman VPN-yhteyttä, sillä sisäverkkoihin ei osoitettu staattisia reittejä, vaan ainoa yhdistävä tekijä on Palo Alton palomuurin ja asiakaslaitteen reitittimenä toimivan VyOS:n välinen connected-verkko, Vmnet3. VPN-yhteyden käyttäjät oli tarkoitus hakea Active Directorysta ja itse kaksivaiheinen todennus toteutettiin Radius-palvelulla. Kuitenkin lopulta valittujen palvelujen luonteen vuoksi käyttäjät haettiin lopulta muista palveluista joka osoittaa, että kaksivaiheisen todennuksen kyseiseen VPN-palveluun voi toteuttaa lukuisalla eri tavalla. Toimialueena ympäristössä toimi creditbanken.de.

Käytettävistä todennuspalveluista ensimmäinen on FreeIPA. FreeIPA on avoimen lähdekoodin tuotteista rakennettu palvelukokonaisuus ja sisältää hakemistopalvelimen, MIT:n Kerberostoteutuksen ja muun muassa NTP ja DNS palvelut. FreeIPA voidaan tarvittaessa myös integroida Active Directoryyn, jolloin käyttäjäkanta voidaan edelleen ensisijaisesti pitää ympäristöjen olemassa olevilla ohjainpalvelimilla. Lisäksi

FreeIPA:n neljännestä versiosta asti palvelussa on ollut tuki OTP-pohjaiseen kaksivaiheiseen todennukseen. (FreeIPA – About N.d.)

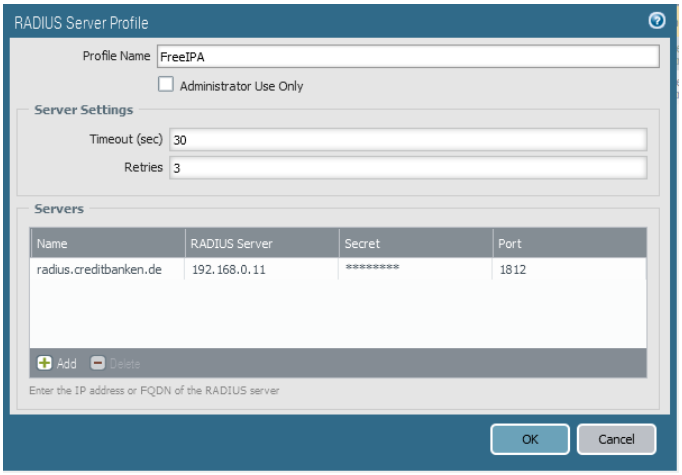
Kaksivaiheinen todennus toteutetaan käyttämällä joko HOTP- tai TOTP-algoritmia ja tästä johtuen vain kyseisiä algoritmeja tukevat tokenit käyvät yhteen FreeIPA:n kanssa. Työssä käytettiin Google Authenticator –sovellusta ja TOTP-algoritmilla luotuja kertakäyttöisiä salasanoja. Itse järjestelmä sidottiin Palo Alton VPN-yhteyteen Radius-palvelimen kautta ja sekä Radius, että FreeIPA asennettiin CentOS 7 –palvelimeen. Itse FreeIPA:n asenusta ei käydä tarkemmin läpi, sillä työn tarkastelun kohteen on, miten kaksivaiheinen todennus sidotaan VPN-yhteyteen. Kuviossa 25 esitetään onnistunut Radius-todennus järjestelmän asennuksen jälkeen.

```
[root@radius ~]# radtest h3334 Koira123 localhost 0 testing123
Sending Access-Request Id 172 from 0.0.0.0:56003 to 127.0.0.1:1812
        User-Name = 'h3334'
        User-Password = 'Koira123'
        NAS-IP-Address = 192.168.0.11
        NAS-Port = 0
        Message-Authenticator = 0x00
Received Access-Reject Id 172 from 127.0.0.1:1812 to 127.0.0.1:56003 length 20
(0) -: Expected Access-Accept got Access-Reject
[root@radius ~]# radtest h3334 Koira123780768 localhost 0 testing123
Sending Access-Request Id 149 from 0.0.0.0:33195 to 127.0.0.1:1812
        User-Name = 'h3334'
        User-Password = 'Koira123780768'
        NAS-IP-Address = 192.168.0.11
        NAS-Port = 0
        Message-Authenticator = 0x00
Received Access-Accept Id 149 from 127.0.0.1:1812 to 127.0.0.1:33195 length 20
[root@radius ~]# radtest h3334 Koira123391407 localhost 0 testing123
Sending Access-Request Id 45 from 0.0.0.0:50591 to 127.0.0.1:1812
        User-Name = 'h3334'
        User-Password = 'Koira123391407'
        NAS-IP-Address = 192.168.0.11
        NAS-Port = 0
        Message-Authenticator = 0x00
Received Access-Accept Id 45 from 127.0.0.1:1812 to 127.0.0.1:50591 length 20
```

Kuvio 25. Radius-todennus käyttäen TOTP-salasanaa

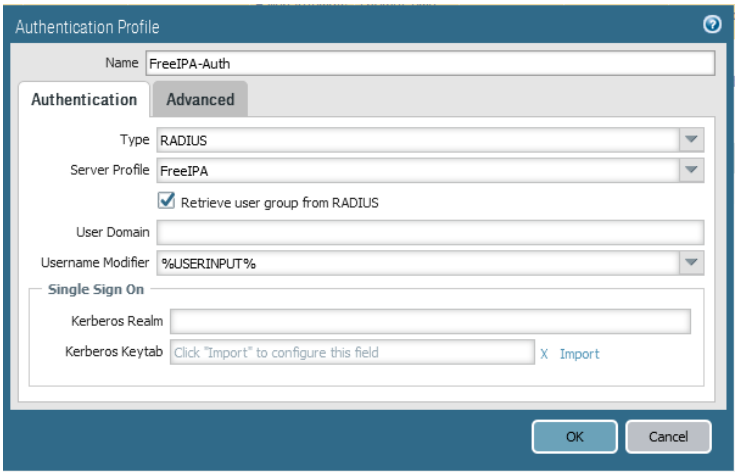
Kuviosta käy ilmi, kuinka käyttäjä ”h3334” tekee pääsypyynnön salasanalla ”Koira123” käyttäen testausyhteyttä localhostiin. Radius vastaa Access-Reject viestillä pyyntöön, sillä käyttäjän tiliin on määritetty OTP-salasanan käyttö. FreeIPA:n tapauksessa OTP kirjoitetaan käyttäjän salasanan perään ilman välilyöntejä tai välimerkkejä. Kuviossa on esitetty vielä kaksi onnistunutta Radius-todennusta peräkkäisillä OTP-salasanoilla.

Testattu ja toimiva järjestelmä sidotaan Palo Alton palomuuriin Radius-profiilin kautta. Palomuurin konfiguroimiseen käytetään tuotteen omaa Web-käyttöliittymää. Radius-profiili määritetään Server Profiles –valikkoon seuraavan sivun kuvion 26 esittämällä tavalla.



Kuvio 26. Radius-profiilin määrittäminen

Asetuksiin määritetään Radius-palvelimen IP-osoite ja jaettu salaisuus (shared secret), jolla Radius-palvelin tunnistaa asiakkaan, eli palomuurin. Seuraavaksi sidotaan luotu Radius-profiili todennusprofiiliin (Authentication Profile). Profiiliin määritetään luotu Radius-profiili. Kuviossa 27 esitetään tarvittavat asetukset yhteydelle.



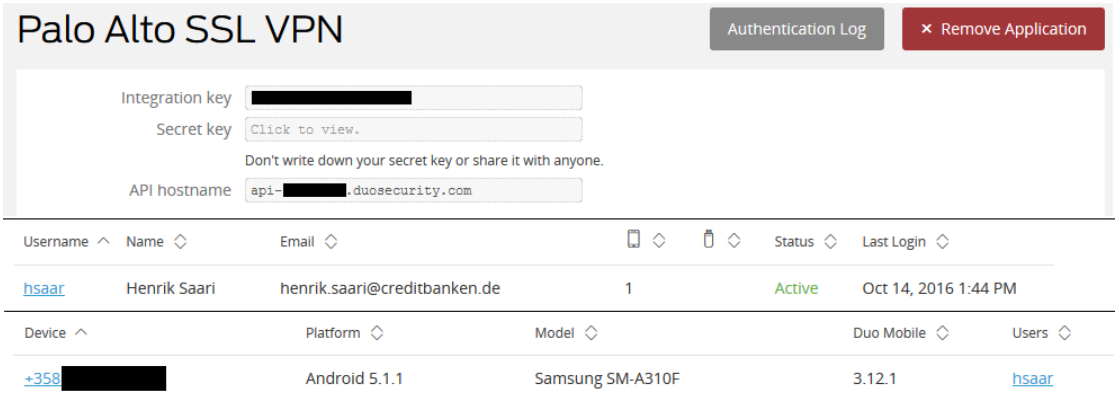
Kuvio 27. Profiilin luominen

Huomattavaa on, että palvelun toimiminen vaatii, että käyttäjät noudetaan Radiukselta, sillä Radius itse noutaa käyttäjät FreeIPA:n hakemistosta. Lopuksi todennusprofiili täytyy vielä sitoa itse luotuun GlobalProtect –palveluun. Todennusprofiili sidotaan sekä luotuun yhdyskäytävään, että portaaliin. Tämän jälkeen palvelu on käytettä-

vissä. Kirjautuminen tapahtuu GlobalProtect –asiakassovelluksella, syöttämällä käyttäjätunnus sekä salasana ja OTP yhdistettynä, kuten edellä Radius-todennusta testatessa.

Toinen testattu kaksivaiheisen todennuksen toteuttava järjestelmä on Duo Securityn todennuspalvelu. Duon palvelu nojaa myös Radius-protokollaan ja käyttäjäkanta sidotaan Duon omaan pilvipalveluun. Omaa ympäristöään voi hallita selaimen kautta ja admin-paneelin kautta sidotaan yhteen suojattavat palvelut, käyttäjät ja käyttäjien tokenit. Järjestelmä on myös integroitavissa Active Directoryyn, mutta koska todennuksen onnistuminen vaatii yhteyden julkiseen Internetiin, tämä tietenkin tarkoittaa, että suljettuun RGCE-ympäristöön palvelu ei sovellu, mutta työssä haluttiin myös kokeilla jotain kaupallista tuotetta. Testikäyttöön sopivan tilin saa ilmaiseksi, ja se tukee kymmentä käyttäjää. Automaattinen Active Directory –synkronointi on saatavilla maksullisissa versioissa. (Duo – Features 2016.)

Duo tilin aktivoinnin myötä käyttäjä pääsee luomaan suojatun sovelluksen, jolle generoidaan palvelun yksilöivät tunnistetiedot. Lisäksi käyttäjät voidaan luoda käsin ja käyttäjälle voidaan sitoa token todennusta varten. Kuviossa 28 on esitetty muutokset admin-paneelissa. Tokenina toimii Duon puhelinsovellus.



Kuvio 28. Duo admin-paneeli

Duon oma Radius-palvelin tukee sekä Windows, että Linux –käyttöjärjestelmiä ja toimii välityspalvelimena julkisen palvelun ja sisäverkon Radius-todennuksen välillä. Työhön valittiin CentOS 7 –palvelin ja palvelimelle asennettiin tarvittavat paketit komennoilla:

wget https://dl.duosecurity.com/duoauthproxy-latest-src.tgz
yum install gcc make openssl-devel python-devel
tar xzf duoauthproxy-<latest-src>.tgz
cd duoauthproxy-<version-src>
export PYTHON=python2.7
make
cd duoauthproxy-build
./install

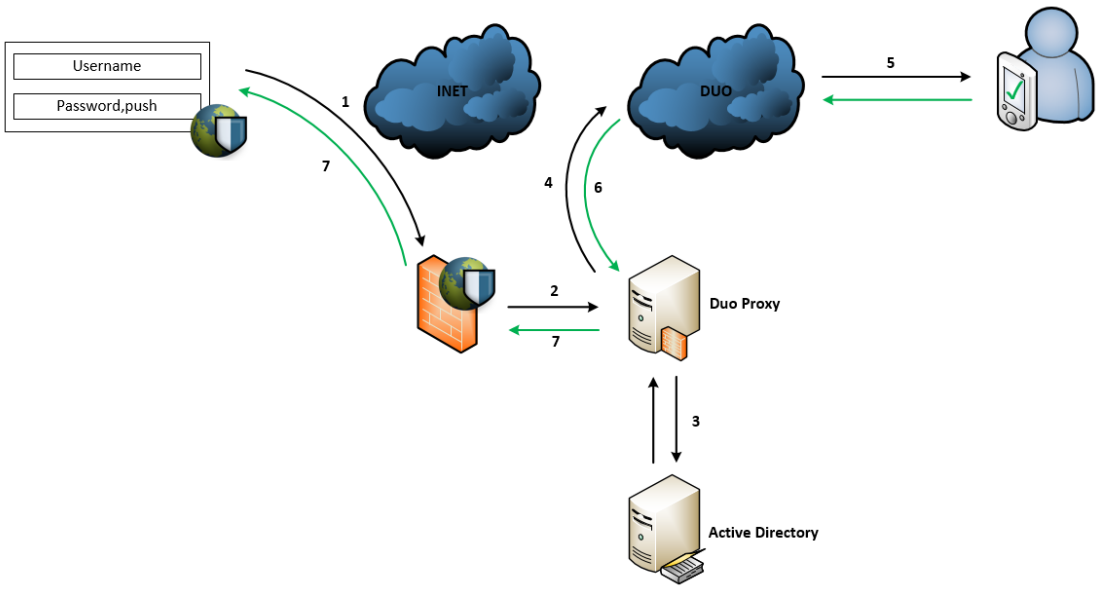
Välityspalvelin konfiguroitiin hakemaan käyttäjät Active Directorysta ja toimimaan Radius-palvelimena Palo Alto –palomuurille. Asetukset tehdään kuvion 29 esittämällä tavalla tiedostoon /opt/duoauthproxy/conf/authproxy.cfg.

Kuvio 29. Authproxy.cfg:n asetukset

Käyttäjähaun onnistuminen vaatii AD-käyttäjän, jolla on lukuoikeus hakemistoon. Yksinkertaisuuden vuoksi tätä varten luotiin uusi käyttäjätili. Radius-konfiguraatioon sidottiin aiemmin luodut avaimet ja Duon rajapinnan osoite, sekä lisäksi Radius-asiakkaan osoite ja jaettu salaisuus.

Palomuuriin tehdään uusi Radius-profiili, joka sidotaan tällä kertaa Duon välityspalvelimeen. Radius-profiili vaihdetaan käyttöön VPN-yhteyden yhdyskäytävään ja portaaliin. Tämän jälkeen järjestelmä on toimintakykyinen ja kirjautuminen voidaan suorittaa GlobalProtect-asiakassovelluksen avulla, joko syöttämällä käyttätunnus ja salasana,otp tai käyttäjätunnus ja salasana,push. Push-toiminto lähettää nimensä mukaisesti push-ilmoituksen käyttäjän puhelimena toimivaan tokeniin, jonka käyttäjä voi

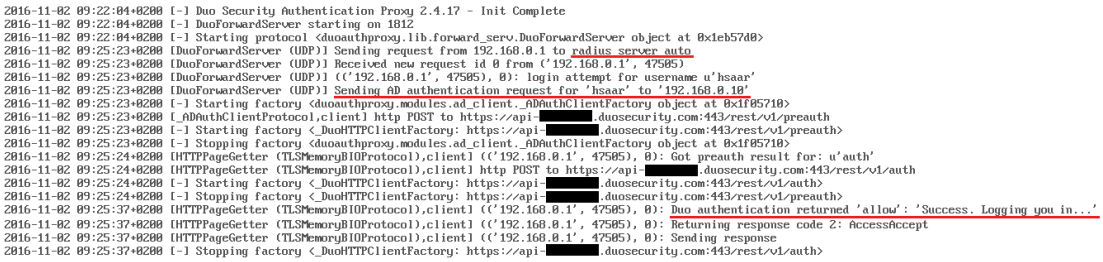
hyväksyä tai hylätä. Kuviossa 30 esitetään vielä järjestelmän toiminta ja siihen liittyvät erilliset komponentit.



Kuvio 30. Duon välityspalvelimen toiminta (Duo - Palo Alto 2016).

Kuvio esittää tilanteen, jossa käyttäjä kirjautuu GlobalProtect-asiakassovellukseen käyttäen todennusmuotonaan push-toimintoa (1). Palomuuri ottaa yhteyttä Duon välityspalvelimeen Radius-protokollalla (2) ja välityspalvelin suorittaa ohjainpalvelimelle todennuspyynnön (3). Tämän jälkeen välityspalvelin avaa HTTPS-yhteyden Duon pilvipalveluun (4) ja palvelusta haetaan AD-käyttäjää vastaava tili ja tähän tiliin sidottuun puhelinnumeroon lähetetään push-viesti, jonka käyttäjä hyväksyy (5). Pilvipalvelu palauttaa tiedon välityspalvelimelle (6) ja palvelin palauttaa Radius Access-Accept –viestin (7). (Duo - Palo Alto 2016).

Seuraavalla sivulla, kuviossa 31, esitetään välityspalvelimen lokiviestit onnistuneesta todennuksesta. Punaisella korostetuista kohdista voidaan nähdä, kuinka muurilta saapuva Radius-pyyntö ohjataan konfiguraatiotiedostossa määritetylle Radius-profiilille, joka kääntää pyynnön ohjainpalvelimelle. Myös kutsut Duon pilvipalvelun rajapintaan ovat nähtävissä.



Kuvio 31. Onnistunut todennus Duon välityspalvelimen kautta

8 Ympäristön toiminta

8.1 Yleistä

Tässä kappaleessa havainnollistetaan käytettyjä tekniikoita muutaman esimerkin avulla. Kappaleella pyritään luomaan silta tehtyjen asetusten ja loppukäyttäjän näkökulman välille. Todennus esitetään sähköpostipalvelimen selainpohjaiseen palveluun ja intrasivustoon, sillä nämä kaksi palvelua sisältävät kaikki tässä työssä käytetyt tekniikat.

2FA todennetaan Duo-palvelun avulla, sillä todennuksen kulku on hyödyllistä käydä lävitse. Lisäksi palvelussa hyödynnettävät push-viestit sitovat asiakkaan eli loppukäyttäjän näkökulman muuhun lokidataan, joka on saatavilla todennustapahtumasta.

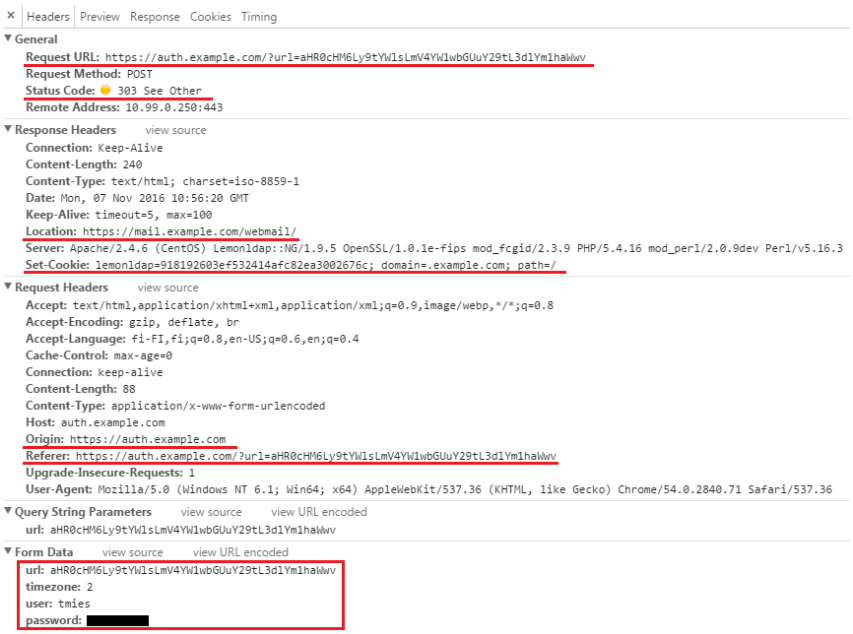
8.2 Roundcubemail

Roundcubemailin kertakirjautuminen toteutettiin käyttämällä hyväksi Roundcuben omaa HTTP-todennusliitännäistä. HTTP-pyyntö osoitteeseen mail.example.com havaintaan LemonLDAP:n Handler-komponentin toimesta ja käyttäjä ohjataan kirjautumaan Portaaliin, jos LemonLDAP:n SSO-evästettä ei löydy pyynnöstä. Seuraavalla sivulla, kuviossa 32, esitetään selaimen Network Monitor –toiminnosta HTTP-pyynnöt ja vastaus.

Kuvio 32. HTTP-pyyntö webmailiin

Kuten vastausotsikoista (Request Headers) voidaan todeta, pyynnöstä puuttuu LemonLDAP:n eväste. Handler-komponentti vastaa tällöin pyyntöön redirect-koodilla 302 ja ilmoittaa vastausotsikoissa sijainniksi Portal-komponentin osoitteen ja asettaa pyynnölle ainutlaatuisen GET-parametrin, jolla käyttäjä osataan myöhemmin ohjata takaisin.

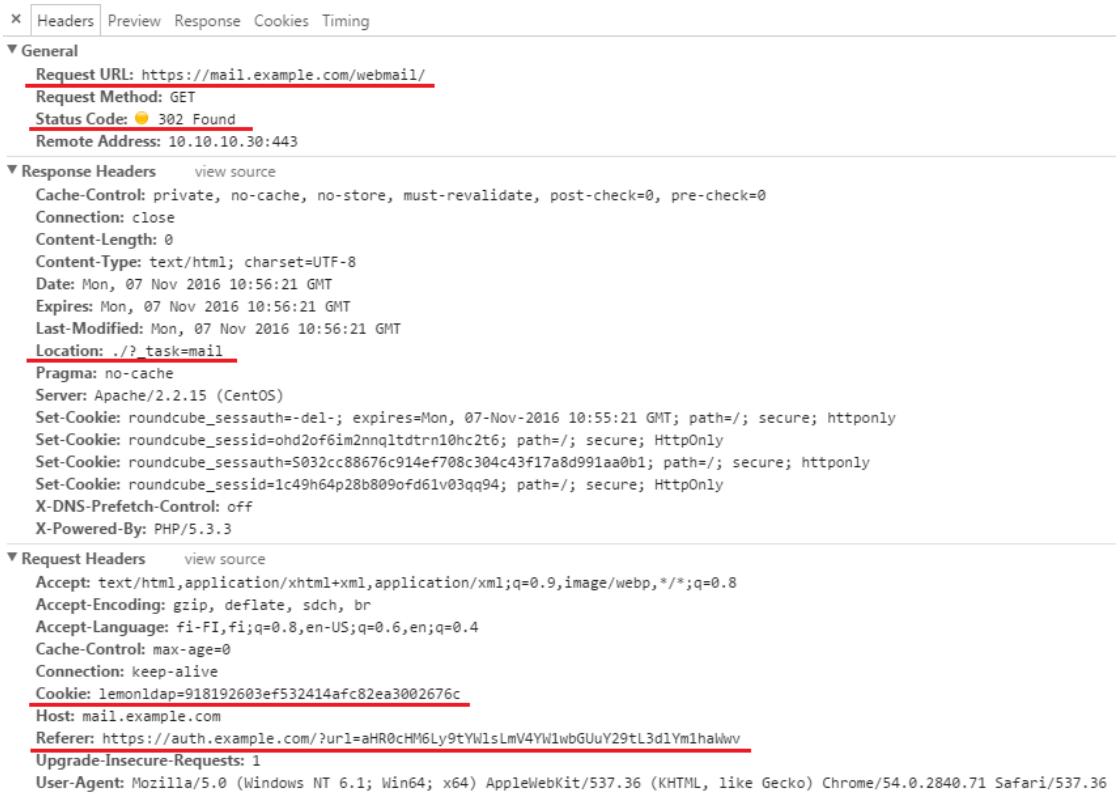
Kuviossa 33 esitetään todennuksen jälkeinen uudelleenohjaus takaisin alkuperäiseen resurssiin. Vastausotsikoista voidaan huomata, että samalla asetetaan LemonLDAP:n SSO-eväste.



Kuvio 33. Todennuksen jälkeinen uudelleenohjaus sähköpostiin

Pyyntö ohjataan alkuperäisessä uudelleenohjauksessa luotuun GET-parametriin, jonka avulla Portal-komponentti osaa ohjata käyttäjän takaisin sähköpostiin. Lisäksi pyynnössä näkyvät Portaalin kirjautumislomakkeeseen syötetyt tiedot, joista esimerkiksi käyttäjän salasana poimitaan $_password-muuttujaan, jos salasana on tarkoitus välittää kohdesovellukselle, kuten tässä tapauksessa on.

Kuviossa 34 esitetään automaattisesti tapahtuva kirjautuminen. Sähköpostipalvelin poimii pyynnön mukana lähetettävät HTTP_AUTH_USER ja HTTP_AUTH_PW –otsikot ja suorittaa niillä käyttäjän kirjautumisen.

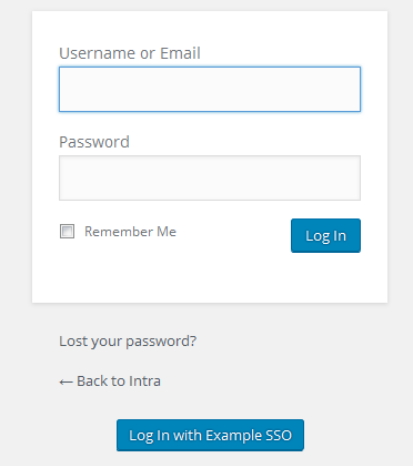


Kuvio 34. Kirjautuminen webmailiin

Kuten kuviosta voidaan todeta, pyynnössä on mukana LemonLDAP:n SSO-eväste. Handler-komponentti päästää pyynnön läpi ja palvelin luo yllämainituilla HTTP-otsikoilla käyttäjälle istunnon ja asettaa Roundcuben omat evästeet, sekä uudelleenohjaa käyttäjän omaan postilaatikkoonsa, joka sijaitsee osoitteessa ”/?_task=mail”.

8.3 Intra

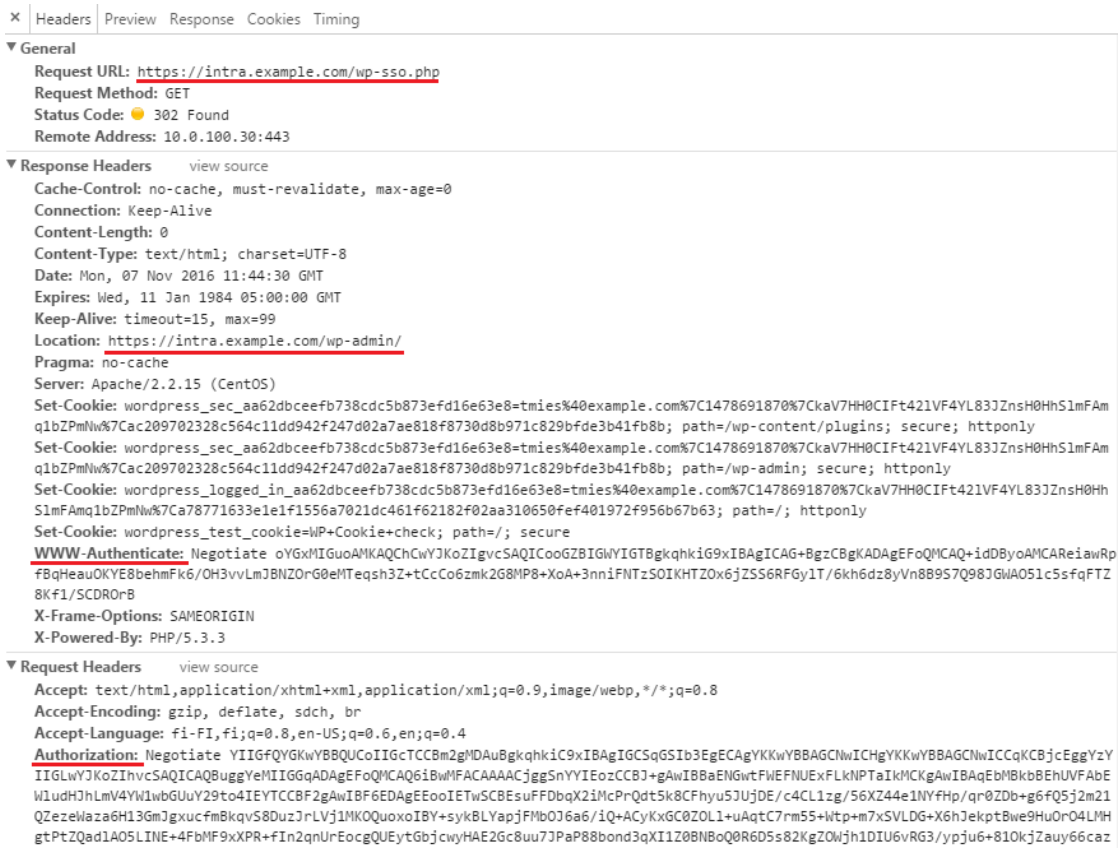
Intran kertakirjautuminen toteutettiin yhdistämällä LemonLDAP:n Handler-komponentti ja Kerberos-todennus. Handler-komponentti on vastuussa käyttäjän pyyntöjen suodattamisesta ja Kerberos-moduuli toimii SSO-agenttina ja vie loppukäyttäjän kirjautumistiedot Wordpress-sovellukselle. Kuviossa 35 esitetään intran kirjautumissivu. Kertakirjautuminen on toteutettu sillä tavoin, että myös paikallisilla tileillä voidaan kirjautua.



Kuvio 35. Intran kirjautumissivu

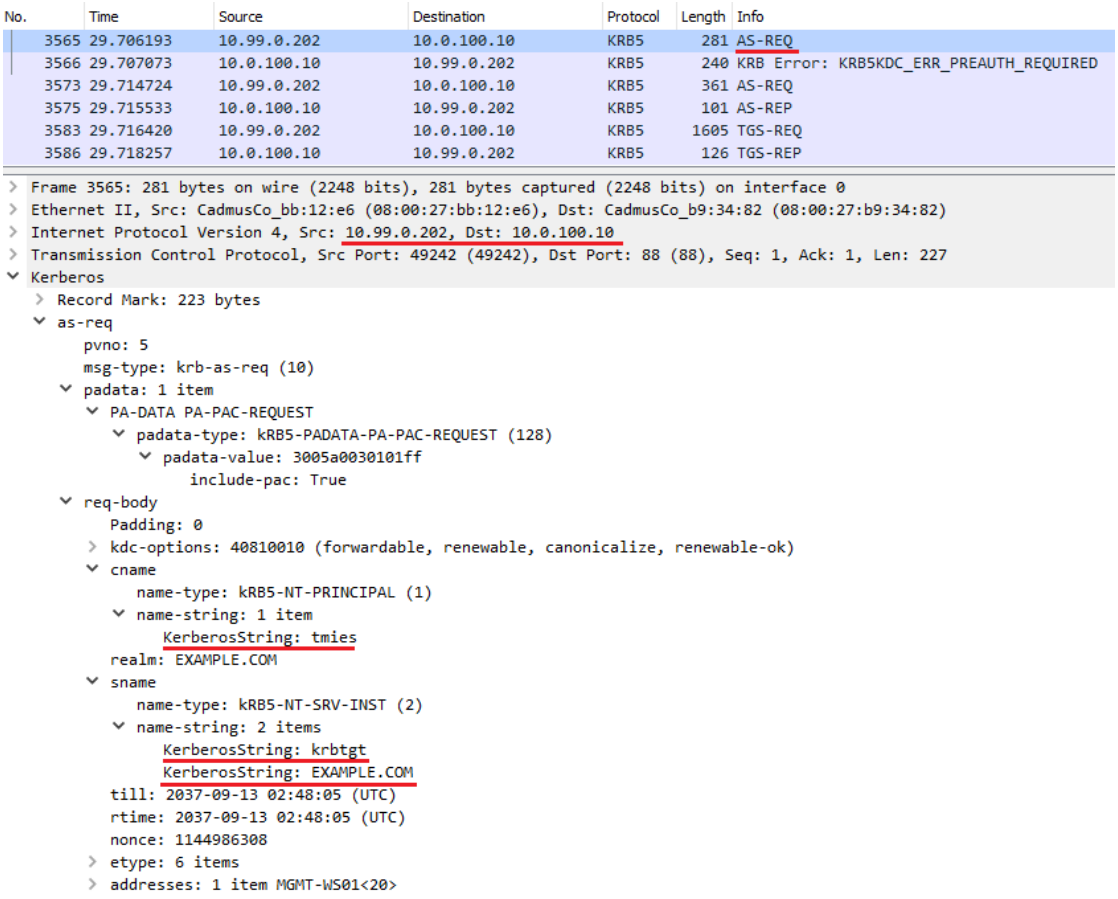
SSO-kirjautumiseen käytetään kuvion alaosassa näkyvää ”Log in with Example SSO” –painiketta, joka ohjaa käyttäjän ”./wp-sso.php” –osoitteeseen, jossa Handler-komponentti kuuntelee pyyntöjä. Jos eväste puuttuu pyynnöstä, käydään todennus läpi Roundcubemailin tavoin.

Seuraavan sivun kuviossa 36 esitetään HTTP-pyyntö onnistuneen Portal-kirjautumisen jälkeen. Kuviosta on havaittavissa intra-palvelimen vaatima HTTP-todennus, WWW-Authenticate, johon loppukäyttäjän selain vastaa Authorize-otsikolla. Onnistuneen Kerberos-todennuksen johdosta käyttäjä ohjataan ”/wp-admin/” –paneeliin ja samalla asetetaan Wordpressin omat evästeet.



Kuvio 36. HTTP-todennus intra-palvelimella

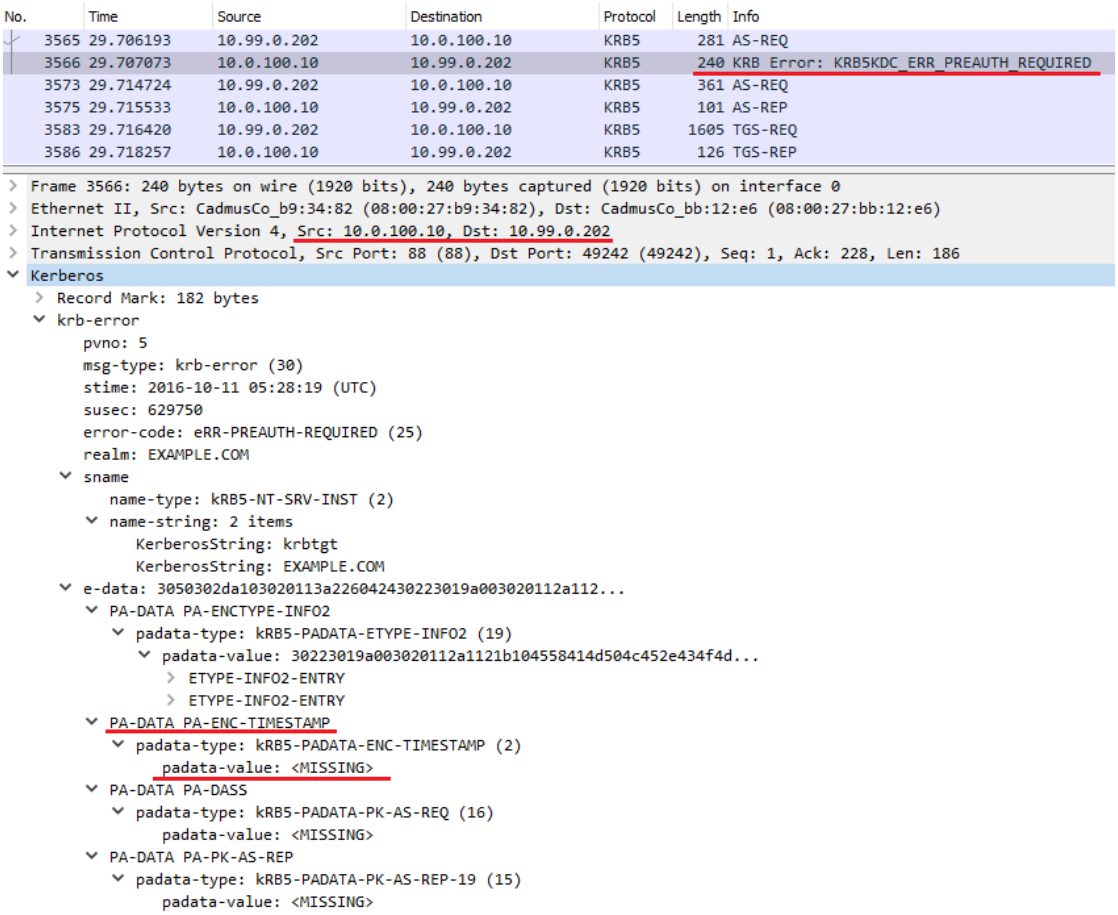
Taustalla vaikuttava Kerberos pyytää tiketin palveluun automaattisesti ja loppukäyttäjältä piilossa. Ensimmäiseksi AS-palvelimelta pyydetään TGT-tiketti. Seuraavalla sivulla, kuviossa 37, esitetään Wireshark-kaappaus Kerberos-todennuksesta.



Kuvio 37. AS_REQ-viesti

Kuten kuviosta huomataan, todennus lähtee liikkeelle AS_REQ-pyynnöllä AS-palvelimelle. Pyynnöstä on selkeästi havaittavissa pyynnön suorittavan käyttäjän päänimi ja pyydettävät tikettityyppi, krbtgt, eli Kerberoksen TGT-tiketti.

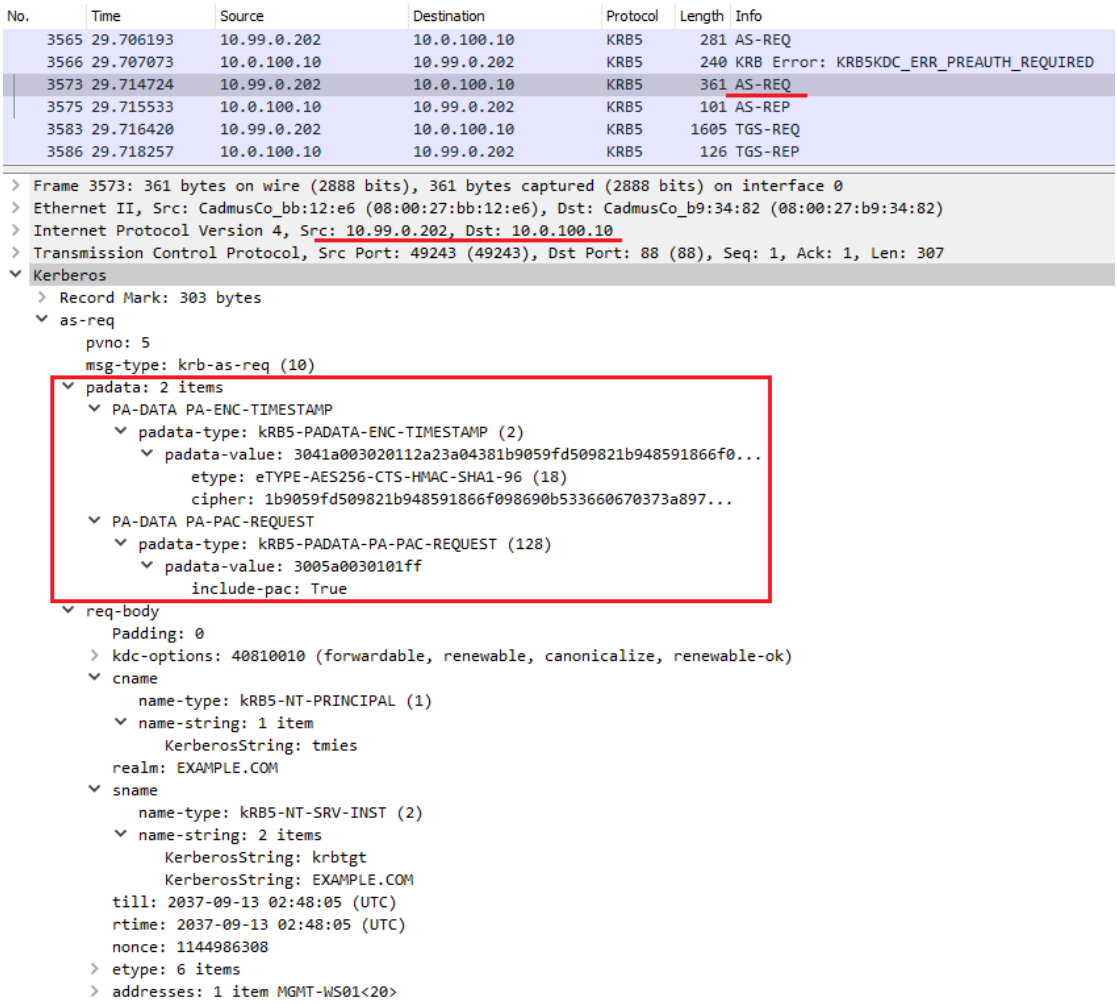
AS-palvelin vastaa ensimmäiseen pyyntöön KRB_ERROR-viestillä, sillä AS-palvelin vaatii loppukäyttäjän todentavan itsensä palvelulle. Seuraavalla sivulla sijaitsevassa kuviossa 38 esitetään KRB_ERROR-viestin rakenne.



Kuvio 38. KRB_ERROR-viesti

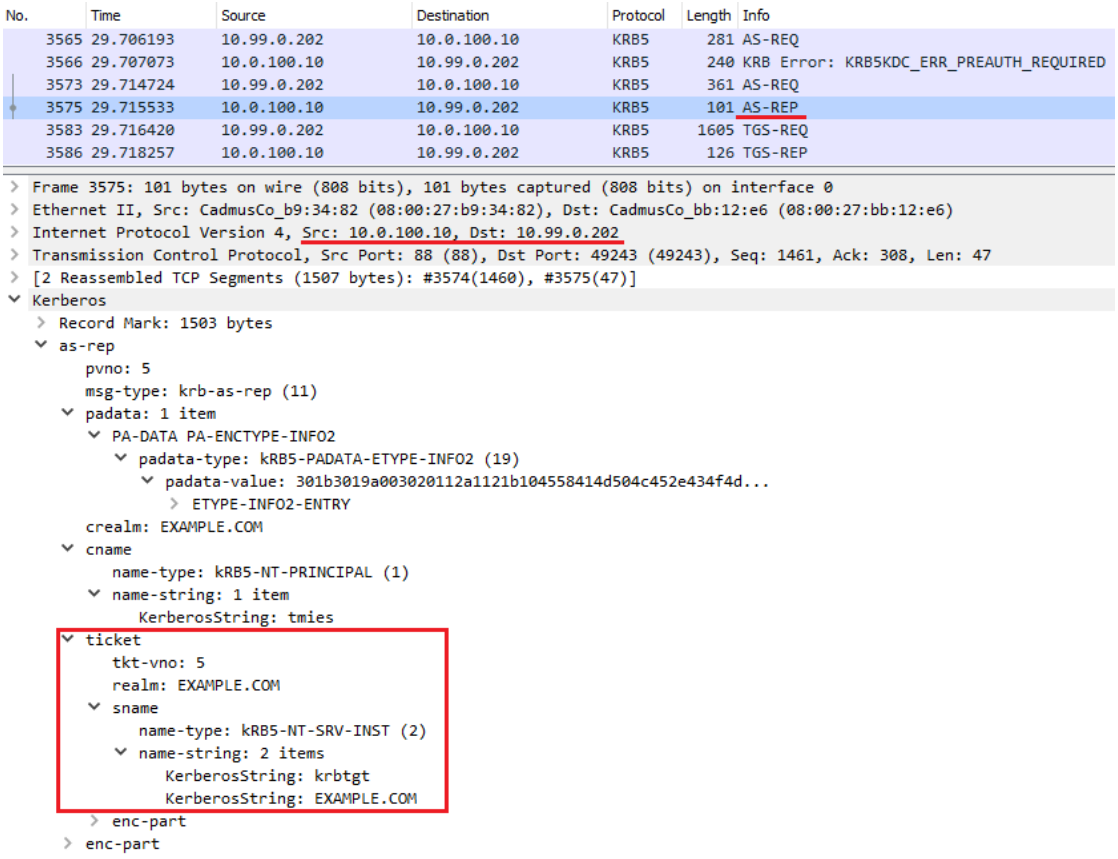
Kuviosta on todettavissa, että alkuperäisestä AS_REQ-viestistä puuttui esitodennuksen mahdollistava PA-ENC-TIMESTAMP –kenttä. Kyseinen kenttä on käytännössä aikaleima salattuna loppukäyttäjän avaimella. Tämä on yksi syy sille, miksi Kerberos on niin herkkä aikasynkronoinnin suhteen. Liian suuri aikaero palvelimen ja loppukäyttäjän sisäisissä kelloissa estää esitodennuksen onnistumisen.

Kuviossa 39 esitetään AS_REQ-pyyntö, joka sisältää esitodennuksen. Muuten pyyntö on identtinen alkuperäisen AS_REQ-pyynnön kanssa, sisältäen jälleen kerran käyttäjän päänimen ja halutun tikettityypin. Kuvio sijaitsee seuraavalla sivulla.



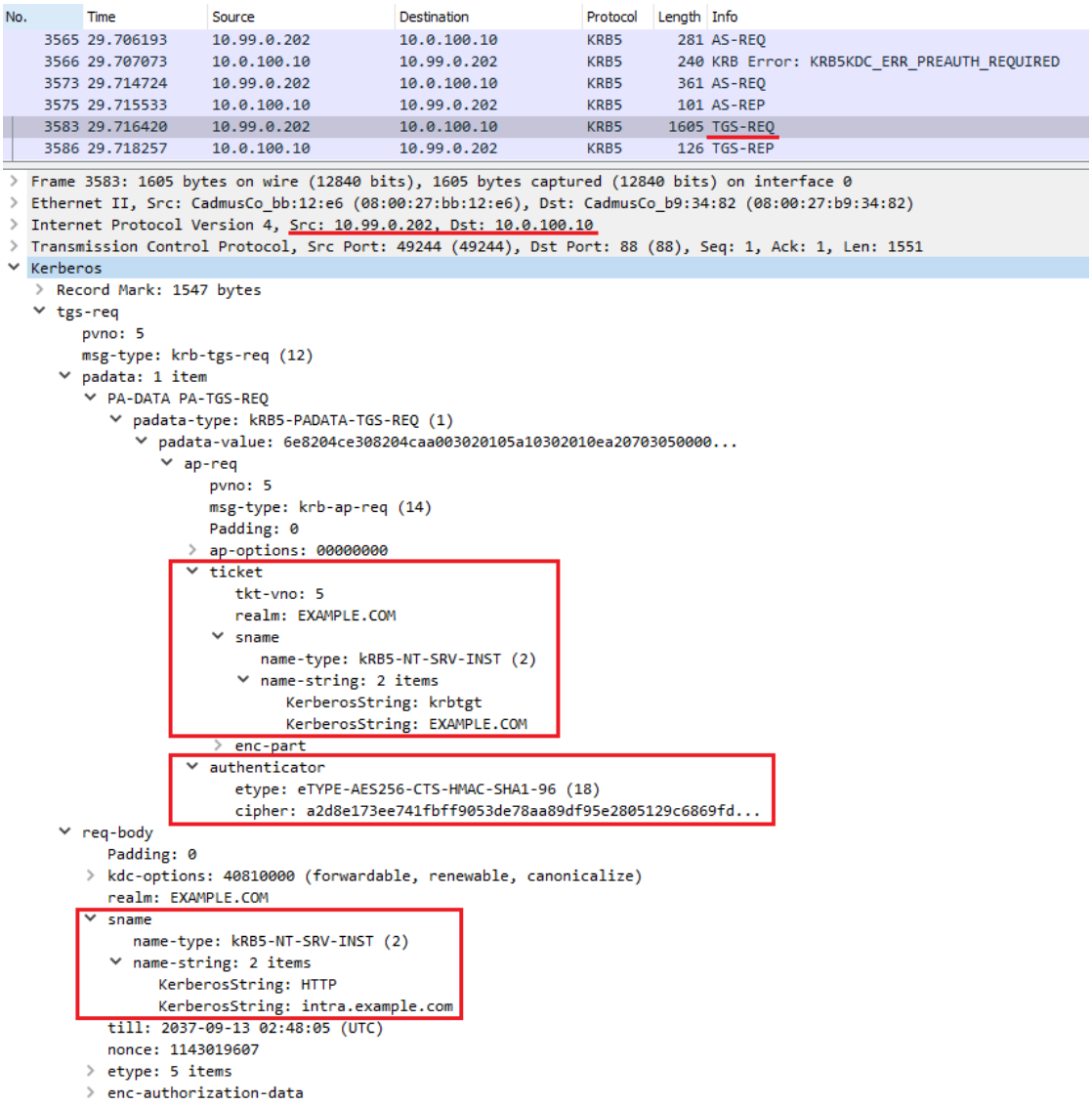
Kuvio 39. AS_REQ-viesti esitodennuksella

Palvelin vastaa pyyntöön AS_REP-viestillä. Viesti sisältää loppukäyttäjän päänimen ja myönnetyn tiketin ”krbtgt/EXAMPLE.COM@EXAMPLE.COM”. Nyt loppukäyttäjän on mahdollista noutaa palvelutiketti itse halutulle palvelimelle, intra.example.com, käyttäen hyväksi TGS-palvelinta. Seuraavan sivun kuviossa 40 on esitetty AS_REP-viestin rakenne.



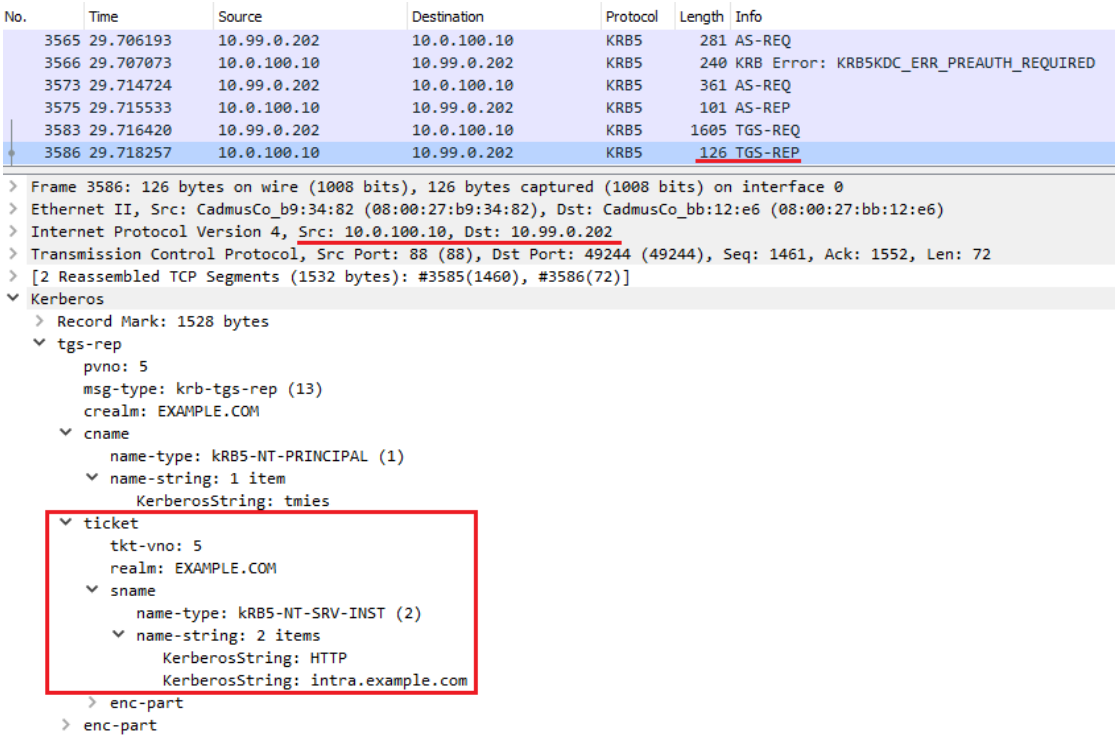
Kuvio 40. AS_REP-viesti

Pyyntö TGS-palvelimelle suoritetaan TGS-REQ-pyynnöllä. Pyyntöön sidotaan myönnetty TGT-tiketti, autentikaattori ja haluttu SPN. Kuviossa 41 esitetään TGS_REQ-pyyntö TGS-palvelimelle dc.example.com. Kuvio sijaitsee seuraavalla sivulla.



Kuvio 41. TGS_REQ-viesti

TGS-palvelin vastaa pyyntöön TGS_REP-viestillä, jossa loppukäyttäjälle myönnetään palvelutiketti haluttuun palveluun. Seuraavalla sivulla, kuviossa 42, esitetään TGS_REP-viesti ja viestin sisällöstä voidaan selkeästi nähdä, kuinka asiakkaalle ”tmies@EXAMPLE.COM”, myönnetään palvelutiketti ”HTTP/intra.example.com@EXAMPLE.COM”.



Kuvio 42. TGS_REP-viesti

Luodut tiketit voidaan tarkistaa vielä asiakaskoneelta. Kuviossa 43 esitetään asiakaskoneella pyyntö listata voimassaolevat Kerberos-tiketit. Kuviosta on huomattavissa, että TGT-tiketti ja intran palvelutiketti ovat määritetty voimassaoleviksi TGT-tiketin mukaan. Ilman voimassaolevaa TGT-tikettiä ei siis ole pääsyä muihinkaan kerberoituihin palveluihin.

```
C:\Users\tmies>klist

Current LogonId is 0:0x1b44d

Cached Tickets: (3)

#0>     Client: tmies @ EXAMPLE.COM
        Server: krbtgt/EXAMPLE.COM @ EXAMPLE.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
        Start Time: 11/7/2016 13:41:49 (local)
        End Time:   11/7/2016 23:41:49 (local)
        Renew Time: 11/14/2016 13:41:49 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96


#1>     Client: tmies @ EXAMPLE.COM
        Server: HTTP/intra.example.com @ EXAMPLE.COM
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 11/7/2016 13:44:29 (local)
        End Time:   11/7/2016 23:41:49 (local)
        Renew Time: 11/14/2016 13:41:49 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
```

Kuvio 43. Käyttäjän Ticket Cache

Uloskirjaus intra-palvelusta ohjaa käyttäjän takaisin portaaliin, mutta ei pura SSO-istuntoa. Kuviossa 44 on esitetty selaimen Network Monitor –toiminnosta uloskirjautumisen uudelleenohjaus. Uloskirjautuminen ohjaa käyttäjän 302-tilakoodilla Portal-komponentille.

Name	Status	Type	Initiator	Size	Time
wp-login.php?action=logout&_wpnonce=99cea807df	302	text/html	Other	2.0 KB	156 ms
auth.example.com	200	document	https://intra.exampl...	3.5 KB	50 ms

Kuvio 44. Uloskirjautuminen intrasta

8.4 Kaksivaiheinen todennus

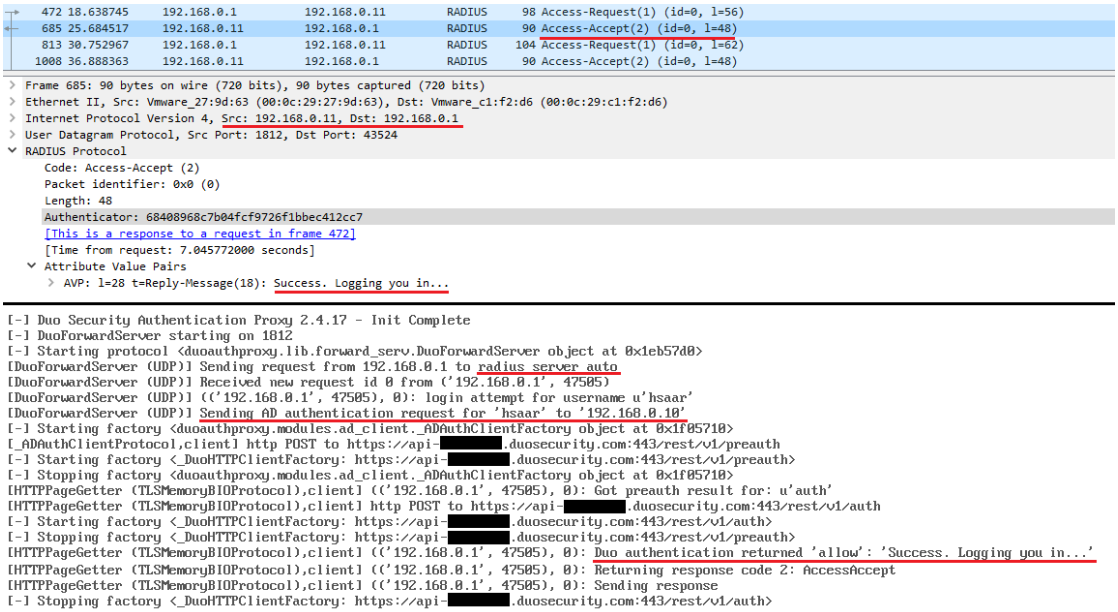
Kaksivaiheinen todennus esitetään Duon palvelua käyttäen, sillä verrattuna FreeIPA:n, todennuksen vaiheet ovat paremmin havaittavissa visuaalisesti. Kun käyttäjä tekee kirjautumispyynnön GlobalProtect-asiakasohjelmalla, palomuuri välittää pyynnön Radius-palvelimelle. Kuviossa 45 esitetään Wireshark-kaappaus Radius-viesteistä palomuurin ja Radius-palvelimen (192.168.0.11) välillä. Radius-palvelimena toimii siis Duon välityspalvelin.

```
   472 18.638745    192.168.0.1     192.168.0.11    RADIUS     98 Access-Request(1) (id=0, l=56)
   685 25.684517    192.168.0.11    192.168.0.1     RADIUS     90 Access-Accept(2) (id=0, l=48)
   813 30.752967    192.168.0.1     192.168.0.11    RADIUS    104 Access-Request(1) (id=0, l=62)
  1008 36.888363    192.168.0.11    192.168.0.1     RADIUS     90 Access-Accept(2) (id=0, l=48)

> Frame 472: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
> Ethernet II, Src: Vmware_c1:f2:d6 (00:0c:29:c1:f2:d6), Dst: Vmware_27:9d:63 (00:0c:29:27:9d:63)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.11
> User Datagram Protocol, Src Port: 43524, Dst Port: 1812
v RADIUS Protocol
    Code: Access-Request (1)
    Packet identifier: 0x0 (0)
    Length: 56
    Authenticator: c3072a8f7f64b9041500000000000000
    [The response to this request is in frame 685]
  v Attribute Value Pairs
    > AVP: l=7 t=User-Name(1): hsaar
    > AVP: l=18 t=User-Password(2): Encrypted
    > AVP: l=5 t=NAS-Identifier(32): DUO
    > AVP: l=6 t=NAS-IP-Address(4): 192.168.0.1
```

Kuvio 45. Pyyntö Duon välityspalvelimelle

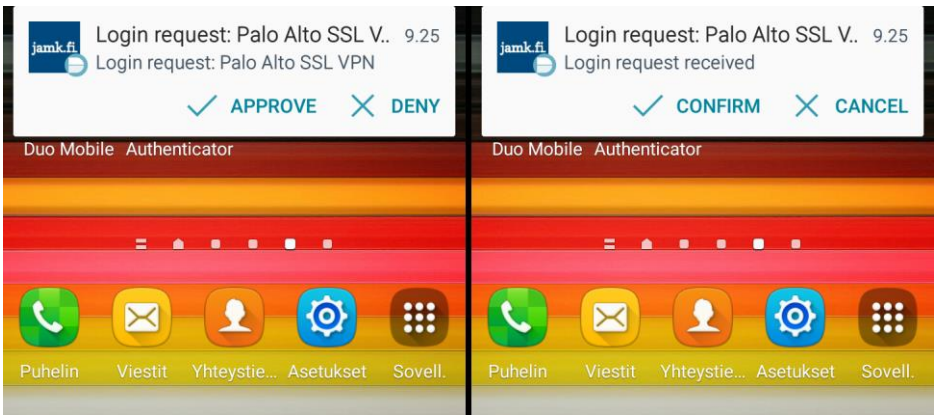
Duon välityspalvelin ottaa vastaan Radius-pyynnön ja suorittaa ensimmäisen todennuksen AD-kantaa vastaan. Todennuksen onnistuessa välityspalvelin suorittaa toisen todennuksen Duon pilvipalvelun yksilölliseen rajapintaan. Duon omat lokitiedot ja

Wireshark-kaappaus näyttävät todennuksen kulun. Kuviossa 46 esitetään pääsyn salliva Radius-viesti rinnastettuna alla näytettävään Duon välityspalvelimen lokitietoon.



Kuvio 46. Duon välityspalvelimen vastaus

Kuviosta on huomattavissa onnistuneesta todennuksesta kertova viesti ”Success. Logging you in…”. Viesti on havaittavissa sekä Radius-palvelimen vastauksesta, että välityspalvelimen lokiviestistä. Lisäksi lokiviestistä huomataan, että ennen pääsyn sallimista, välityspalvelin tekee toisen todennuksen palveluun sidottuun API-rajapintaan. Tämä rajapinta on vastuussa joko OTP-salasanan verifioinnista tai push-viestin lähettämisestä. Kuviossa 47 esitetään todennuksessa käytetyn push-toiminnon näkyminen asiakkaalle.



Kuvio 47. Duon push-ilmoitus

Käyttäjän hyväksyttyä varmistuksen, palauttaa Duon pilvipalvelu tiedon tästä välityspalvelimelle ja välityspalvelin lähettää Access-Accept –viestin palomuurille, jonka jälkeen VPN-yhteys muodostuu.

Koko toiminta on helposti havaittavissa palomuurin liikennelokista. Kuviossa 48 on esitetty suodatettu pätkä liikennelokia. Kuviosta voidaan havaita, että lähteestä 192.168.0.11 eli Radius-palvelimelta tulee liikennettä API-rajapintaan ja vasta tämän jälkeen GlobalProtect-yhteys yhdistyy. Asiakkaan IP-osoite 192.0.2.1 on sama kuin asiakkaan VyOS-reitittimen, sillä asiakkaan päässä käytettiin PAT-tekniikkaa (Port Address Translation).

(addr.src in 192.0.2.1) or (addr.dst in 52.59.243.192)

	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule
	11/07 14:50:56	end	Untrust	Untrust	192.0.2.1		192.0.2.2	443	panos-global-protect	allow	Default ANY
	11/07 14:50:01	end	Untrust	Untrust	192.0.2.1		192.0.2.2	443	web-browsing	allow	Default ANY
	11/07 14:50:01	end	Untrust	Untrust	192.0.2.1		192.0.2.2	443	web-browsing	allow	Default ANY
	11/07 14:49:56	end	Trust	Untrust	192.168.0.11	creditbanken\hs...	52.59.243.192	443	ssl	allow	Default ANY
	11/07 14:49:50	end	Trust	Untrust	192.168.0.11	creditbanken\hs...	52.59.243.192	443	ssl	allow	Default ANY
	11/07 14:49:40	end	Untrust	Untrust	192.0.2.1		192.0.2.2	443	ssl	allow	Default ANY

Kuvio 48. Palomuurin liikenneloki

9 Pohdinta

9.1 Tulokset

Työn tarkoituksena oli toteuttaa organisaatiokohtainen autentikaatiojärjestelmä ja kertakirjautuminen RGCE-kehitysympäristöön ja lisäksi tutkia ja toteuttaa kaksivaiheinen todennus Palo Alton GlobalProtect VPN-palveluun. Työn toteutukselle pyrittiin teoriaosuudessa luomaan tarvittava tietopohja, sekä esittämään vaihtoehtoja. Vaihtoehdot käytiin yleisellä tasolla läpi niin ominaisuuksiltaan, kuin toimintatavoiltaankin ja tällä pyrittiin perustelemaan käytetyt ratkaisut ja tehdyt valinnat.

Lähtökohtaisesti työssä pyrittiin käyttämään uudempia tekniikoita, pääasiallisesti OpenID Connectia ja OAuthia, mutta toteutuksen mutkistuessa tämän vuoksi päätettiin alkuperäisestä suuntauksesta luopua. Tämä sivuseikkailu antoi kuitenkin näkemyksen ja kokemusta siitä, kuinka hankalaa voi SSO:n toteuttaminen pahimmillaan olla.

Suurin osa valitun palvelun ominaisuuksista testattiin työn aikana ja tältä pohjalta saatiin kehitettyä valmis komponentti ja sen asennusta helpottavat skriptit. Valmis komponentti saatiin siirrettyä RGCE-ympäristöön ja organisaatioiden palveluja varten onnistuttiin luomaan oma repository, joka helpottaa huomattavasti suljetussa ympäristössä tapahtuvia asennuksia.

Työssä saavutettuihin tuloksiin voidaan siinä mielessä olla tyytyväisiä, että alkuperäisen toimeksiannon vaatimukset täytettiin ja järjestelmä on käyttövalmis. Kaksivaiheinen todennus onnistuttiin toteuttamaan ja toimeksiantajalle esiteltiin vaihtoehtoisia palveluita 2FA:n toteuttamiseen.

Suurimmat vaikeudet työtä tehdessä johtuivat työn tekijän olettamuksista. Esimerkiksi Handler-komponentin asennus etäpalvelimelle tuotti ongelmia, sillä yumin kautta asennettu paketti ei vetänyt kaikkia riippuvuuksia perässään, siinä missä koko järjestelmän asentaminen kyllä asensi myös tarvittavat riippuvuudet. Oletuksena oli, että myös yksittäisen komponentin riippuvuudet olisivat asentuneet suoraan. Kaksivaiheisen todennuksen toteutuksessa suurin ongelma oli Palo Alton Service Routing –ominaisuus. Radius ja LDAP-liikenne kulkivat oletuksena hallintarajapinnan kautta,

joka taas ei ollut sisäverkossa. Liikenteen pakottaminen tietyn kohde IP:n perusteella toimi LDAP-liikenteelle, mutta ei taas jostain syystä Radius-liikenteelle. Vasta kun palvelut asetettiin palvelupohjaisesti kulkemaan tarvittavan rajapinnan läpi, myös Radius-yhteys toimi.

Lisäksi SSO-järjestelmän dokumentaatio osoittautui hyvin ympäripyöreäksi. Esimerkiksi Roundcubemailin dokumentoitu integrointi sisälsi kohdan ”Konfiguroi Apache”. Tarpeeksi suuri määrä yritystä ja erehdystä vei lopulta ratkaisun äärelle, mutta lisäsi työtaakkaa huomattavasti. Dokumentoinnin sokea seuraaminen ei ollut mahdollista niissäkään tilanteissa, jossa dokumentoinnin sisältö oli kattavampi. Esimerkiksi tietokannan luomista varten annetut valmiit taulut sisälsivät virheitä, mutta työn tekijän aikaisempi kokemus auttoi huomaamaan tämän suoraan.

9.2 Jatkokehitys

Itse komponentin toimintaan liittyviä parannuksia ja lisätestauksia on vielä suoritettavana työn valmistumisen jälkeenkin. Active Directoryn LDAP-liikenne ja MySQL-liikenne voidaan salata tietoturvan lisäämiseksi, kunhan tarvittava PKI-infrastruktuuri on olemassa. Lisäksi organisaatioympäristön palveluista valittiin vain yleisimmin käytetyt palvelut työssä toteutettaviksi. Uusien palvelujen integrointi onkin jatkuvaa kehitystä kertakirjautumispalvelulle.

Kerberos-protokollasta tulisi päästä myös eroon, jotta SSO voitaisiin toteuttaa mahdollisimman ympäristöriippumattomasti. LemonLDAP ei itse sisällä KDC-palvelinta, joten Kerberoksen poistaminen ympäristöstä ja työssä esitettyjen vaihtoehtojen ottaminen käyttöön palvelisi tavoitteita paremmin. Lisäksi palvelun klusterointi olisi mielenkiintoista testata. Jos myöhemmin osoittautuu, että Kerberos on ainoa ratkaisu SSO:n toteuttamiseen joissakin palveluissa, voidaan SSO-komponenttiin mahdollisesti sitoa vaikkapa työssä käytetty FreeIPA.

Kaksivaiheinen todennus keskittyi työssä lähinnä vain siihen, miten Palo Alton VPN-yhteyteen saadaan integroitua vahva käyttäjän tunnistus. Yksi jatkokehitysidea tälle on eri avoimeen lähdekoodiin pohjautuvien OTP-palvelujen tutkiminen ja liittäminen

Radius-palvelimeen. Näin RGCE-organisaatioympäristöihin saataisiin suoraan käyttöön kaksivaiheinen todennus, sillä Radiuksen kautta vahvan todennuksen saa sidottua myös kertakirjautumisjärjestelmään.

Myös itse työssä käytetty FreeIPA näytti hyvin monipuoliselta ja kätevältä IAM-palvelulta (Identity and Access Management), jolle löytyisi varmasti myös käyttöä ainakin vaihtoehtona muille RGCE:n identiteetin hallinta –palveluille. Varsinkin tuki OATH-standardoiduille 2FA-toteutuksille on erinomainen ominaisuus korkeamman turvallisuusluokituksen verkkoja mallintaessa.

Kertakirjautumisjärjestelmän tukemat ominaisuudet mahdollistavat sen käytön esimerkiksi yleisenä OpenID Connect Providerina, joten järjestelmällä voitaisiin toteuttaa RGCE-ympäristöön jokin kolmannen osapuolen IdP, mallintaen esimerkiksi accounts.google.comin toimintaa. Täten ympäristön julkisiin palveluihin voitaisiin tarjota käyttäjän tunnistaminen kolmannen osapuolen avulla. Tällaisen järjestelmän käyttö hyökkäystoiminnassa olisi hyödyllinen lisä RGCE-ympäristön harjoituksiin.

Harjoitustoimintaa ajatellen olisi mielenkiintoista testata, miten nyt toteutetun järjestelmän SSO-evästettä voitaisiin väärinkäyttää. Evästeen kaappaaminen mahdollistaa suuremman kokoluokan istunnonkaappaamisen. Lisäksi olisi tärkeää testata palvelun toimintaa niin, että SSO-tarjotaan vain sisäverkon pyynnöille. Tätä ei nyt tehdyssä työssä ole vielä huomioitu. Lisäksi, koska SSO-komponentti tuo verkkoon niin sanotun single point of failuren eli yksittäisen solmukohdan verkossa, joka kaatuessaan lamauttaa verkon palvelut, olisi oleellista testata myös suojauskeinot ja hyökkäyskeinot palvelunestoskenaarioita ajatellen.

Lähteet

CAS. N.d. CAS – Enterprise Single Sign-On for All. CAS Documentation. Viitattu 13.11.2016. https://apereo.github.io/cas/5.0.x/index.html

Duo – Features. 2016. Account and feature information. Viitattu 1.11.2016. https://duo.com/pricing

Duo – Palo Alto. 2016. Palo Alto GlobalProtect Integration Guide. Viitattu 1.11.2016. https://duo.com/docs/paloalto

FreeIPA – About. N.d. FreeIPA-järjestön verkkosivut. Viitattu 1.11.2016. https://www.freeipa.org/page/About

Garman, J. 2003. Kerberos: The Definitive Guide. Sebastopol: O'Reilly & Associates, Inc.

Google Authenticator. 2016. Google Authenticator OpenSource GitHub-sivusto. Viitattu 20.11.2016. https://github.com/google/google-authenticator

Hursti, J. 1997. Single Sign-On. Department of Computer Science, Helsinki University of Technology. Viitattu 12.11.2016. http://www.tml.tkk.fi/Opinnot/Tik-110.501/1997/single_sign-on.html

Jamsa, K. 2013. Cloud Computing. Viitattu 12.11.2016. http://www.jamk.fi/kirjasto, Oppaat, e-aineistot, Tietotekniikka, Books24x7.

JYVSECTEC – RGCE. 2016. JYVSECTEC:n verkkosivut. Viitattu 10.10.2016. http://jyvsectec.fi/fi/kyberymparisto/

JYVSECTEC – Tietoa meistä. 2016. JYVSECTEC:n verkkosivut. Viitattu 10.10.2016. http://jyvsectec.fi/fi/tietoa-meista/

Kerberos. 2009. How the Kerberos Version 5 Authentication Protocol Works. Microsoft TechNet -verkkosivut. Viitattu 3.11.2016. https://technet.microsoft.com/en-us/library/cc772815(v=ws.10).aspx

Killeen, E. 2012. What is federation? And how is it different from SSO? Viitattu 13.11.2016. http://blog.empowerid.com/blog-1/bid/164625/What-is-federation-And-how-is-it-different-from-SSO

LemonLDAP – AD. 2016. Cannot store Conf or Sessions in AD. Viitattu 11.10.2016. https://jira.ow2.org/browse/LEMONLDAP-759

LemonLDAP – Documentation. N. d. LemonLDAP documentation and wiki. Viitattu 15.11.2016. http://lemonldap-ng.org/documentation/1.9/start

LemonLDAP – OATH. 2016. OATH authentication module. Viitattu 20.11.2016. https://jira.ow2.org/browse/LEMONLDAP-786

LemonLDAP – Presentation. N.d. LemonLDAP documentation and wiki. Viitattu 10.10.2016. http://lemonldap-ng.org/documentation/presentation

LemonLDAP – References. N.d. LemonLDAP documentation and wiki. Viitattu 20.11.2016. http://lemonldap-ng.org/references

LemonLDAP – Start. N.d. LemonLDAP documentation and wiki. Viitattu 11.10.2016. http://lemonldap-ng.org/documentation/1.9/start

Mathers, B. 2016. What's new in Active Directory Federation Services for Windows Server 2016. Microsoft Technet –sivusto. Viitattu 14.11.2016. https://technet.microsoft.com/en-us/windows-server-docs/identity/ad-fs/overview/whats-new-active-directory-federation-services-windows-server-2016

OpenID Connect. 2014. OpenID Connect Core 1.0 incorporating errata set 1. Viitattu 14.11.2016. http://openid.net/specs/openid-connect-core-1_0.html#IDToken

Palo Alto – 2FA. N.d. GlobalProtect Administrator's Guide Version 6.0. Set up Two-Factor Authentication. Viitattu 22.11.2016. https://www.paloaltonetworks.com/documentation/60/globalprotect/global_protect_6-0/set-up-the-globalprotect-infrastructure/set-up-two-factor-authentication

RFC 3875. 2004. The Common Gateway Interface (CGI) Version 1.1. Viitattu 28.11.2016. https://tools.ietf.org/html/rfc3875

RFC 4120. 2005. The Kerberos Network Authentication Service (V5). Viitattu 11.10.2016. https://tools.ietf.org/html/rfc4120

RFC 6649. 2012. Deprecate DES, RC4-HMAC-EXP, and Other Weak Cryptographic Algorithms in Kerberos. Viitattu 11.10.2016. https://tools.ietf.org/html/rfc6649

RFC 7230. 2014. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. Viitattu 9.11.2016. https://tools.ietf.org/html/rfc7230

RFC7231. 2014. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. Viitattu 9.11.2016. https://tools.ietf.org/html/rfc7231

RFC 7235. 2014. Hypertext Transfer Protocol (HTTP/1.1): Authentication. Viitattu 10.10.2016. https://tools.ietf.org/html/rfc7235

Shibboleth. N.d. Tuotteen verkkosivut. Viitattu 13.11.2016. https://shibboleth.net/products/service-provider.html

Stanislav, M. 2015. Two-Factor Authentication. Viitattu 15.11.2016. http://www.jamk.fi/kirjasto, Oppaat, e-aineistot, Tietotekniikka, Books24x7.

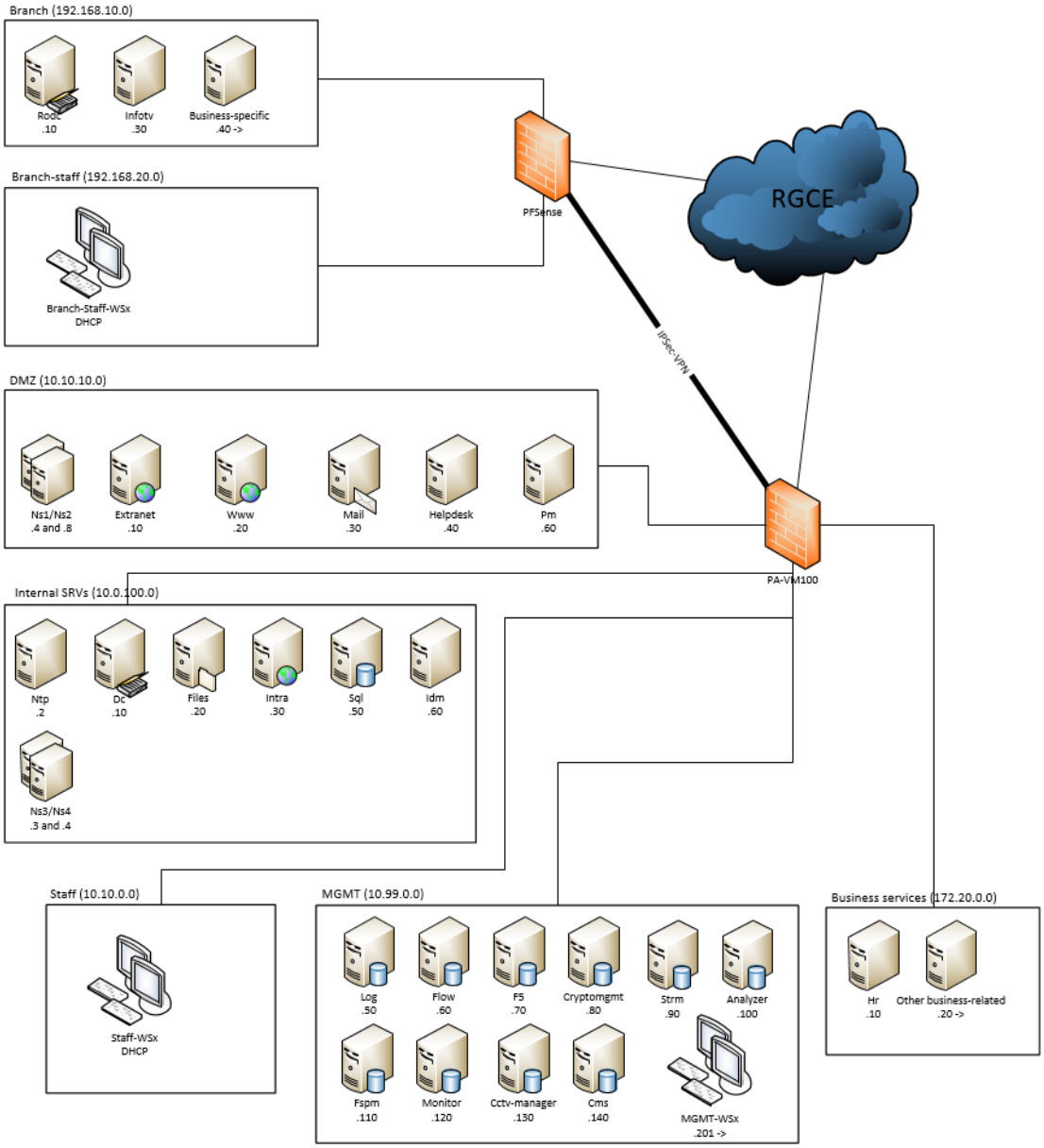
Tarkoma, S. & Kangasharju, J. 2009. Mobile Middleware: Supporting Applications and Services. Viitattu 13.11.2016. http://www.jamk.fi/kirjasto, Oppaat, e-aineistot, Tietotekniikka, Books24x7.

Tipton, F. & Krause, M. 2007. Information Security Management Handbook – Volume 1. 6. p. Viitattu 12.11.2016. http://www.jamk.fi/kirjasto, Oppaat, e-aineistot, Tietotekniikka, Books24x7.

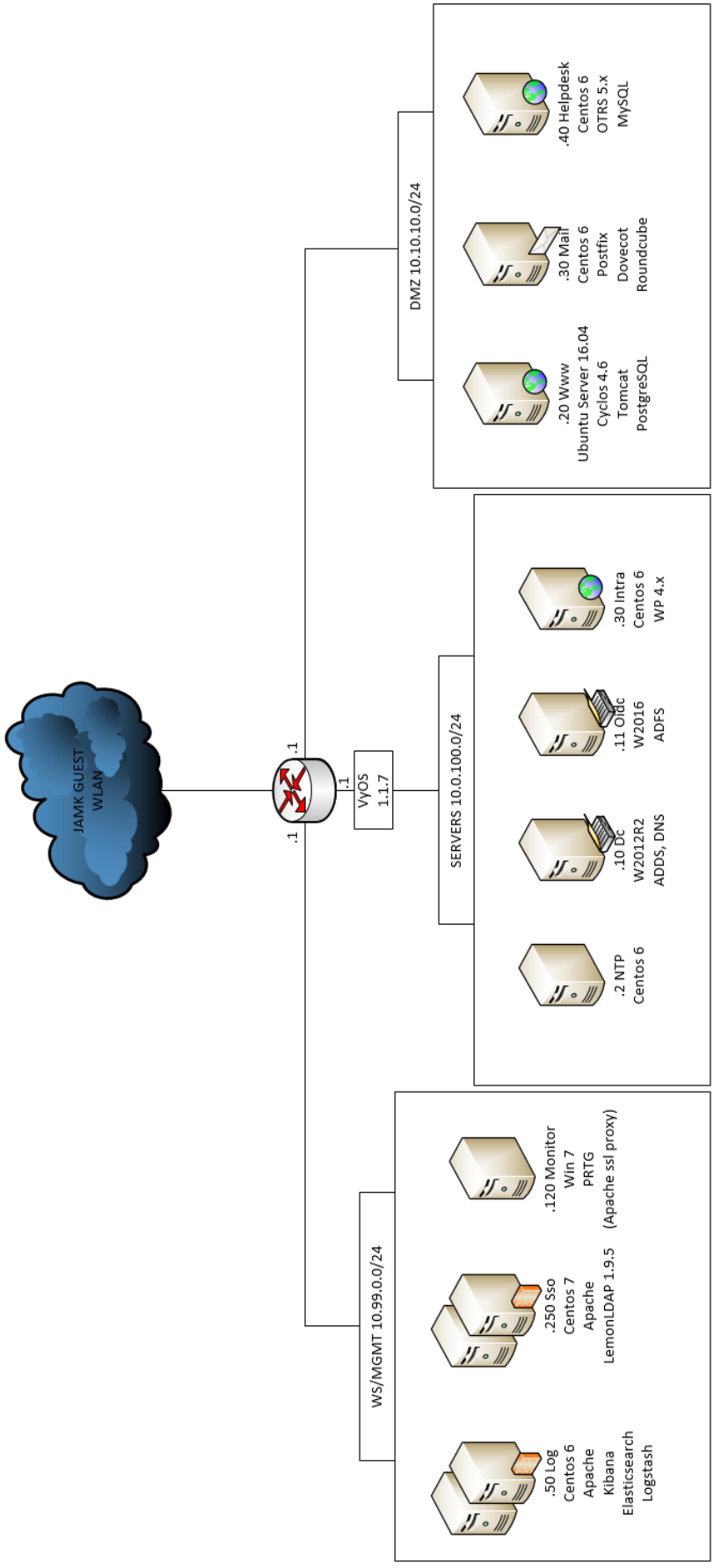
Understanding HTTP Authentication. N.d. Understanding HTTP Authentication. Microsoftin tietosivu. Viitattu 11.10.2016. https://msdn.microsoft.com/en-us/library/ms789031(v=vs.110).aspx

Liitteet

Liite 1. Organisaatiomalli



Liite 2. Testiympäristö



Liite 3. LemonLDAP templaatin asennusskripti

```
#!/bin/bash
# Argument = -d dbname -u dbuser -s dbhostserver
usage()
{
cat << EOF
usage: $0 options

This script converts lemonldap config backend to SQL.

OPTIONS:
    -h      Show this message
    -d      Domain ie. example.com
    -n      Database name
    -u      Database user
    -s      ( Optional ) Database FQDN or IP ( defaults to local db
if -s flag is not used )
EOF
}

# Variables needed
DBNAME=
DBUSER=
DBPASS=$(date +%s%N | sha512sum | base64 | head -c 8; echo)
DBHOST=
DOMAIN=

# Parse command line
while getopts "hd:n:u:s:" OPTION
do
    case $OPTION in
        h)
            usage
            exit 1
            ;;
        d)
            DBNAME=$OPTARG
            ;;
        n)
            DOMAIN=$OPTARG
            ;;
        u)
            DBUSER=$OPTARG
            ;;
        s)
            DBHOST=$OPTARG
            ;;
        ?)
            usage
            exit
            ;;
    esac
done

# Needed variables
if [[ -z $DBNAME ]] || [[ -z $DBUSER || [[ -z $DOMAIN ]]; then
    usage
    exit 1
fi
hostnamectl set-hostname sso."$DOMAIN"
rm -f /root/templatize.sh
```

```
domain=$(hostname -d | cut -d "." -f1)
tld=$(hostname -d | cut -d "." -f2)
host=$(hostname -f)
# Add check for DBHOST != localhost in any form
userip=$DBHOST
if [[ $DBHOST == "localhost" ]]; then
    echo "Only use -s flag with a remote DB -> Remote handlers won't
work with a local session database. Bind to FQDN or IP."
    exit 1
fi
if [[ $userip =~ ^^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$
]]; then
    OIFS=$IFS
    IFS='.'
    userip=($userip)
    IFS=$OIFS
    if [[ ${userip[0]} -le 255  && ${userip[1]} -le 255 &&
${userip[2]} -le 255 && ${userip[3]} -le 255 ]]; then
        if [[ ${userip[0]} -eq 127 ]]; then
            echo "Do not use -s flag for a local database"
            exit 1
            fi
        else
            echo "Invalid IP address"
            exit 1
    fi
fi

# Prerequisites
# Self-signed cert
echo "Creating self-signed cert"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/certs/sso.key -out /etc/ssl/certs/sso.crt -subj "/O=${do-
main^}/OU=IT/CN=*.$domain.$tld" > /dev/null 2>&1
# Change example domain
sed -i "s/example\.com/$domain.$tld/g" /etc/hosts
sed -i "s/example\.com/$domain.$tld/g" /root/lemonldap-ng.ini.template
sed -i "s/example\.com/$domain.$tld/g" /etc/lemonldap-ng/*
sed -i "s/example\.com/$domain.$tld/g" /var/lib/lemonldap-ng/conf/*
sed -i "s/example\.com/$domain.$tld/g" /var/lib/lemonldap-ng/test/in-
dex.pl
sed -i "s/example\.com/$domain.$tld/g" /etc/httpd/conf.d/*

if [[ $DBHOST == "" ]]; then
    DBHOST="$host"
fi

if [[ $DBHOST == "$host" ]]; then
    echo "Creating local database"
    systemctl enable mariadb > /dev/null 2>&1
    systemctl start mariadb > /dev/null 2>&1

    # DB information -> /root/lemonldap-ng.ini
    sed "s/DBNAME/$DBNAME/" lemonldap-ng.ini.template > /root/lem-
onldap-ng.ini
    sed -i "s/DBHOST/$DBHOST/" /root/lemonldap-ng.ini
    sed -i "s/DBUSER/$DBUSER/" /root/lemonldap-ng.ini
    sed -i "s/DBPASS/$DBPASS/" /root/lemonldap-ng.ini

    # Create local DB
    sed "s/DBNAME/$DBNAME/" lemon.sql.template > /root/lemon.sql
    sed -i "s/DBUSER/$DBUSER/" /root/lemon.sql
```

```
    sed -i "s/DBPASS/$DBPASS/" /root/lemon.sql
    mysql -u root -proot66 < lemon.sql

    # Firewalld
    firewall-cmd --permanent --add-service=mysql > /dev/null 2>&1
    firewall-cmd --reload > /dev/null 2>&1

else
    echo "Creating remote database"

    # DB information -> /root/lemonldap-ng.ini
    sed "s/DBNAME/$DBNAME/" lemonldap-ng.ini.template > /root/lem-
onldap-ng.ini
    sed -i "s/DBHOST/$DBHOST/" /root/lemonldap-ng.ini
    sed -i "s/DBUSER/$DBUSER/" /root/lemonldap-ng.ini
    sed -i "s/DBPASS/$DBPASS/" /root/lemonldap-ng.ini

    # Create remote DB
    sed "s/DBNAME/$DBNAME/" lemon.sql.template > /root/lemon.sql
    sed -i "s/DBUSER/$DBUSER/" /root/lemon.sql
    sed -i "s/DBPASS/$DBPASS/" /root/lemon.sql

    # User Provision account
    mysql -u Provision -pProvision -h $DBHOST < lemon.sql

    # Firewalld
    if [[ $(firewall-cmd --list-all | grep mysql) != "" ]]; then
        firewall-cmd --permanent --remove-service=mysql > /dev/null
2>&1
        firewall-cmd --reload > /dev/null 2>&1
    fi
fi

# Convert config from default file to SQL
echo "Enabling MySQL configuration backend"
/usr/share/lemonldap-ng/bin/convertConfig -l --current=/etc/lem-
onldap-ng/lemonldap-ng.ini --new=/root/lemonldap-ng.ini > /dev/null
2>&1
mv /etc/lemonldap-ng/lemonldap-ng.ini /etc/lemonldap-ng/lemonldap-
ng.ini.old
cp /root/lemonldap-ng.ini /etc/lemonldap-ng/lemonldap-ng.ini

# Turn off local db if not needed
if [[ $DBHOST != "$host" ]]; then
    echo "Turning off local database"
    systemctl disable mariadb > /dev/null 2>&1
    systemctl stop mariadb > /dev/null 2>&1
fi

echo "Enabling MySQL session backend"
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "globalStor-
age" "Apache::Session::MySQL" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "DataSource" "DBI:mysql:database=$DBNAME;host=$DBHOST" >
/dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "UserName" "$DBUSER" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "Password" "$DBPASS" > /dev/null 2>&1
```

```
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "LockDataSource"
"DBI:mysql:database=$DBNAME;host=$DBHOST" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "LockUserName" "$DBUSER" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "LockPassword" "$DBPASS" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "globalStor-
ageOptions" "TableName" "sessions" > /dev/null 2>&1

echo "Enabling Active Directory user backend"
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "authentica-
tion" "AD" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "passwordDB"
"AD" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "userDB" "AD" >
/dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "ldapServer"
"ldap://dc.$domain.$tld" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "ldapPort"
"389" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "ldapBase"
"ou=EndUsers,ou=$domain.$tld,dc=$domain,dc=$tld" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "managerPass-
word" "root66" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "managerDn"
"cn=Administrator,cn=Users,dc=$domain,dc=$tld" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "ldapTimeout"
"120" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "ldapVersion"
"3" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "AuthLDAPFil-
ter" "(&(sAMAccountName=\$user)(objectClass=person))" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "mailLDAPFil-
ter" "(&(mail=\$mail)(objectClass=person))" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli -yes 1 set "storePass-
word" "1" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "ldapExport-
edVars" "uid" "sAMAccountName" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "ldapExport-
edVars" "cn" "cn" > /dev/null 2>&1
/usr/share/lemonldap-ng/bin/lemonldap-ng-cli addKey "ldapExport-
edVars" "mail" "mail" > /dev/null 2>&1

systemctl restart httpd > /dev/null 2>&1

echo "Your DB user is $DBUSER and account password is $DBPASS"

exit 0
```

Liite 4. OTRS Config.pm

```
### SSO
    $Self->{'AuthModule'} = 'Kernel::System::Auth::HTTPBasicAuth';
    $Self->{'AuthModule::HTTPBasicAuth::Replace'} = 'example.com\\';
    $Self->{'AuthModule::HTTPBasicAuth::ReplaceRegExp'} =
'^(.+?)@.+?$';

### DB
    $Self->{AuthModule1} = 'Kernel::System::Auth::DB';

### LDAP Agents
    $Self->{'AuthModule2'} = 'Kernel::System::Auth::LDAP';
    $Self->{'AuthModule::LDAP::Host2'} = 'dc.example.com';
    $Self->{'AuthModule::LDAP::BaseDN2'} = 'ou=example.com,dc=exam-
ple,dc=com';
    $Self->{'AuthModule::LDAP::SSCOPE2'} = 'sub';
    $Self->{'AuthModule::LDAP::UID2'} = 'sAMAccountName';

    # Check if the user is allowed to auth in a posixGroup
    # (e. g. user needs to be in a group Administrators to use otrs)
        # Change as needed
    $Self->{'AuthModule::LDAP::GroupDN2'} =
'CN=Administrators,CN=Builtin,DC=example,DC=com';
    $Self->{'AuthModule::LDAP::AccessAttr2'} = 'member';
    $Self->{'AuthModule::LDAP::UserAttr2'} = 'DN';

    # Bind credentials to log into AD
        # Change to proper binduser
    $Self->{'AuthModule::LDAP::SearchUserDN2'} =
'CN=otrs,OU=SystemUsers,OU=example.com,DC=example,DC=com';
    $Self->{'AuthModule::LDAP::SearchUserPw2'} = '********';

    # in case you want to add always one filter to each ldap query,
use
    # this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter
=> '(objectclass=user)'
    $Self->{'AuthModule::LDAP::AlwaysFilter2'} = '';

    # in case you want to add a suffix to each login name,  then
    # you can use this option. e. g. user just want to use user but
    # in your ldap directory exists user@domain.
    #$Self->{'AuthModule::LDAP::UserSuffix2'} = '';

    # Net::LDAP new params (if needed - for more info see perldoc
Net::LDAP)
    $Self->{'AuthModule::LDAP::Params2'} = {
        port => 389,
        timeout => 120,
        async => 0,
        version => 3,
    };

    # Now sync data with OTRS DB
    $Self->{'AuthSyncModule2'} = 'Kernel::System::Auth::Sync::LDAP';
    $Self->{'AuthSyncModule::LDAP::Host2'} = 'dc.example.com';
    $Self->{'AuthSyncModule::LDAP::BaseDN2'} = 'ou=example.com,dc=ex-
ample,dc=com';
    $Self->{'AuthSyncModule::LDAP::SSCOPE2'} = 'sub';
    $Self->{'AuthSyncModule::LDAP::UID2'} = 'sAMAccountName';
    # Change to proper binduser
        $Self->{'AuthSyncModule::LDAP::SearchUserDN2'} =
'CN=otrs,OU=SystemUsers,OU=example.com,DC=example,DC=com';
```

```
    $Self->{'AuthSyncModule::LDAP::SearchUserPw2'} = '********';

    $Self->{'AuthSyncModule::LDAP::UserSyncMap2'} = {
        # DB -> LDAP
        UserFirstname => 'givenName',
        UserLastname  => 'sn',
        UserEmail     => 'mail',
    };

    # AuthSyncModule::LDAP::UserSyncInitialGroups
    # (sync following group with rw permission after initial create
of first agent login)
    $Self->{'AuthSyncModule::LDAP::UserSyncInitialGroups2'} = [
                'admin',
                'stats',
                'users',
    ];
 #AuthSyncModule::LDAP::UserSyncGroupsDefinition
# (If "LDAP" was selected for AuthModule and you want to sync LDAP
# groups to otrs groups, define the following.)
        $Self->{'AuthSyncModule::LDAP::UserSyncGroupsDefinition2'} =
{
    # your ldap group
    'cn=Administrators,cn=Builtin,dc=example,dc=com' => {
        # otrs group(s)
        'admin' => {
            # permission
            rw => 1,
            ro => 1,
        },
        'stats' => {
            rw => 1,
            ro => 1,
        },
        'users' => {
            rw => 1,
            ro => 1,
        },
    }
};
#-------------------------------------------------------------------
-------------------------#
#LDAP Customers
  $Self->{'Customer::AuthModule1'} = 'Kernel::System::Custom-
erAuth::LDAP';
  $Self->{'Customer::AuthModule::LDAP::Host1'} = 'dc.example.com';
  $Self->{'Customer::AuthModule::LDAP::BaseDN1'} = 'ou=EndUs-
ers,ou=example.com,dc=example,dc=com';
  $Self->{'Customer::AuthModule::LDAP::UID1'} = 'sAMAccountName';

#Add the following lines when users are allowed to login only if they
reside in the specified security group
#Remove these lines if you want to provide login to all users speci-
fied in the User Base DN
#example: $Self->{'Customer::AuthModule::LDAP::BaseDN'} = 'ou=BaseOU,
dc=example, dc=com';
#  $Self->{'Customer::AuthModule::LDAP::GroupDN1'} =
'CN=group,OU=ou,DC=example,DC=com';
#  $Self->{'Customer::AuthModule::LDAP::AccessAttr1'} = 'member';
#  $Self->{'Customer::AuthModule::LDAP::UserAttr1'} = 'DN';

#The following is valid but would only be necessary if the anonymous
user do NOT have permission to read from the LDAP tree
```

```
  # Change to proper binduser
  $Self->{'Customer::AuthModule::LDAP::SearchUserDN1'} =
'CN=otrs,OU=SystemUsers,OU=example.com,DC=example,DC=com';
  $Self->{'Customer::AuthModule::LDAP::SearchUserPw1'} = '********';

#Filter ie. '(mail=*)'
  $Self->{'Customer::Authmodule::LDAP::AlwaysFilter1'} = '';

#Customer LDAP connection params
  $Self->{'Customer::Authmodule::LDAP::Params1'} = {
        port => 389,
        timeout => 120,
        async => 0,
        version => 3,
  };

#CustomerUser
#(customer user database backend and settings)
    $Self->{CustomerUser1} = {
      Module => 'Kernel::System::CustomerUser::LDAP',
      Params => {
      Host => 'dc.example.com',
      BaseDN => 'OU=EndUsers,OU=example.com,DC=example,DC=com',
      SSCOPE => 'sub',
          # Change to proper binduser
      UserDN =>
'CN=otrs,OU=SystemUsers,OU=example.com,DC=example,DC=com',
      UserPw => '********',
    },
# customer unique id
    CustomerKey => 'sAMAccountName',
    # customer #
    CustomerID => 'mail',
    CustomerUserListFields => ['sAMAccountName', 'cn', 'mail'],
    CustomerUserSearchFields => ['sAMAccountName', 'cn', 'mail'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['mail'],
    CustomerUserNameFields => ['givenname', 'sn'],
    Map => [
      # note: Login, Email and CustomerID needed!
      # var, frontend, storage, shown, required, storage-type
      #[ 'UserSalutation', 'Title', 'title', 1, 0, 'var' ],
      [ 'UserFirstname', 'Firstname', 'givenname', 1, 1, 'var' ],
      [ 'UserLastname', 'Lastname', 'sn', 1, 1, 'var' ],
      [ 'UserLogin', 'Login', 'sAMAccountName', 1, 1, 'var' ],
      [ 'UserEmail', 'Email', 'mail', 1, 1, 'var' ],
      [ 'UserCustomerID', 'CustomerID', 'mail', 0, 1, 'var' ],
      [ 'UserPhone', 'Phone', 'telephonenumber', 1, 0, 'var' ],
      #[ 'UserAddress', 'Address', 'postaladdress', 1, 0, 'var' ],
      #[ 'UserComment', 'Comment', 'description', 1, 0, 'var' ],
    ],
  };
#-------------------------------------------------------------------
-------------------------#

### DB customers (external) double check smtp settings for validation
and registration
    # (customer user database backend and settings)
    $Self->{CustomerUser2} = {
        Name => 'Database Datasource',
        Module => 'Kernel::System::CustomerUser::DB',
```

```
        Params => {
            # if you want to use an external database, add the
            # required settings (or map external db in db settings)
#            DSN => 'DBI:odbc:yourdsn',
#            DSN =>
'DBI:mysql:database=customerdb;host=customerdbhost',
#            User => '',
#            Password => '',
            Table => 'customer_user',
        },
        # customer unique id
    CustomerKey => 'login',
        # customer #
    CustomerID => 'customer_id',
    CustomerValid => 'valid_id',
        CustomerUserListFields => ['first_name', 'last_name',
'email'],
        CustomerUserSearchFields => ['login', 'last_name', 'cus-
tomer_id'],
        CustomerUserSearchPrefix => '',
        CustomerUserSearchSuffix => '*',
        CustomerUserSearchListLimit => 250,
        CustomerUserPostMasterSearchFields => ['email'],
        CustomerUserNameFields => ['saluta-
tion','first_name','last_name'],
        CustomerUserEmailUniqCheck => 1,
#        # show not own tickets in customer panel, CompanyTickets
#        CustomerUserExcludePrimaryCustomerID => 0,
#        # generate auto logins
#        AutoLoginCreation => 0,
#        AutoLoginCreationPrefix => 'auto',
#        # admin can change customer preferences
#        AdminSetPreferences => 1,
#        # cache time to live in sec. - cache any database queries
#        CacheTTL => 0,
#        # just a read only source
#        ReadOnly => 1,
        Map => [
            # note: Login, Email and CustomerID needed!
            # var, frontend, storage, shown (1=always,2=lite), re-
quired, storage-type, http-link, readonly, http-link-target
 [ 'UserSalutation', 'Salutation', 'salutation', 1, 0, 'var', '', 0
],
 [ 'UserFirstname',  'Firstname',  'first_name', 1, 1, 'var', '', 0
],
 [ 'UserLastname',   'Lastname',   'last_name',  1, 1, 'var', '', 0
],
 [ 'UserLogin',      'Username',   'login',      1, 1, 'var', '', 0
],
 [ 'UserPassword',   'Password',   'pw',         0, 0, 'var', '', 0
],
 [ 'UserEmail',      'Email',      'email',      1, 1, 'var', '', 0
],

 [ 'UserEmail',      'Email', 'email',          1, 1, 'var',
'$Env{"CGIHandle"}?Action=AgentTicketCompose&ResponseID=1&Tick-
etID=$Data{"TicketID"}&ArticleID=$Data{"ArticleID"}', 0 ],
 [ 'UserCustomerID', 'CustomerID', 'customer_id', 0, 1, 'var', '', 0
],

 [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0, 'var', '',
0 ],
```

```
 [ 'UserPhone',        'Phone',       'phone',        1, 0, 'var',
'', 0 ],
 [ 'UserFax',          'Fax',         'fax',          1, 0, 'var',
'', 0 ],
 [ 'UserMobile',       'Mobile',      'mobile',       1, 0, 'var',
'', 0 ],
 [ 'UserStreet',       'Street',      'street',       1, 0, 'var',
'', 0 ],
 [ 'UserZip',          'Zip',         'zip',          1, 0, 'var',
'', 0 ],
 [ 'UserCity',         'City',        'city',         1, 0, 'var',
'', 0 ],
 [ 'UserCountry',      'Country',     'country',      1, 0, 'var',
'', 0 ],
 [ 'UserComment',      'Comment',     'comments',     1, 0, 'var',
'', 0 ],
 [ 'ValidID',          'Valid',       'valid_id',     0, 1, 'int',
'', 0 ],
],
        # default selections
        Selections => {
            UserSalutation => {
                'Mr.' => 'Mr.',
                'Mrs.' => 'Mrs.',
            },
        },
    };
```

Liite 5. Mod_auth_krb –mallipohja

```
#<VirtualHost *:80>
ServerName FQDN
#LogLevel debug
#RewriteEngine on
#RewriteRule ^/$ https://FQDN/
#</VirtualHost>

#<VirtualHost *:443>
ServerName FQDN
#LogLevel debug
<Directory /some/protected/directory>
#   <Files someprotectedfile.?>
# Kerberos SSO
       AuthName "Kerberos AUTH"
       AuthType Kerberos
       KrbServiceName SERVICE
       KrbAuthRealms DOMAINUC
       Krb5Keytab /etc/httpd/KEYTAB
       KrbMethodNegotiate On
       KrbMethodK5Passwd Off
# Provide SSO for everyone
      require valid-user
# LDAP Auth
# Provide SSO for designated users only
#AuthzLDAPAuthoritative on
#AuthBasicProvider ldap
#AuthLDAPBindDN binduser@domain.tld
#AuthLDAPBindPassword “Pass”
#AuthLDAPURL
“ldap://ipfqdn:port/OU=searchbase,DC=domain,DC=tld?userPrincipalName?
sub?(objectClass=*)”
#Require ldap-group CN=Admin,DC=domain,DC=tld
#      require valid-user
#   </Files>
</Directory>
#</VirtualHost>
```

Liite 6. Skripti Kerberos-palvelun toteuttamiseen

```
# Variables
# User can exist? Mapping multiple SPNs to single user
Param(
        [string]$user,
        [string]$passwd,
        [string]$server,
        [string]$type
)

# Required parameters check
if ( !$user -or !$passwd -or !$server -or !$type ) {
        echo ""
        echo "Usage ./script -user <username> -passwd <user pass> -
server <remote server fqdn> -type <service type>"
        echo ""
        exit 1
}

# Get domain name and distingquished name
$domain = Get-ADDomain | FL DNSRoot | Out-String | %{ "{0}" -f
$_.Split(':')[1].Trim() };
$dn = Get-ADDomain | FL DistinguishedName | Out-String | %{ "{0}" -f
$_.Split(':')[1].Trim() };
$type = $($type.ToUpper())

# Create serviceuser, Check if exists
$check = $(try { Get-ADUser -Identity kerberos_$user } catch { $Null
})
if ( $check -eq $Null ) {
        # Create serviceuser
        New-ADuser -SamAccountName kerberos_$user -UserPrincipalName
kerberos_$user@$domain -GivenName kerberos_$user -DisplayName ker-
beros_$user -Name kerberos_$user -AccountPassword(ConvertTo-Secur-
eString $passwd -AsPlainText -Force) -PasswordNeverExpires:$True -En-
abled:$True -Path "OU=SystemUsers,OU=$domain,$dn"
} else {
        # Check for existing SPN/user if found abort?
        $counter = $( setspn -U -l kerberos_$user | measure ).Count
        # Modify to check if duplicate SPNs would be created, allow
if not?
        if ( $counter -gt 1 ) {
                echo "User has pre-existing SPN mapping:"
                setspn -U -l kerberos_$user | Select -Skip 1
                echo "Exiting"
                echo ""
                exit 1
        }
}

# Assign service principal names
setspn -A host/$server kerberos_$user
setspn -A $type/$server kerberos_$user

# Create keytabs
ktpass -princ host/$server@$($domain.ToUpper()) -crypto AES256-SHA1 -
pass $passwd -mapuser kerberos_$user -pType KRB5_NT_PRINCIPAL -out
c:\$user.host.keytab
ktpass -princ $type/$server@$($domain.ToUpper()) -crypto AES256-SHA1
-pass $passwd -mapuser kerberos_$user -pType KRB5_NT_PRINCIPAL -out
c:\$user.$type.keytab
```

```
# Echo info
echo ""
echo "User kerberos_$user created and mapped to following SPNs:"
setspn -U -l kerberos_$user | Select -Skip 1
echo "Keytabs created and found in C:\$user.host.keytab and
C:\$user.$type.keytab"
echo "To undo, run C:\DeleteKerberosService.ps1 -user $user -server
$server -type $type"
echo ""
```

Liite 7. Skripti Kerberospalvelun poistamiseen

```
# Variables
Param(
        [string]$user,
        [string]$server,
        [string]$type
)

# Required parameters check
if ( !$user -or !$server -or !$type ) {
        echo ""
        echo "Usage ./script -user <username> -server <remote server
fqdn> -type <service type>"
        echo ""
        exit 1
}

# Check if exists
$check = $(try { Get-ADUser -Identity kerberos_$user } catch { $Null
})
if ( $check -eq $Null) {
        echo "User kerberos_$user doesn't exist, check for typos"
        exit 1
}

# Remove service user account?
do { $delusr = Read-Host "Delete service user account? Yes/No" }
until ("Yes","yes","No","no" -ccontains $delusr)
# Remove keytabs?
do { $deltab = Read-Host "Delete existing keytabs? Yes/No" }
until ("Yes","yes","No","no" -ccontains $deltab)

# Get domain name and distingquished name
#$domain = Get-ADDomain | FL DNSRoot | Out-String | %{ "{0}" -f
$_.Split(':')[1].Trim() };
#$dn = Get-ADDomain | FL DistinguishedName | Out-String | %{ "{0}" -f
$_.Split(':')[1].Trim() };
$type = $($type.ToUpper())

# Unregister service principal names
setspn -D host/$server kerberos_$user
setspn -D $type/$server kerberos_$user

# Delete serviceuser
if ( $delusr -eq "Yes" -or $delusr -eq "yes" ) {
        Remove-ADUser -Identity kerberos_$user -Confirm:$False
}

if ( $deltab -eq "Yes" -or $deltab -eq "yes" ) {
        # Remove keytabs
        rm C:\$user.*.keytab
}
```

Liite 8. Bash skripti kerberoitavaan kohdepalvelimeen

```
#!/bin/bash

SERVICE=$1
LOCATION=$2
# Usageinfo
if [[ $SERVICE == "" || $LOCATION == "" ]]; then
        echo ""
        echo "Usage ./script.sh SERVICETYPE KEYTABPATH"
        echo "Example ./kerberos.sh HTTP /tmp"
        echo ""
        exit 1
fi
# Checking if needed packages are installed, if not, installs them
if [[ $(yum list installed | grep "krb5-workstation") == "" ]]; then
        yum install -y krb5-workstation
fi

if [[ $(yum list installed | grep "mod_auth_kerb") == "" ]]; then
        yum install -y mod_auth_kerb
fi

if [[ $(yum list installed | grep "mod_authz_ldap") == "" ]]; then
        yum install -y mod_authz_ldap
fi

DOMAIN=$(hostname -d)
# Checking if domain name exists
if [[ $DOMAIN == "" ]]; then
        echo ""
        echo "Assign hostname -> /etc/hosts"
        echo ""
        exit 1
fi

# Checking if keytabs exist
if [[ $(ls $LOCATION/*.keytab)  == "" ]]; then
        echo ""
        echo "Retrieve keytabs or check path"
        echo ""
        exit 1
fi

# Domain name in NetBIOS format
DOMAINUC=$(echo $DOMAIN | awk '{print toupper($0)}')
DOMAINLC=$DOMAIN
# Populate krb5.conf
sed s/DOMAINUC/$DOMAINUC/ krb5.conf.template > /etc/krb5.conf
sed -i s/DOMAINLC/$DOMAINLC/ /etc/krb5.conf

# Process keytabs
cp $LOCATION/*.*.keytab /etc/httpd
keytabs=()
for i in $(ls /etc/httpd/*.*.keytab); do
        keytabs+=($i)
done

tab1=${keytabs[0]}
tab2=${keytabs[1]}
host=$(hostname -s)
# Merge keytab
ktutil <<EOF
```

```
rkt $tab1
rkt $tab2
wkt /etc/httpd/$host.keytab
q
EOF

chown apache:apache /etc/httpd/$host.keytab

# If conf vs htaccess
# KeepAlive on UseCanonicalName on ServerName FQDN
# if conf
# Create auth conf template

read -r -p "Create $host.conf template, .htaccess template or both?
[cnf/hta/bth] " response
response=${response,,}
while [[ ! $response =~ ^(cnf|c|hta|bth|b)$ ]]; do
        read -r -p "Create $host.conf template, .htaccess template or
both? [cnf/hta/bth] " response
        response=${response,,}
done
        if [[ $response =~ ^(cnf|c)$ || $response =~ ^(bth|b)$ ]];
then
        sed s/SERVICE/$SERVICE/ service.conf.template >
/etc/httpd/conf.d/$host.conf
        sed -i s/DOMAINUC/$DOMAINUC/ /etc/httpd/conf.d/$host.conf
        sed -i s/KEYTAB/$host.keytab/ /etc/httpd/conf.d/$host.conf
        sed -i s/FQDN/$(hostname -f)/ /etc/httpd/conf.d/$host.conf
        echo "/etc/httpd/conf.d/$host.conf created"
fi
# if htaccess
# Create htaccess template
if [[ $response =~ ^(hta|a)$ || $response =~ ^(bth|b)$ ]]; then
        sed s/SERVICE/$SERVICE/ service.htaccess.template >
/root/.htaccess
        sed -i s/DOMAINUC/$DOMAINUC/ /root/.htaccess
        sed -i s/KEYTAB/$host.keytab/ /root/.htaccess
        echo "/root/.htaccess created"
fi
cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.old
sed -i 's/KeepAlive Off/KeepAlive On/g' /etc/httpd/conf/httpd.conf
sed -i 's/UseCanonicalName Off/UseCanonicalName On/g'
/etc/httpd/conf/httpd.conf
echo "ServerName $host.$DOMAIN" >> /etc/httpd/conf/httpd.conf
service httpd restart
```