Samuli Nevalainen

IBM Security Access Managerin automatisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

22.4.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Samuli Nevalainen IBM Security Access Managerin automatisointi 45 sivua + 2 liitettä 22.4.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Juha Kämäri

Insinöörityössä tutustuttiin pääsynhallintaan ja miten IBM Security Access Manager -ohjelma auttaa sen toteuttamisessa. Insinöörityön tarkoituksena oli tutustua IBM Security Access Manager -ohjelmaan ja perehtyä sen automatisointiin ja tämän tuomiin hyötyihin.

IBM Security Access Manager on IBM:n kehittämä ohjelma, joka auttaa yrityksiä keskittämään pääsynhallinnan yhteen helposti käytettävään ohjelmaan. Ohjelmasta löytyy erilaisia tapoja toteuttaa pääsynhallintaa, kuten käyttöliittymän kautta, komentoriviltä, REST-rajapintoja käyttäen JSON-muotoisen datan käsittelyä tai CARA:n avulla XML-muotoisella tiedolla.

Työn teoriaosuudessa syvennytään XML- ja JSON-muotoisen datan hyödyntämiseen ISAM-automaation kehittämisessä sekä vertaillaan, miten tämä eroaa käyttöliittymän kautta tehtäviin asioihin. Käytännön osuudessa kuvataan, miten automaatioskriptejä voidaan luoda ja käyttää hyödyksi.

Lopputuloksena syntyi haluttuja automaatioskriptejä, joita voidaan käyttää uusien instanssien tai pienempien osien luomiseen tai päivittämiseen. Skriptejä on tarkoitus päivittää sitä mukaa kun niitä käytetään, jotta ne pysyvät aina ajan tasalla.

Avainsanat	IBM Security Access Manager, ISAM, Automatisointi

Author(s) Title Number of Pages Date	Samuli Nevalainen Automation of IBM Security Access Manager 45 pages + 2 appendices 22 April 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Juha Kämäri

This thesis focuses on access management with IBM Security Access Manager. Different approaches to achieve the same goal is also covered. The goal of the thesis was to learn more about automation and how to create automation for ISAM.

IBM Security Access Manager is a software that helps companies to focus on access management with one software. Software provides different methods to create access management, like with UI, Command-line Interface, using REST-interface with data in JSON format or with CARA using XML formatted data.

The theory section concentrates on the UI based creating and how do we use XML and JSON format data to create automation. The practical section describes how to create automation scripts and how to use them.

As the result of the thesis was born automation scripts for both XML and JSON. These can be used to create new instances or modify the existing ones. Scripts will be updated if the requirements change so they will always be up to date.

Keywords	IBM Security Access Manager, ISAM, Automation

Sisällys

Lyhenteet

ISAM	IBM Security Access Manager -ohjelma, joka tarjoaa monia mahdollisuuksia pääsynhallinnan ratkaisuihin.
SAM	Security Access Manager. Tuote jonka avulla tuotetaan pääsynhallintaa.
XML	Extensible Markup Language. Käytetään ilmoittamaan tietoa ihmisen ja koneen ymmärtävänä kielenä.
JSON	JavaScript Object Notation, jonka avulla voidaan siirtää dataobjekteja attribuutti arvo -pareina.
ACL	Access control list. Käytetään SAM policy -palvelimella määrittelemään oikeuksia tiettyyn objektiin.
POP	Protected object policy. Käytetään SAM policy -palvelimella määrittelemään käytäntöjä, jotka sisältävät ylimääräisiä ehtoja pyynnöille, jotka menevät SAM:lle.
AuthzRule	Authorization rule. Käytetään SAM policy -palvelimella määrittelemään turvallisuus sääntöjä, joiden avulla hallitaan objektiin liitettyjä käytäntöjä.
EAI	External Authentication Interface. Ulkoinen rajapinta, jonka avulla voidaan tunnistautua yksityisessä etäpalvelussa.
TFIM	Tivoli Federated Identity Manager. Sovellus joka mahdollistaa yhden kirjautumisen monista ohjelmista.
CARA	CA Release Automation. Tuote jonka avulla voidaan tehdä automaatiota eri ohjelmille.
RA	Release Automation. Prosessi joka kokoaa ja asentaa ohjelmiston.

1 Johdanto

Tietoturva sekä pääsynhallinta ovat nousseet nykyään isoksi asiaksi yrityksissä. Yhä useammat yritykset ovat huomanneet sijoittavansa tietoturvaan sekä pääsynhallintaan yhä enemmän rahaa. Yrityksillä on nykyään paljon tietoja käyttäjistään, ja sitä varten tietoturvan sekä pääsynhallinnan täytyy olla kunnossa. Liitteen 1 kuvasta nähdään, kuinka tietoturvaan sijoitetaan entistä enemmän rahaa, sekä miten tulevaisuudessa siihen sijoitetaan vieläkin enemmän.

Automaatiolla tarkoitetaan ohjelmiston tai sen osien ajamista, vaikka päivittäin, jotta varmistetaan, että kaikki toimii samalla tavalla. Automaatioon on syytä panostaa niin ohjelmistoalalla kuin yleisestikin, sillä se helpottaa työntekoa, vähentää virheitä, parantaa laatua ja tarkkuutta sekä tuo mukanaan monia muita hyötyjä. Automaatiolla saadaan myös ISAM:lle hyötyjä, koska tarvitsemme monia tietoja, joita pystymme automatisoimaan.

Insinöörityön aiheena on tutustua IBM Security Access Manager ohjelmaan ja sen automatisointiin. Työn tavoitteena on kehittää automaatioskriptit käyttäen JSON- ja shell-skriptejä sekä tutustua CA Release Automaatio -ohjelmaan ja sen tarjoamaan XML-pohjaiseen lähestymistapaan. Työssä tutustutaan molempiin tapoihin toteuttaa automaatiota, verrataan miten ne eroavat käyttöliittymässä toteutetusta hallinnasta sekä syvennytään ISAM:in toimintaan. Luvuissa 2 ja 3 paneudutaan yleisesti ISAM:in toimintaan. Luvut ovat teoriapainotteisia. Luku 4 on käytännönläheinen katsaus, joka sisältää esimerkkejä skripteistä ja huomioita skriptien kehittämisestä.

2 Tietoturva ja IBM Security Access Manager

Tietoturva on nykyään kasvavassa osassa ohjelmistokehitystä, kuten on ilmaistu liitteessä 1 (2). Tietoturva pitää sisällään paljon asioita ja ratkaisee monia ongelmia. Sen avulla saamme hyödynnettyä datan suojausta ja yksityisyyden hallintaa. Ilman tietoturvaa meillä olisi ohjelmistoja joissa olisi paljon bugeja, joita voidaan käyttää hyödyksi ja sitä kautta päästä käsiksi erilaisiin tietoihin. Tietoturvan avulla pystymme estämään pääsyä ohjelmille, jos halutaan, että vain tietyt henkilöt pääsevät tietoihin käsiksi. Voimme

myös estää monia tietoturva skenaarioita, joiden avulla toinen ihminen pystyisi muokkaamaan tietoja haitallisiksi.

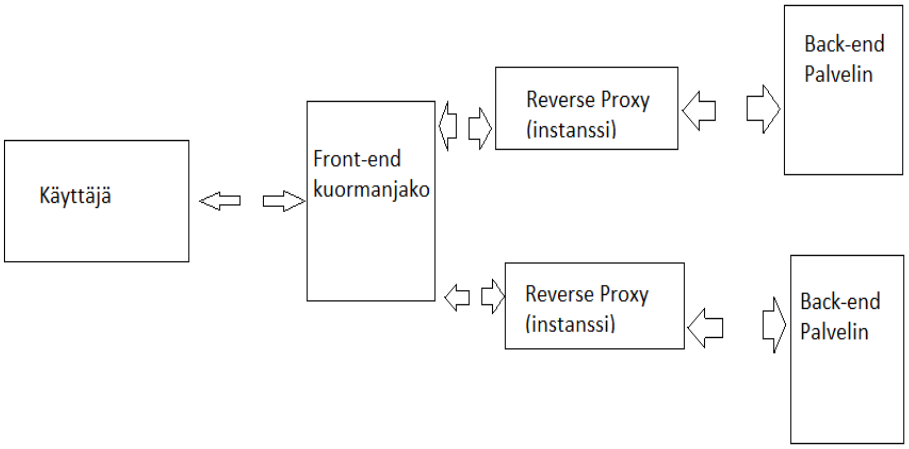
Tietoturvan toimintatapaa kannattaa pitää kolmiportaisena mallina kuten Lehtinen (12) ehdottaa. Ensimmäisenä laitetaan ja pidetään perusta kunnossa. Tähän työhön voidaan pitää hyvänä mallina tietoturvallisuusstandardeja, kuten ISO 27001. Tärkeää on saada tietoturva kattavaksi, jossa huolehditaan kaikista osa-alueista. Toisena tekijänä kannattaa pitää bisnes vaatimuksia. Nämä huomioidaan, kun säädetään perustaa. Kolmantena osana tietoturvatyössä on jatkuva prosessi. Tämä perustuu ihmisen, prosessin ja teknologian oikeaan suhteeseen.

Tietoturvaan investoidaan entistä enemmän (2) ja sen mukana tulee päätöksiä mihin kannattaisi panostaa. Lehtinen mainitsee (12), että tilanteesta riippuen kannattaa miettiä mihin osa-alueeseen kannattaa panostaa. Joskus se voi olla ihmisten koulutus, joskus jokin muu osa-alue.

Tietoturva sekä automaatio kannattaa ottaa osaan jokapäiväistä toimintaa. Pystymme mittaamaan standardien avulla yrityksen tietoturvaa. Näihin prosesseihin on helppo sisällyttää uusia asioita ja ne tulevat huomioitua samalla.

Nykyään ihmisten tietoja on monilla eri sivuilla ja tietoturvan täytyy olla kunnossa, jotta henkilökohtaiset tiedot eivät päädy ulkopuolisten käsiin. Tietojen täytyy olla vain tiettyjen ihmisten tai ryhmien nähtävissä. Niitä ei haluta paljastaa muille. Tietoja yritetään joskus muokata haitallisiksi lähettämällä kutsuja ja koodia, joilla saadaan haitallista vaikutusta aikaan. Näihin ongelmiin IBM Security Access Manager on todella sopiva työkalu.

IBM Security Access Manager on IBM:n kehittämä ohjelma (1), jolla yritykset voivat helpottaa pääsynhallintaa sekä kirjautumista sivuille sekä istunnonhallintaa. Kuten kerrottuna ohjelman omilla sivuilla (1), ohjelmaa voidaan käyttää niin web-, mobiili- sekä pilvipalveluiden kehittämisessä. ISAM yksinkertaistaa sekä luo turvaa käyttäjä kokemuksille yhdellä kirjautumisella ohjelmille sekä suojaa kriittisiä tietoja vahvalla tunnistautumisella sekä riskiperäisellä pääsyllä. ISAM skaalautuu hyvin tarpeisiin sekä on hyvin konfiguroitavissa oleva pääsynhallintasovellus kuten näemme kuvasta 1. Sen avulla voimme toteuttaa monia palveluita yhdelle käyttäjälle.



Kuva 1. Konfiguraation mahdollisuuksia.

3 Sovelluksen käyttäminen

Sovellusta voidaan käyttää neljällä eri tavalla ja kaikilla saadaan sama lopputulos. Tässä osiossa perehdymme neljään eri tapaan, joilla voimme käyttää sovellusta sekä miten ne toimivat. Luomme instanssin käyttöliittymän kautta ja tutkimme, mitä kaikkea tarvitsemme instanssin luomiseen. Tarvitsemme näitä tietoja ymmärtääksemme, mitä teemme, kun luomme automaatioskriptit luvussa 4. Tässä luvussa opimme hyödyllisiä komentoja ja miten kaikki tämä liittyy ISAM-automaatioon.

3.1 Käyttöliittymä

IBM Security Access Manager -ohjelmaa aloitetaan yleensä käyttämään käyttöliittymän kautta koska sillä voidaan tehdä perusasiat helposti ja käyttäjäystävällisesti. Tässä osiossa käymme läpi yleisimmin käytetyt palvelut (10) mutta emme kaikkea mahdollista. ISAM:in kirjaudutaan sisään normaalilla käyttäjätunnus- ja salasanayhdistelmällä kuvan 2 näköiseltä sivulta.

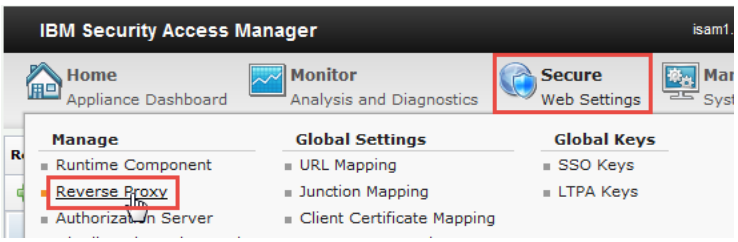
Kuva 2. Kirjautumisikkuna (9).

Seuraavaksi pääsemme itse käyttöliittymään, josta löytyy monia eri vaihtoehtoja kuten sovelluksen hallintaa, instanssien monitorointia ja lokituksia. Käyttöliittymässä voimme esimerkiksi luoda instansseja, muokata niitä sekä poistaa niitä.

3.1.1 Instanssin luominen

Pystymme luomaan uusia instansseja kätevästi käyttöliittymän (9; 10) kautta. Instansseja voi olla monta samaan aikaa ja kaikki ovat helposti hallittavissa yhdestä ja samasta paikasta.

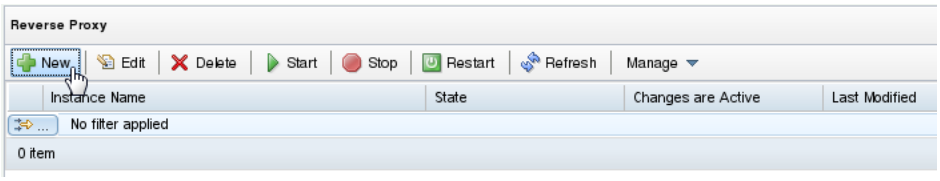
Aloitamme kirjautumalla käyttöliittymään ja tämän jälkeen siirrymme Secure -välilehdellä olevaan Reverse Proxy kohtaan, jonka näemme kuvassa 3.



Kuva 3. Instanssien hallinta (9).

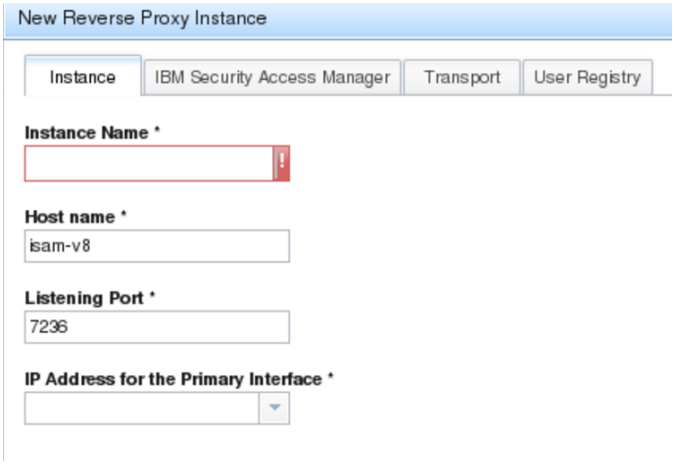
Täältä löydämme kaikki luomamme instanssit sekä voimme luoda niitä lisää, poistaa, muokata, pysäyttää tai käynnistää instanssit sekä muokata instanssien konfiguraatiotiedostoja.

Seuraavaksi luomme uuden instanssin painamalla New-nappia kuvan 4 mukaisesti.



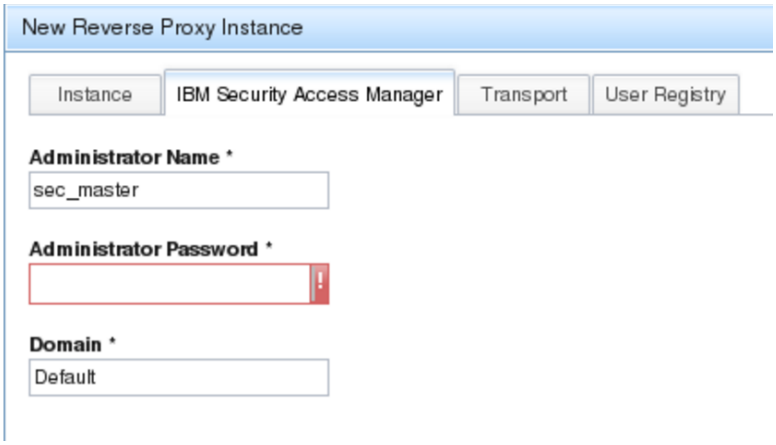
Kuva 4. Instanssin luomisen aloittaminen (9).

Nyt annamme tiedot instanssin nimelle, Host-nimelle, portille, jota kuunnellaan sekä IP-osoitteen. Nämä tiedot pitää hakea etukäteen ja ottaa sellaiset, jotka ovat vapaana.



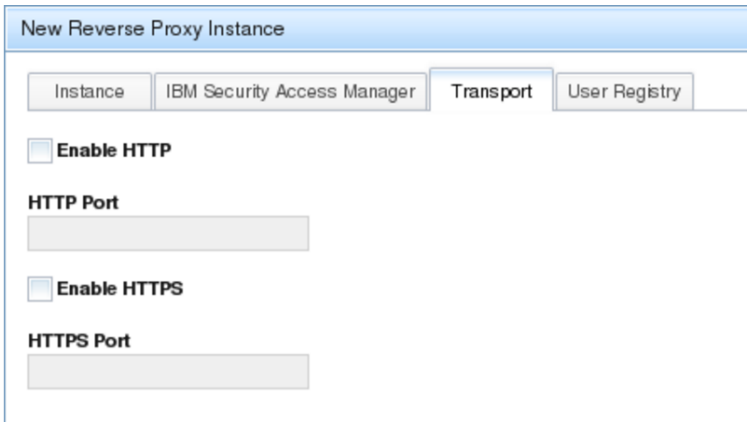
Kuva 5. Instanssin perustietoja (9).

Seuraavalla välilehdellä, joka on IBM Security Access Manager, annamme Administratorin tiedot sekä Domainin, jota käytetään yleensä. Domainina pidetään defaulttia, mutta se voisi olla jotain muutakin.



Kuva 6. IBM Security Access Manager -välilehden tiedot (9).

Seuraavalla Transport-välilehdellä voimme valita, käytämmekö HTTP:tä vaiko HTTPS:ää sekä niiden portit. Yleensä käytetään HTTP-porttina 80 ja HTTPS-porttia 443.



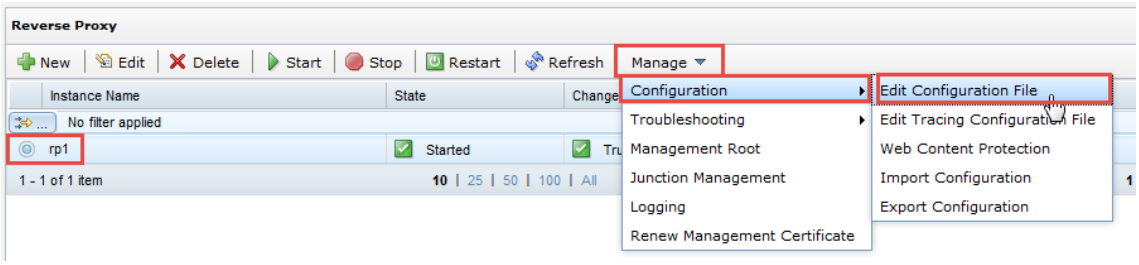
Kuva 7. Transport-välilehden tiedot (9).

Näin olemme luoneet uuden instanssin perusasetuksilla. Luomamme instanssi näkyy nyt listassa Reverse Proxy -sivulla. Tällaisella perusinstanssille emme tee juurikaan mitään ja seuraavassa osassa muokkaamme instanssia ja sen tietoja.

3.1.2 Instanssin tietojen muokkaaminen

Tässä osiossa tutkimme, kuinka voimme muokata instanssia sekä sen konfiguraatiota (9; 10). Yleisimpiä perusasioita on mahdollista muokata valitsemalla instanssi ja painamalla edit. Täältä löytyy samat asiat, joita olemme konfiguroineet jo aiemmin, kun loimme perusinstanssin, sekä tämän lisäksi useampi välilehti, jossa voimme muokata erilaisia tietoja. Pystymme muokkaamaan esimerkiksi Session, Autentikoinnin, Serverin sekä monia muita tietoja. Nämä tiedot löytyvät välilehdiltä ja niiden sisällä on monta tietoa, joita voisi muokata.

Kun haluamme muokata enemmän tietoja kuin mitä yleisimmistä asioista löytyy, meidän täytyy mennä muokkaamaan konfiguraatiotiedostoa kuvan 8 mukaisesti.



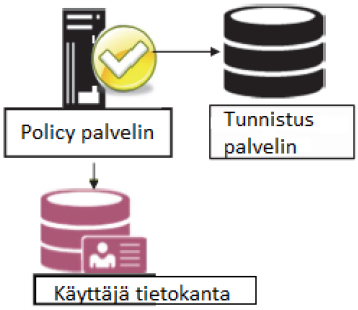
Kuva 8. Konfiguraatiotiedoston löytäminen (9).

Konfiguraatiotiedostossa on kaikki perustiedot sekä monta muuta tietoa, jota instassimme käyttää. Täällä voimme asettaa tietoja kuten esimerkiksi, voiko kirjautumaton käyttäjä kirjautua ulos, Server-nimen sekä kaikki tiedot, mitä löytyy tavallisesti edellä käydystä edit-vaihtoehdosta. Tänne tiedostoon joudutaan tekemään välillä muutoksia, jotka eivät ole konfiguroitavissa tavallisesti käyttöliittymän kautta. Nämä tiedot löytyvät helposti, kun tiedostosta voi etsiä suoraan haluttuja tietoja ja muokata niitä. Tämän jälkeen konfiguraatio pitää viedä palvelimelle ja instanssi käynnistää uudelleen. ISAM ehdottaa näiden muutoksien eteenpäin viemistä ja kertoo, minne muutoksia on tehty.

3.1.3 Security Access Manager policy server

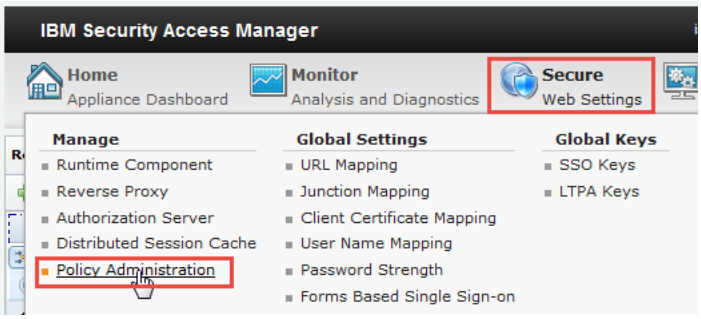
Instanssien käyttöön liittyviä tietoja voidaan hallita Access Manager policy -palvelimella (10). Policy server toimii LDAP:in ja tunnistus palvelimen välillä. Policy server tarkistaa sääntöjä ja päättelee näiden perusteella, voidaanko kutsuja päästää läpi, jotta saadaan

haettua käyttäjätietoja. Policy server toimii yksinkertaisesti kahden palvelun välillä kuten kuva 9 kuvastaa.



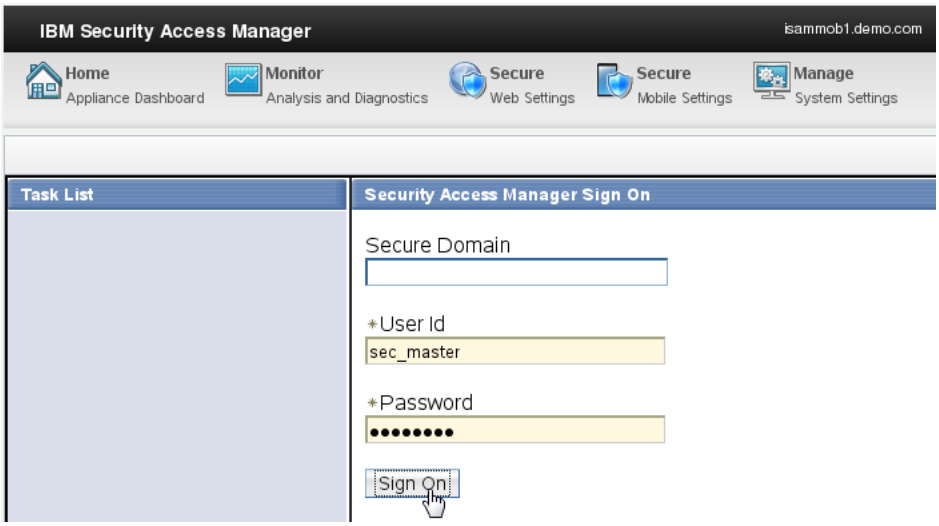
Kuva 9. Policy server.

Käyttääksemme palvelua meidän täytyy ensin mennä kuvan 10 osoittamalle sivulle.



Kuva 10. SAM policy -palvelimelle pääseminen (9).

Päästäksemme muokkaamaan policy-palvelimella olevia tietoja meidän täytyy kirjautua sisään kuvan 11 osoittamalle sivulle. Nämä tiedot annoimme instanssia luodessamme.



Kuva 11. Policy -palvelimelle kirjautuminen (9).

Tältä palvelimelta löydämme kuvan 12 mukaisen listan, josta voimme valita erilaisia tietoja, mitä muokkaamme. Voimme lisätä käyttäjiä ja heidän oikeuksiaan sivuille, voimme luoda ryhmiä käyttäjille ja määrittää heidän oikeutensa tai mahdollisesti poistaa kumpiakin. Pystymme tutkimaan instansseille luotuja junctioneita ja niihin liitettyjä tietoja. Junctionit eli solmukohdat ovat yhteyksiä, jotka liittävät front-end palvelimen ja back-end ohjelman toisiinsa. Voimme luoda solmukohtiin Access control listoja, Protected object policyjä sekä Authorization ruleja joiden avulla määritellään tiettyjä sääntöjä, ja erilaisia käyttäytymisiä kuten minkä tyylinen käyttäjä ja mitä käyttäjä saa tehdä.

Kuva 12. Lista policyistä, joita voi muokata (9).

ACL, POP, AuthzRule lisääminen junctioniin

Voimme lisätä junctionin käyttöön liittyviä rajoituksia lisäämällä junctioniin jonkin seuraavista ACL, POP tai AuthzRule tai sitten kaikki näistä. Halutessamme voisimme luoda kokonaan uuden ACL:n POP:n tai AuthzRulen.

ACL kertoo meille käyttäjien tai ryhmän suhteen tiettyyn toimintoon. Seuraavasta listasta näemme, mitä kaikkea ACL:llä voi tehdä.

- ACL:llä voimme rajoittaa, mitä operaatioita voidaan käyttää ja kuka niitä voi käyttää
- Yhdistää käyttäjät ja ryhmät, joilla on tietyt oikeudet suorittaa operaatioita.
- Voimme lisätä ACL:n tiettyyn objektiin eli instanssiin tai junctioniin.
    - Voidaan liittää junctioniin tai voi periytyä.
    - Voidaan liittää useampaan kuin yhteen objektiin

ACL vaikuttaa vain silloin, kun se on liitetty objektiin

POP antaa meille lisää ylimääräisiä ehtoja, joilla voimme hallita pääsyä objektille. Pystymme rajoittamaan pääsyä esimerkiksi seuraavilla ehdoilla.

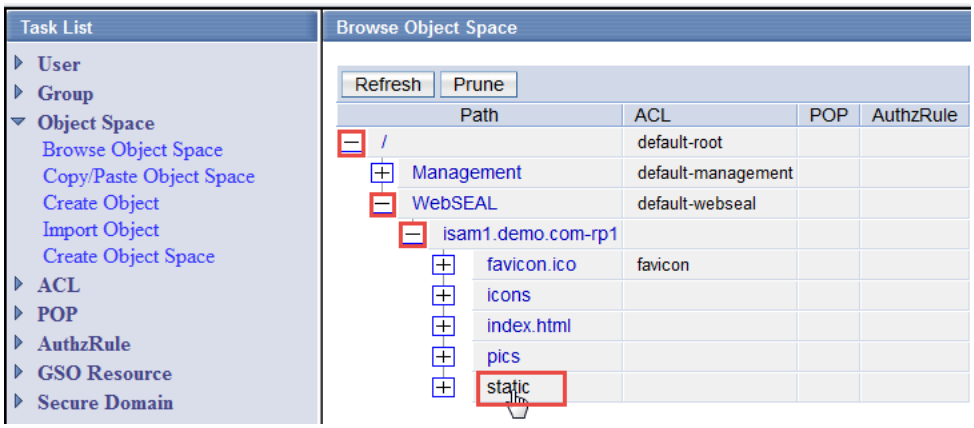
- tietty aika päivästä
- Ttetyt audit-tasot
    - salli
    - kiellä
    - admin
    - error
- IP-autentikointi
- varoitusattribuutti.

Audit-taso on taso, joka kuuluu resurssille, kun sitä käytetään. Näitä voidaan asettaa useampi kuin yksi. Varoitusattribuutilla tarkoitetaan sitä, kun administratorin pitää debugata tai selvitellä ongelmaa pääsynhallinnalle asetettujen policyjen tarkkuudessa.

POP sisältää ylimääräisiä ehtoja, jotka lähetetään takaisin resurssien hallinnalle. SAM ja resurssien hallinta vahvistavat POP:n käyttöönoton.

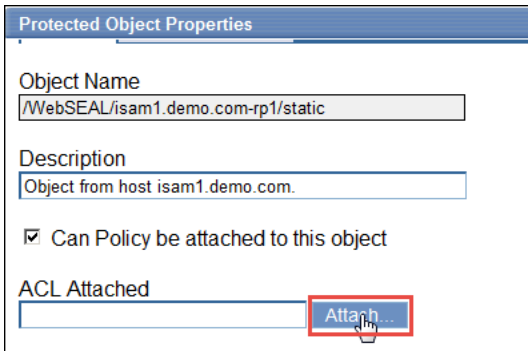
AuthzRulen avulla voidaan rajoittaa entisestään pääsyä. Voimme asettaa tiettyjä ehtoja, joita sitten käydään tarkastamassa ja verrataan käyttäjätietoja sekä application contextia.

Nyt tiedämme, mitä kaikkea voimme lisätä ja mihin ne vaikuttavat, joten lisäämme seuraavaksi ACL:n edellä luomaamme junctioniin. Avataan Object Space, josta otamme Browse Object Spacen ja sieltä menemme instanssiin kuvan 13 mukaisesti.



Kuva 13. Polku ACL:n lisäämiselle (9).

Tästä meille avautuu uusi ikkuna, jossa voimme lisätä ACL:n, POP:n tai AuthzRulen instanssiimme. Voimme liittää valmiin ACL:n kuvan 14 mukaisesti.



Kuva 14. Liitä ACL (9).

Samalla tavalla voisimme liittää kyseiseen instanssiin POP:n tai AuthzRulen.

ACL, POP, AuthzRule poistaminen

Halutessamme poistaa ACL;n, POP:n tai AuthzRulen instanssista voimme tehdä sen käyttöliittymässä ja SAM policy serverillä helposti. Pystymme myös poistamaan ACL:n POP:n ja AuthzRulen kokonaan halutessamme.

Poistamme ACL:n helposti näillä toimenpiteillä.

1. Mene SAM policy serverille ja kirjaudu sisään.
2. Mene ACL -> List ACL.
3. Valitse haluttu ACL ja klikkaa sitä aukaistaksesi uuden ikkunan.
4. Tältä sivulta valitaan Attach välilehti, klikataan sitä.
5. Jos ACL on liitetty useampaan instanssiin, pystymme täältä valitsemaan kätevästi kaikki instanssit, joista se otetaan pois.
6. Kun kaikki tarvittavat instanssit on valittu, klikataan Detach nappia ja vahvistetaan poistaminen.

Näin olemme poistaneet ACL:n kaikista halutuista instansseista. ACL on yhä olemassa sitä ei vain ole nyt liitetty instanssiin.

Voimme poistaa POP:n instanssista helposti seuraavilla toimenpiteillä.

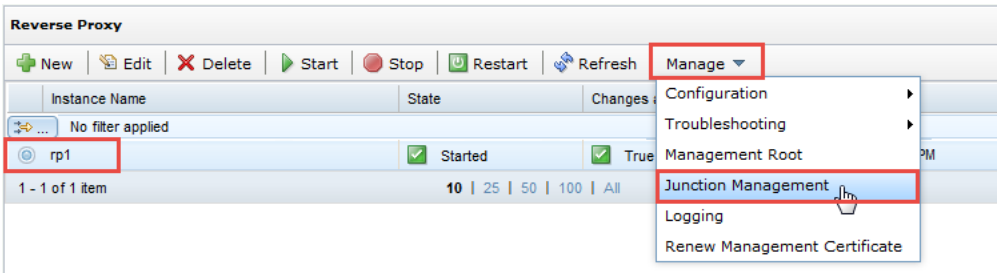
1. Mene SAM policy serverille ja kirjaudu sisään.
2. Mene POP -> List POP.
3. Valitse haluttu POP.
4. Siirry Attach välilehdelle.
5. Valitse yksi tai useampi instanssi joista POP poistetaan.
6. Valinnan jälkeen paina Detach nappia ja vahvista POP:n poistaminen instanssista.

Samoilla toimenpiteillä voisimme poistaa AuthzRulen muuttamalla vain edellä mainitut POP kohdat.

3.1.4 Solmukohdan luominen

Pystymme luomaan instansseihimme junctioneita (9; 10). Näitä voidaan tehdä kahdenlaisia: Static junctioneita tai Virtual Host Junctioneita (10). Junction on yhteys Reverse Proxyn eli instanssimme ja Back-end web-application-palvelimemme välillä.

Luodaksemme junctioneita instansseihin meidän täytyy mennä ISAM-käyttöliittymän kautta Security -> Reverse Proxy niin kuin kuvassa 3 teimme. Tämän jälkeen valitsemme instanssimme ja päästäksemme luomaan uusia junctioneita menemme Managereen ja sieltä Junction Managementiin kuten kuvassa 15.

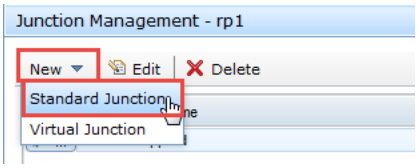


Kuva 15. Junction Management (9).

Tästä meille aukeaa uusi ikkuna, jossa voimme luoda Standard Junctioneita tai Virtual Host Junctioneita (10).

Standard Junction

Nyt luomme Standard Junctionin (10) eli valitsemme New ja sieltä Standard Junction kuten kuvassa 16.

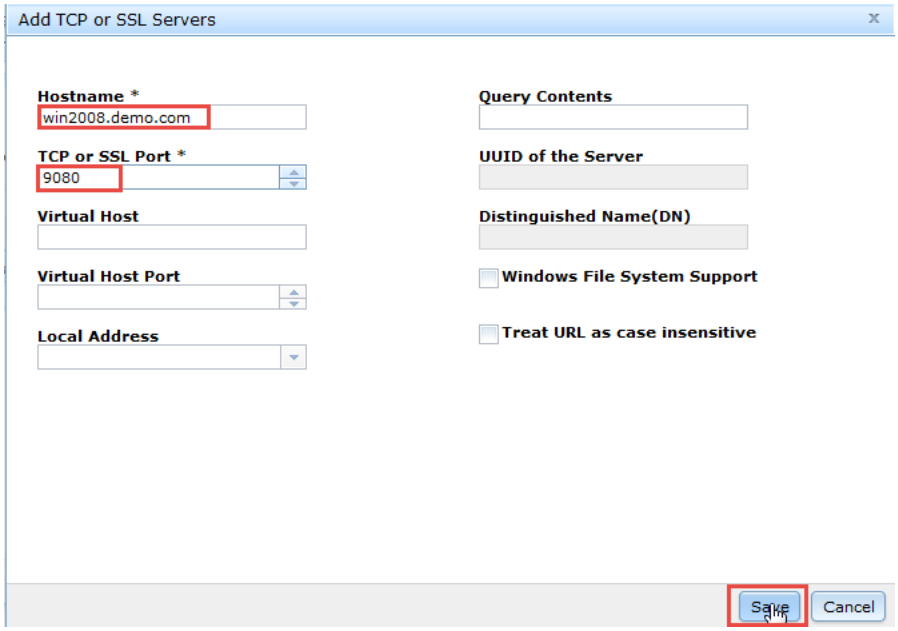


Kuva 16. Standard Junctionin valitseminen (9).

Seuraavaksi meidän pitää määritellä Junctionin tiedot (10). Täällä määrittelemme, mihin junctionimme osoittaa sekä sen tyypin ja onko se Transparent Path Junction vaiko Stateful junction. Kuvasta 17 nähdään esimerkki, mitä voitaisiin laittaa näihin arvoksi.

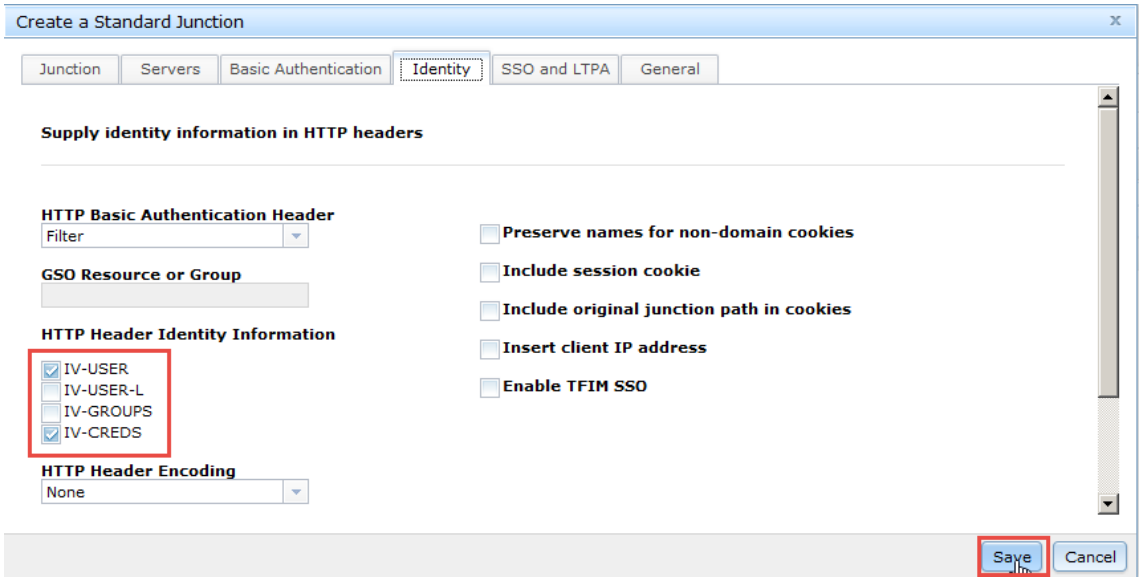
Kuva 17. Esimerkkiarvoja Junctionille (9).

Seuraavalla Server-välilehdellä meidän pitää määrittää, mille palvelimelle otamme yhteyttä. Luomme uuden yhteyden ja määritämme sille esimerkki tiedot kuvan 18 mukaisesti.



Kuva 18. Esimerkki palvelimen tiedoista (9).

Seuraavaksi menemme identity-välilehdelle ja valitsemme muutaman yleisen tiedon, jotka kulkevat mukana kutsuissa. Kuva 19 näyttää esimerkin, mitä voitaisiin ottaa mukaan. Valitsemamme tiedot kulkevat Headereissa mukana ja niitä voitaisiin käyttää esimerkiksi ohjelmiston koodissa tai ISAM:n pääsynhallinnassa.



Kuva 19. Esimerkkitiedot mukaan otettavista tunnistetiedoista (9).

Nyt olemme luoneet tavallisen junctionin, tähän junctioniin voisimme liittää samalla tavalla ACL, POP tai AuthzRulen kuin edellisessä 3.1.3-kohdassa.

Virtual Host Junction

Virtual Host Junctioneilla (10) tarkoitetaan sitä, kun halutaan säilyttää useampia hosteja yhdellä koneella. Nämä ovat erotettuja pelkästään host-nimen perusteella. Tämän avulla voimme käyttää useita web-palveluita samalla palvelimella, vaikka ne vaikuttavat täysin erilaisilta. Virtuaaliset Hostit tarvitsevat DNS-muokkauksia, koska ne osoittavat samaan IP-osoitteeseen.

Luodaksemme Virtual Host Junctionin meidän pitää muokata ISAM:n Host fileä joka löytyy Manage -> Network Settings ->Hosts File. Täältä valitaan ensin tietty IP-osoite, johon lisätään haluttu Hostname. Tämän tallennettua muutokset pitää Deployata serverille,

jotta ne tulisivat voimaan. Nyt meillä voisi olla kaksi eri Hostnamea osoittamassa samalle IP:lle.

Tämän jälkeen menemme Secure -> Reverse Proxy ja valitsemme instanssimme, johon haluamme lisätä Junctionin. Seuraavaksi Manage -> Junction Management ja täältä luodaan uusi Virtual Host Junction valitsemalla New -> Virtual Host Junction. Junctionille annetaan haluttu Label ja tyyppi. Tämän jälkeen siirrytään Server-sivulle, johon tehdään uusi yhteys liittäen haluttu Virtual Hostname sekä portti. Sitten tekemiset tallennetaan ja olemme luoneet Virtual Host Junctionin, joka osoittaa tiettyyn osoitteeseen.

3.1.5 Solmukohdan poistaminen

Voimme poistaa luomamme solmukohdat, olipa se Standard tai Virtual Host Junction (10). Tämän tehdäkseen täytyy siirtyä Secure -> Reverse Proxy valita oikea instanssi siirtyä Manage -> Junction Management ja tästä valita haluttu solmukohta. Kun oikea solmukohta on valittu, voimme painaa Delete-nappia, joka haluaa vahvistuksen, olemmeko varmoja, että haluamme poistaa kyseisen solmukohdan.

Pystymme poistamaan solmukohdan myös toisesta paikasta menemällä Secure -> Policy Administrator ja kirjautumalla sisään. Tämän jälkeen valitaan Object Space -> Browse Object Space ja etsitään listasta instanssi, johon solmukohta on liitetty. Kun oikea instanssi on löytynyt, avaamme instanssin listasta ja valitsemme sen alta poistettavan solmukohdan. Klikkaamalla solmukohtaa pääsemme uudelle sivulle, josta voimme poistaa solmukohdan painamalla Delete-nappia ja vahvistamalla tapahtuman. Näin helposti solmukohdan pystyy poistamaan kahdella eri tavalla.

3.1.6 Instanssin poistaminen

Pystymme poistamaan myös kokonaisia instansseja. Toinen vaihtoehtomme on mennä suoraan ISAM:n Secure -> Reverse Proxy ja valita instanssi. Tämän jälkeen painamme Delete-nappia ja vahvistaa, haluammeko varmasti poistaa instanssin.

Toinen vaihtoehtomme poistaa instanssi on mennä Secure -> Policy Administrator, jossa kirjaudumme sisään. Tämän jälkeen täytyy klikata Object Space -> Browse Object Space ja valita listasta instanssimme, joka halutaan poistaa. Klikkaamalla kyseistä instanssia

päädymme sivulle, jossa voimme painaa Delete-nappia ja vahvistaa poiston. Näin helposti pystymme poistamaan instansseja kahdesta eri paikasta.

3.1.7 Yhteenveto

Loimme instanssin sekä kaikki siihen tarvittavat osat käyttöliittymän kautta, tutkimme miten tietoja muokataan sekä miten instansseja voidaan poistaa. Kuten huomaamme, tämä on todella suuri homma hoitaa kaikki käsin käyttöliittymän kautta. Nämä samat asiat käydään vielä läpi ja tehdään ne eri tavoilla mikä helpottaa tätä urakkaa. Tätä käytetään automaatiossa hyödyksi, josta kerrotaan seuraavissa luvuissa.

3.2 Komentorivi (Command-line Interface)

Komentoriviin pääsemme käsiksi joko SSH:ta käyttäen tai sitten suoraan konsolista. Komentorivillä voimme tehdä samat asiat kuin käyttöliittymässä. Kuten oppaissa (3; 4) on käsitelty komentorivin avulla hyödynnettäviä komentoja. Tässä osiossa luomme omaan tarkoitukseemme sopivat. Käymme tässä osiossa läpi samat osa-alueet kuin edellisissä osioissa eli yleisimmin käytetyt palvelut ISAM:ssa. Nyt näytämme kutsut, miten komentorivillä voisimme tehdä kaikki samat asiat. Kyseisiä kutsuja tarvitaan luvussa 4 tehtävissä skripteissä.

3.2.1 Instanssin luominen

Voimme luoda instanssin seuraavalla tavalla. Ensin täytyy kirjautua sisään domainiin. Tämän jälkeen annamme seuraavan komennon, jolla luomme instanssin.

```
object create /WebSeal/test-instance ”Test Instance” 14 ispolicyattachable yes
```

Komennossa kerromme ISAM:lle, että pdadmin (5) luo instanssni, seuraavaan polkuun /WebSeal/test-instance, pieni kuvaus mikä instanssi on. Seuraavaksi voimme valita eri tyyppejä, minkälaisen instanssin haluamme luoda sekä määrittää, voidaanko tähän liittää eri policyjä. Instanssin tyyppejä on seuraavanlaisia.

0. Unknown

1. Secure Domain
2. File
3. Executable Program
4. Directory
5. Junction
6. WebSEAL Server
7. Unused
8. Unused
9. HTTP Server
10. Nonexistent object
11. Container object
12. Leaf object
13. Port
14. Application container object
15. Application leaf object
16. Management object
17. Unused

Näin pystymme komentoriviltä tekemään instansseja ja määrittämään niiden tyyppejä.

3.2.2 Instanssin tietojen muokkaaminen

Pystymme muokkaamaan instanssin tietoja seuraavilla komennoilla. Voimme esimerkiksi vaihtaa tyypin komennolla object modify /WebSEAL/test-instance type 15, tai valitsemalla jonkin muun tyypin. Voimme muuttaa instanssia ja määrittää, onko tähän mahdollista lisätä policyjä seuraavalla komennolla object modify /WebSEAL/test-instance isPolicyAttachable no. Tähän voimme valita vain yes tai no. Instanssin kuvausta voi muokata seuraavalla komennolla object modify /WebSEAL/test-instance set description ”Esimerkki”.

Komennot ovat pääsääntöisesti seuraavankaltaisia, kun halutaan muokata tietoja.

```
object modify objektin_nimi komento arvo
```

3.2.3 ACL, POP ja AuthzRule

Pystymme myös liittämään ACL:n, POP:n sekä AuthzRulen instasseihin komentoriviltä (3; 5). Ensin täytyy kirjautua domainiin, jotta pystyisimme liittämään ja poistamaan ACL:n.

ACL:n voimme lisätä edellisessä luvussa olevaan instanssiin seuraavalla komennolla. acl attach /WebSEAL/test-instance Test-ACL. Tässä määrittelemme mihin instanssiin laitamme jo olemassa olevan ACL:n. Voimme myös poistaa ACL:n instanssista seuraavalla komennolla. acl detach /WebSEAL/test-instance.

Pystymme myös luomaan ACL:n joko perusarvoilla tai itse määrittämillämme arvoilla, poistamaan sekä muokkaamaan jo olemassa olevia ACL:iä komentoriviltä seuraavilla komennoilla. acl delete ACL_nimi, acl create ACL_nimi tai acl modify ACL_nimi komento arvo. Kun muokkaamme ACL:ää, haluamme tietää komennossa, mitä muokataan ja arvoksi laitetaan se, mikä siihen halutaan.

Myös POP (5) on mahdollista liittää komentoriviltä instanssiin seuraavalla komennolla pop attach /WebSEAL/test-instance poptest. Se voidaan myös poistaa instanssista yhtä helposti komennolla pop detach /WebSeal/test-instance.

Voimme myös luoda uusia POP:eja perusarvoilla tai itse määrittämillä arvoilla, poistaa niitä tai muokata jo olemassa olevia seuraavilla komennoilla. pop create POP_nimi, pop delete POP_nimi, pop modify POP_nimi komentoarvo. Muokatessa tietoja komentoja on useita erilaisia, mutta perusrakenne pysyy samana.

AuthzRuleja (5) pystytään lisäämään instansseihin sekä poistamaan instansseista seuraavilla komennoilla. authzrule attach /WebSEAL/test-instance authzruletest, authzrule detach /WebSEAL/test-instance.

AuthzRuleja pystytään luomaan, poistamaan sekä muokkaamaan seuraavilla komennoilla. authzrule create Rule_name, authzrule delete Rule_name, authzrule modify Rule_name komentoarvo.

3.2.4 Solmukohdan luominen

Tässä osiossa käymme läpi, kuinka komentorivillä pystytään luomaan instansseihin junctioita eli solmukohtia. Solmukohdat liittävät front-end palvelimen ja back-end ohjelman toisiinsa. Tähän käytämme komentoa server task. Seuraavalla komennolla voisimme luoda junctionin edellä luomaamme instanssiin. Ensin täytyy kirjautua sisään, jotta voimme ajaa komentoja. Komennolla server task test-instance-webseald-example

create -t tcp -h www.test.com /test-junction saamme luotua uuden junctionin instanssiimme. Tässä komennossa täytyy määritellä instanssin sekä hostin nimi eli test-instance, jota seuraa vakioarvona webseald, jonka jälkeen voisimme laittaa examplen tilalle joko host nimen tai tässä tapauksessa käytetyn server-nimen, voimme määritellä komennossa host-nimen myös myöhemmin komennolla -h ja arvo sen perään. Tässä komennossa pystymme määrittelemään eri arvoja, joita instanssille asetetaan, jos tahdotaan, että arvot eivät ole vakioita.

Helpottaaksemme solmukohdan luontia saamme helposti listattua instanssien nimet tälle komennolle muotoon instance-webseald-host_name, seuraavalla komennolla server list (3). Kyseinen komento antaa meille listan kaikista instansseistamme ja saamme samalla koko instanssin nimen, jota tarvitaan server task -komennon suorittamiseen.

3.2.5 Solmukohdan poistaminen

Solmukohdan poistamiseen meidän kannattaa katsoa ensin server list -komennolla (3) instanssimme osoite kokonaisuudessaan, koska seuraavaksi tarvitsemme tätä tietoa. Pystymme poistamaan solmukohdan instanssista komennolla. server task test-instance-webseald-example delete /test-junction. Komennossa määritellään, mistä instanssista halutaan poistaa haluttu solmukohta. Komento ei toimi, jos siihen ei ole määritelty koko instanssi-webseald-host_name-polkua.

3.2.6 Instanssin poistaminen

Voimme poistaa, myös kokonaisia instansseja, mikäli haluamme (4). Seuraavalla komennolla poistamme instanssin palvelimeltamme. Komento object delete /WebSEAL/test-instance poistaa instanssin. Komentoon pitää määritellä koko polku instanssille, jotta se toimii.

3.2.7 Yhteenveto

Olemme luoneet instanssin ja junctionin komentoriviltä, muokanneet niiden tietoja sekä poistaneet molemmat. Kuten huomaamme, ISAM:ssa on monia mahdollisia komentoja,

joita voi suorittaa komentoriviltä. Näitä komentoja on helppo hyödyntää, kun tehdään automaatiota.

3.3 JSON

IBM Security Access Manager ymmärtää käsitellä tietoa myös JSON-tyyppisenä tietona. JSON eli Java Script Object Notation on tiedostomuoto tiedonvälitykseen. Pystymme määrittelemään JSON:iin avain-arvo-pareja jotka ISAM osaa selvittää. Tätä tietoa tarvitsemme, kun luomme JSON-muotoiset skriptit luvussa 4.

```
{ ”avain” : ”arvo” }
```

Esimerkki 1. Esimerkki JSON-muodosta.

3.3.1 REST-rajapinta

ISAM:ssa on sisään rakennettu REST-rajapinta (8), jota voimme kutsua tietyllä kutsulla ja lähettää sinne JSON-tiedostoja. REST-rajapinta eli Representational state transfer tarjoaa web-palveluille yhteyden ohjelman ja internetin välille. REST-rajapinta tukee JSON- ja XML-muotoja, joita lähetämme palvelimelle. Voidaksemme käyttää hyödyksi ISAM:n REST-rajapintaa tarvitsemme sovelluksen, jossa on hyödyllinen työkalu curl kuten esimerkiksi Cygwin.

ISAM REST -rajapintoja käytetään, jotta voidaan suorittaa administrator-tehtäviä, konfiguroida tietoja, deployata muutoksia, kerätä dataa. Pystymme lähettämään verbejä, joita tarvitsemme komennoissa eli PUT, GET, DELETE ja POST.

3.3.2 Käyttö

Kuten Shane (6) mainitsee voimme automatisoida komentoja. Voimme käyttää JSON-muotoista dataa ja REST-rajapintaa seuraavalla komennolla. curl -s -k -h ’accept:application/json’ -u käyttäjätunnus:salasana -X GET instanssin_osoite. Tämä komento palauttaa meille instanssin nimen, tarvitseeko se uudelleenkäynnistämistä, onko se päällä, id:n, versionumeron sekä onko se käytössä.

Päästäksemme käyttämään JSON-muotoista dataa ISAM:n automaatiossa tarvitsemme edellisessä osiossa käytettyjä komentorivi kutsuja (5), joita ISAM ymmärtää.

3.4 Automaation luominen Release-automaatiolla

ISAM:ia on mahdollista automatisoida myös esimerkiksi CA Release Automaation (11) avulla. Tämä vaatii tosin sen, että käyttäjillä on kyseinen tuote. Tämän tuotteen avulla pystymme kuitenkin luomaan esimerkiksi XML-muotoista dataa, jonka CARA (11) sitten muuttaa ISAM:lle sopivaan muotoon eli JSON:ksi. Tämä helpottaa työntekoa ja instanssien ylläpidettävyyttä ja hallintaa selvästi, sillä tiedot eivät pääse muuttumaan välissä ja niitä on helppo säilyttää yhdessä paikassa.

CA Release Automation (7) on tuote, joka sisältää end-to-end applikaation releasointihallinan, jotta voidaan käyttää DevOpsia sekä muita asioita hyödyksi. Voimme automatisoida ja standardisoida applikaatioita. Tämä on valmis paketti, joka sisältää kaikki tarvittavat tiedot yhdessä paketissa, lisäksi tämä on todella hyvin muokattavissa.

CARA:n (7) avulla saadaan ratkaisuja moniin bisnesongelmiin. Bisnesongelmiin kuuluu muun muassa paine tuottaa ratkaisuja nopeammin ja useammi. Yleensä yrityksessä on monia monimutkaisuuksia, joita saadaan tämän avulla ratkaistua sekä tietyt virhetilanteet ja palvelimien downtimet. CARA (7) on tuote, joka mahdollistaa monia etuja automaation luomiselle. Tuotteen avulla pystytään nopeasti ja luotettavasti deployaamaan erilaisia palvelimia, datakeskuksia tai vaikka pilvipalveluita. CARA:n (10) avulla on mahdollista suunnitella, hallita sekä optimoida vientiputkea. Tätä pystytään muokkaamaan sitä mukaa, kun itse halutaan ja koetaan tarpeelliseksi.

4 Automaatioskriptien luominen

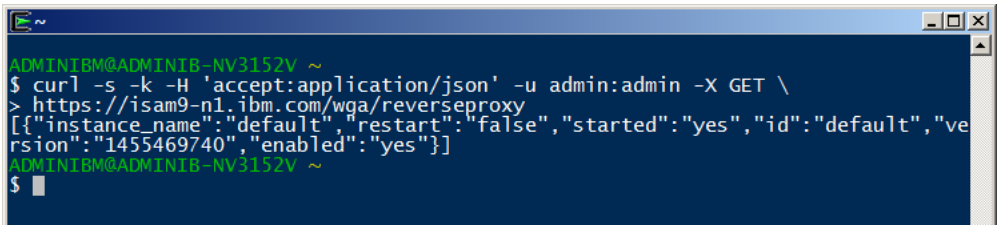
Tässä osiossa luomme skriptit, joita voidaan käyttää jatkossa ISAM:n automatisointiin (6) sekä toteutetaan hyvä pohja skriptien jatkokehitykselle. Luomme JSON:ia hyväksikäyttävät shell skriptit sekä tarvittavat JSON:t, sekä katsomme, miten voisimme käyttää hyväksi Release Automaatiota. Teemme tätä varten muutaman XML-pohjaisen skriptin esimerkiksi, miten RA:n avulla voidaan toteuttaa automaatiota.

4.1 JSON

Pystymme luomaan instansseja, muokkaamaan niitä sekä poistamaan niitä. Luomme REST-rajapintaa (6; 8) kutsuvan shell-skriptin, jonka avulla saamme hallittua myös yhteyksiä helposti.

4.1.1 REST-komentojen automatisointi

Aloitamme ottamalla komennon curl -s -k -h 'accept:application/json' -u käyttäjätunnus:salasana -X GET instanssin_osoite (6; 8) ja rupeamme tekemään skriptiä. Skriptin nimeksi laitamme rest_cmd.sh. Määrittelemme ensin polkua, jolla komentoa ajetaan sekä kolme muuttujaa path, method, data, jotka kaikki löytyvät esimerkistä 1.



Kuva 20. Komennon vastaus JSON-muodossa (6).

Seuraavaksi voimme määrittää itse komentoa. Määrittelemme curl ja sen erityyppiset muuttujat kuten käyttäjän, sisältötyypin, mitä hyväksytään, kun saadaan vastaus, metodin, datan, sekä osoitteen, johon tiedot lähetetää. Viimeisenä laitetaan tulostamaan saatu vastaus, jotta tiedetään, onko komento mennyt läpi. Tämä löytyy Esimerkistä 1.

Yhdistämällä nämä kaksi edellä mainittua asiaa saamme kokonaisen skriptin, joka luo yhteyden ISAM-serveriin sekä lähettää seuraavaksi tehtävät JSON:it palvelimelle käsiteltäviksi. Tässä esimerkissä on mainittu kaikki tarvittavat tiedot yhteyden luomiselle ja komentojen lähettämiselle.

```
#! /bin/sh
path=$1
method=$2
data=$3
curl -k \
```

```
--user käyttäjätunnus:salasana \
-H Content-type:application/json \
-H Accept:application/json \
-s \
-X $method \
-d @$data \
https://127.0.0.1/$path | sed 's/,/,\n/g'
echo
```

Esimerkki 2. shell-skripti yhteyden luomiselle ja komento, jolla saadaan lähetettyä JSON.

4.1.2 Instanssin luominen JSON:ien ja skriptien avulla

Seuraavaksi luomme shell-skriptin joka käyttää hyödyksi äsken luomaamme shell-skriptiä rest_cmd.sh sekä JSON:it, joilla välitämme tiedon. Luomme kokonaisen instanssin tämän skriptin ja JSON:ien avulla. Aloitamme luomalla JSON:it, joita tarvitsemme ja lopuksi yhdistämme tarvittavat JSON:it yhdelle luontiskriptille.

4.1.2.1 Instanssin luomiseen tarvittavat JSON:it ja komennot

Tarvitsemme instanssin luonnille ja uudelleenkäynnistämiselle omat JSON:it sekä komennon sen suorittamiselle. Aloitamme instanssin luonti JSON:lla, jossa määritämme instanssin nimen, host-nimen, portin sekä muita tietoja.

```
{
 "inst_name": "testinst",
 "host":"WEB1",
 "listening_port":"8080",
 "domain":"Default",
 "admin_id":"käyttäjätunnus",
 "admin_pwd":"salasana",
 "ssl_yn":"yes",
 "ssl_port":"636",
 "key_file":"keyfile.kdb",
 "cert_label":"LDAP",
 "http_yn":"yes",
 "http_port":"80",
```

```
"https_yn":"no",
"https_port":"8443",
"nw_interface_yn":"yes",
"ip_address":"127.0.0.1"
}
```

Esimerkki 3. Instanssin tiedot JSON.

Tässä olemme määrittäneet kaikki tarvittavat tiedot instanssille. Tietoja voi muokata itselleen sopivaksi muuttamalla vain arvoja. Seuraavaksi luomme komennon deploylle ja restart JSON:iin.

```
./rest_cmd.sh pending_changes/deploy GET empty
```

Esimerkki 4. Deploy-komento

Komennossa käytetään hyödyksi aiemmin luomaamme rest_cmd shell -skriptiä ja tämän jälkeen tehdään kutsu ISAM:n omalla kutsulla.

Restart JSON on yhden rivin pituinen, jossa määritellään operaatio, joka ISAM:n pitää ajaa.

```
{"operation": "restart"}
```

Esimerkki 5. Restart JSON

Seuraavaksi meidän pitää määritellä tietty määrä pakollisia muuttujia, jotta instanssimme toimii moitteettomasti. Luomme kaikille muuttujille JSON:it ja laitamme niille tarvittavat arvot. Luomme muutaman esimerkin malliksi, koska tulemme käyttämään näitä sekä muita vastaavia, kun kasaamme yhden skriptin, joka kasaa kaikki tarvitsemamme JSON:it yhteen.

```
{ "value":"WEB1-testinst" }
```

Esimerkki 6. server-name.json, joka määrittää palvelimen nimen.

```
{"value":"auto"}
```

Esimerkki 7. utf8-url-support-enabled.json, joka asettaa utf8-tuen.

```
{"value":"yes"}
```

Esimerkki 8. yes-valuen omaava JSON, jota voidaan käyttää hyödyksi monissa komennoissa.

```
{ entries:
[ ["web-host-name","wwwtest.testi.fi "] ]
}
```

Esimerkki 9. Tarvittavan web-host-namen määrittäminen

```
{ entries:
[
["server","9,https://TFIM:9443/osoite_josta_halutaan_tietoa"],
["handle-pool-size","1"],
["handle-idle-timeout","30"],
["timeout","30"],
["basic-auth-user","käyttäjätunnus"],
["basic-auth-passwd","salasana"],
["ssl-keyfile","keyfile.kdb"],
["ssl-keystore-stash","keystore.sth"]
]
}
```

Esimerkki 10. TFIM-klusterin määrittäminen, joka kertoo, mistä TFIM löytyy ja tarvittavat tiedot siihen.

```
{ entries:
[
["token-type",
"http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0"],
["applies-to", "http://wwwtest.testi.fi"],
["service-name", "*"],
["one-time-token", "true"],
["renewal-window", "15"],
["preserve-xml-token", "false"],
["token-collection-size", "1"],
["token-transmit-type", "header"],
["token-transmit-name", "SAMLv2Auth"],
["always-send-tokens", "true"],
```

```
["tfim-cluster-name", "fimCluster"]
]
}
```

Esimerkki 11. TFIM Single-SignOn:iin tarvittavat tiedot.

Nyt meillä on luotu tarvittavia JSON:eja, joilla voimme konfiguroida instanssimme oikein. Seuraavaksi voimme luoda junctionin JSON:in, jonka voimme sitten liittää instanssiimme.

```
{
 "server_hostname":"TFIM",
 "junction_point":"test1",
 "junction_type":"tcp",
 "junction_hard_limit":"0",
 "junction_soft_limit":"0",
 "basic_auth_mode":"filter",
 "client_ip_http":"yes",
 "tfim_sso":"yes",
 "remote_http_header":"iv-creds",
 "request_encoding":"utf8_uri",
 "virtual_hostname":"wwwtest.testi.fi",
 "case_sensitive_url":"yes",
 "server_port":"8080",
 "http_port":"8080"
}
```

Esimerkki 12. Virtual Host Junctionin luontiin tarvittavat tiedot JSON-muodossa.

Tässä olemme määrittäneen tarvittavat tiedot Virtual host junctionille, jotta ISAM voi luoda sen. Luomme samalla periaatteella myös Standard Junctionin.

```
{
 "server_hostname":"TFIM",
 "junction_point":"/TEST",
 "junction_type":"tcp",
 "junction_hard_limit":"0",
 "junction_soft_limit":"0",
```

```
"basic_auth_mode":"filter",
"client_ip_http":"yes",
"tfim_sso":"no",
"remote_http_header":"iv-creds",
"request_encoding":"utf8_uri",
"case_sensitive_url":"yes",
"transparent_path_junction":"yes",
"server_port":"9080",
"http_port":"9080"
}
```

Esimerkki 13. Standard Junctionin tarvitsemat tiedot JSON-muodossa.

Loimme nyt Standard junctionin, joka näyttää todella paljon samalta kuin Virtual host junction. Standard junctionissa (10) ei määritellä virtuaaliosoitetta ollenkaan ja voidaan määrittää transparent_path_junction Tässä on suurimmat erot, kun näitä kahta erityyppistä junctionia luodaan. Virtual Host Junction (10) saa osoitteensa www-muodossa ja siksi on paljon käytännöllisempi kuin Standard Junction.

Voimme asettaa edellä luotuihin junctioneihin ACL:n, POP:n tai AuthzRulen (6) seuraavalla JSON:lla. JSON:ssa määritellään pdadmin-kutsut, joihin tutustuimme luvussa 3.2. JSON:iin voisi laittaa niin monta ACL:ää, POP:ia tai AuthzRulea kuin tarvitaan. Tähän voitaisiin liittää myös muita komentoja, joihin tarvitaan pdadminia (5).

```
{
"admin_id":"käyttäjätunnus",
"admin_pwd":"salasana",
"commands":
[
"acl attach /WebSEAL/WEB1-testinst/@test1/ test1 test-ACL",
"authzrule attach /WebSEAL/WEB1-testinst/@test1/test1 test_AuthzRule",
"pop attach /WebSEAL/WEB1-testinst/@test1/ test1 test-POP"
]
}
```

Esimerkki 14. pdadmin-komennot JSON-muodossa, jotta voidaan liittää tarvittavat tiedot solmukohtiin.

Loimme yksinkertaisen pdadmin JSON:in, jota voi muokata tarpeiden mukaan ja johon on helppo lisätä tarvittavia pdadmin-komentoja.

Seuraavaksi luomme shell skriptin, jonka avulla luomme instanssin, konfiguroimme sen, liitämme instanssin sekä halutut ACL:t, POP:it ja AuthzRulet.

Seuraavaksi luomme esimerkki shell skriptin, jota voidaan käyttää luomaan kerralla instanssi, junction sekä kaikki tarvittavat tiedot. Instanssin luontiin JSON:ien avulla tarvitaan aiemmin luomamme skripti, joka sisälsi palvelimen yhteystiedot, tarvittavat jsonit sekä komennon, jolla ajamme nämä kaikki. Luomme seuraavassa esimerkissä skriptiin kaikki instanssin luontia varten tarvittavat komennot.

```
echo ’create instance’
./rest_cmd.sh reverseproxy POST revproxy_testinst.json
./rest_cmd.sh pending_changes/deploy GET empty
./rest_cmd.sh reverseproxy/testinst PUT restart.json
```

Esimerkki 15. Instanssin luontiin tarvittavat komennot.

Näillä komennoilla saamme luotua tyhjän instanssin, mutta haluamme lisätä vielä tarvittavia tietoja, jotta instanssimme toimii moitteettomasti ja on helppo konfiguroida.

Liitämme edellisten komentojen perään konfiguraatio JSON:it sekä tarvittavat komennot ja tulostukset selventämään, mitä tehdään.

```
echo 'set server-name'
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/server-name PUT server-name.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/worker-threads PUT worker-threads.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/request-body-max-read PUT request-body-max-read.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/request-max-cache PUT request-max-cache.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/utf8-url-support-enabled PUT utf8-url-support-enabled.json
```

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/suppress-server-identity PUT val_yes.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/suppress-backend-server-identity PUT val_yes.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/use-http-only-cookies PUT val_yes.json

echo 'set web-host-name'

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/v1 POST web-host-name_testinst.json

echo 'configure TFIM cluster'

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/tfim-cluster:fimCluster/entry_name/v1 POST fimcluster.json

echo configure test1 TFIM SSO settings

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/tfimsso:test1/entry_name/v1 POST tfimsso_test1.json.

echo local-response-redirect-uri stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/local-response-redirect/entry_name/local-response-redirect-uri PUT local-response-redirect-uri.json

echo local-response-redirect:test1 stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/local-response-redirect:test1/entry_name/local-response-redirect-uri PUT local-response-redirect-uri_test1.json

echo local-response-macros stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/local-response-macros/entry_name/v1 POST macros.json

echo eai stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-auth PUT val_both.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-user-id-header PUT eai-user-id-header.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-redir-url-priority PUT val_yes.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-pac-header PUT eai-pac-header.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-redir-url-header PUT eai-redir-url-header.json

echo system-environment-variables stanza

```
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/system-environment-variables/entry_name/LANG PUT lang.json
echo session stanza
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/session/entry_name/user-session-ids PUT val_yes.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/session/entry_name/use-same-session PUT val_yes.json
```

Esimerkki 16. Instanssille lähetettävät konfiguroinnit JSON-muodossa ja komennot.

Viimeisenä lähetetään deploy-käsky, jonka jälkeen instanssi pitää käynnistää uudelleen muutosten myötä. Seuraavassa esimerkissä on tarvittavat komennot luotu, jotta tämä saadaan aikaiseksi.

```
echo deploy changes and restart reverse proxy instance
./rest_cmd.sh pending_changes/deploy GET empty
./rest_cmd.sh reverseproxy/testinst PUT restart.json
```

Esimerkki 15. Deploy- ja restart-komennot.

Seuraavaksi skriptiin lisätään, miten liitetään Virtual Host ja Standard Junctiot instanssiin sekä pdadmin-komennot.

```
echo Virtual host junction
./rest_cmd.sh reverseproxy/testinst/junctions POST junction_test1.json
echo standard junction
./rest_cmd.sh reverseproxy/testinst/junctions POST junction_LRR.json
echo set acl and other needed rules
./rest_cmd.sh isam/pdadmin POST pdadmin.json
```

Esimerkki 16. Tarvittavat komennot junctioiden luomiseen ja sääntöjen liittämiseen.

Luontiskripti on nyt valmis, se löytyy kokonaisuudessaan liitteestä 2.

Kun kaikki on valmiina, voimme ajaa seuraavan komennon kansiosta, jossa kaikki tiedostomme ovat cygwinin avulla. Skriptit on nimetty näin helpottamaan käsitystä, mitä suoritetaan. Komennossa määrittelemme rest_cmd.sh-skriptillä, minne muutokset tehdään ja minkä tyylistä dataa lähetetään. create_and_configure.sh sisältää kaikki komennot, mitä aiemmin teimme esimerkeissä.

```
./rest_cmd.sh create_and_configure.sh
```

Esimerkki 17. Komento jolla instanssi luodaan.

Mikäli tarvitsemme lisää instansseja, voimme helposti käydä muuttamassa muutamaa tiedostoa ja sitä kautta luoda uusia instansseja nopeasti ja vähällä vaivalla.

4.1.3 Instanssin tietojen muokkaaminen

Mikäli meidän tarvitsee muokata jotain tietoja instanssissa, voimme tehdä sen helposti ajamalla vain tietyn komennon ja määrittelemällä tietyn JSON:in, joka vastaa komentoa. Teemme tässä luvussa muutaman esimerkin, miten voimme muokata tietoja tai lisätä uutta tarvittaessa. Kaikki muutokset ja lisäykset olisi hyvä lisätä myös luontiskriptiin, jotta se pysyy ajan tasalla.

Lisätään ensimmäisenä ISAM:n headeri kulkemaan mukana tietyllä ISAM:n arvolla. JSON on seuraavanlainen. Sillä saadaan lisättyä user-session-id headeri junctioniin.

```
{
"admin_id":"käyttäjätunnus",
"admin_pwd":"salasana",
"commands":
["object modify /WebSEAL/WEB1-testinst/@test1 set attribute HTTP-Tag-Value user_session_id=isam-session-id"]
}
```

Esimerkki 18. Headerin lisääminen junctioniin.

Näin helposti saamme yhden headerin kulkemaan joka kutsulla ISAM:n päässä. Seuraavalla komennolla voimme ajaa headerin muutoksen. Samaan JSON:iin olisi voitu lisätä myös muita komentoja, jos niitä olisi ollut tarve tehdä.

```
./rest_cmd.sh isam/pdadmin POST httpheader.json
```

Esimerkki 19. Komento headerin lisäämiseen.

Lähetämme ISAM:lle headerimme ja ISAM osaa laittaa sen junctioomme määrittelemillämme tiedoilla.

Seuraavaksi muokkaamme jo olemassa olevaa tietoa kuten, vaikka eai-auth tasoa, jonka aiemmin loimme. Nyt laitamme sen hyväksymään vain HTTPS-kutsuja. JSON on hyvin yksinkertainen. Siinä annetaan yksi avain-arvo-pari.

```
{"value":"https"}
```

Esimerkki 20. Avain-arvo-pari.

Muuta ei tarvita JSON:iin; seuraavaksi komento jolla ajamme ja määrittelemme minne muutos tapahtuu.

```
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-auth PUT https.json
```

Esimerkki 21. Komento jolla muuttaa instanssin tietoja.

Määrittelemällä PUT-toiminnon kirjoitamme yli sen, mitä siellä oli ja laitamme omamme sen tilalle. Näin yksinkertaista on muuttaa instanssimme tietoja, kun vain tiedetään mitä halutaan muuttaa. Samalla periaatteella pystymme muokkaamaan instanssin tietoja. Edellä mainitulla headerin lisäystoiminnolla voisimme myös muokata solmukohdann tietoja tai luoda uusia tarvittavia attribuutteja junctioihimme. Mikäli haluamme lisätä vaihtoehtoja, kuten vaikka trigger-urleja instansiimme, voisimme lisätä niitä POST-metodin avulla. Tämä lisää tietoja jo alkuperäisten perään eikä korvaa jo olemassa olevia.

4.1.4 Instanssin poistaminen

Poistaaksemme instanssin voimme tehdä poistoskriptin samalla tavalla kuin loimme aiemmin. Tähän tarvitaan muutamia JSON:eja ja teemme ne sekä lopullisen poistoskriptin tässä osuudessa. Teemme esimmäisenä JSON:in, jolla poistamme junctionista kaikki liitetyt ACL:ät, POP:it sekä AuthzRulet:it. Käytämme esimerkissä detach-komentoa, jonka avulla saamme otettua liitokset pois.

```
{
"admin_id":"käyttäjätunnus",
"admin_pwd":"salasana",
"commands":
[
"acl detach /WebSEAL/WEB1-testinst/@test1/ test1 test-ACL",
```

```
"authzrule detach /WebSEAL/WEB1-testinst/@test1/test1 test_AuthzRule",
"pop detach /WebSEAL/WEB1-testinst/@test1/ test1 test-POP"
]
}
```

Esimerkki 22. pdadmin-komentojen (5) poistaminen JSON:in avulla.

Näin voimme poistaa myös muita arvoja, joita olisi laitettu. Lisäämällä vain object modify -rivin ja määrittämällä, mistä ja mikä halutaan ottaa pois.

Seuraavaksi teemme JSON:in junction poistolle ja sen jälkeen itse instanssin poistolle. Käytämme tässä esimerkissä DELETE-kutsua, jonka avulla määrittelemme ISAM:lle poisto-operaation.

```
{ "admin_id":"käyttäjätunnus",
"admin_pwd":"salasana",
"commands":
[ server task test-instance-webseald-example delete /test1 ] }
```

Esimerkki 23. Junctionin poisto JSON.

Tämä JSON on yksinkertainen, koska tässä pitää vain määritellä komento, joka tässä tapauksessa on server task, ja sitten mistä instanssista poistetaan ja mikä junction.

Viimeisenä on vuorossa instanssin poistamisen JSON.

```
{ "admin_id":"käyttäjätunnus",
"admin_pwd":"salasana",
"commands":
[ object delete test-instance-webseald-example ] }
```

Esimerkki 24. Instanssin poisto JSON.

Määrittelemme vain, että kyseinen instanssi eli objekti poistetaan ja sitten instanssin tarkka nimi. Nyt kun JSON:it on tehty, voimme luoda poistoskriptistä esimerkin.

Poistoskripti on paljon lyhyempi kuin luontiskripti, koska kaikkia tietoja ei tarvitse erikseen asettaa ja poistaa. Aloitetaan laittamalla ensin ACL:n, POP:n ja AuthzRulen (5) poisto ja tulostus.

```
echo detach ACL, POP, AuthzRule
./rest_cmd.sh isam/pdadmin POST remove_pdadmin.json
echo delete junction
./rest_cmd.sh isam/pdadmin POST remove_junction.json
echo delete instance
./rest_cmd.sh isam/pdadmin POST remove_instance.json
```

Esimerkki 25. Instanssin ja sen osien poistamiseen tarvittavat komennot.

Näin helposti pystymme poistamaan instanssin ja siihen liitetyt osat vain muutamalla rivillä. Tämä helpottaa tulevaisuutta, jos joskus todetaan, että kyseistä instanssia ei tarvita, niin se voidaan poistaa helposti vain muutamalla komennolla.

Olemme nyt automatisoineet instanssin luomisen, sen muokkaamista ja poistamisen. Kuten Weeden mainitsee (6), automaatioon kannattaa panostaa. Kaikesta edellä luodusta työstä voimme vain pienillä muokkauksilla tehdä toisenlaisen instanssin. Mikäli haluaisimme, pystyisimme myös ketjuttamaan nämä luomiset ja poistamiset. Tämän avulla voisimme luoda monta uutta, muokata monia kerralla tai poistaa monta instanssia. JSON-muotoisen automaation ylläpidossa on paljon työtä, koska kaikki muutokset pitää käydä tekemässä joka paikkaan käsin. Tämä hidastaa omalta osaltaan työtä.

4.2 CA Release Automation XML

Tässä osiossa käymme läpi, miten voisimme tehdä asioita CARA:n (11) avulla käyttäen hyödyksi XML-pohjaista dataa. Teemme esimerkki XML:t, joita pystytään lähtemään jatkokehittämään. Teemme tässä osassa vain luontia varten tarvittavat XML-tiedot, toiset ilman muuttujia ja toiset muuttujien avulla. Tämä edistää automaatiota huomattavasti, koska on paljon helpompi sekä monikäyttöisempää käyttää tietoja, jotka tulevat muuttujista. Jatkossa voimme tehdä myös poistamiseen tarvittavat XML-tiedot.

4.2.1 Instanssin luominen

Tässä osiossa teemme CARA:lle tarvittavan XML-muotoisen ISAM:n instanssin luontiin tarkoitetun automaation. Aloitamme määrittelemällä tarvittavia toimenpiteitä kuten dep-

loy ja restart, jonka jälkeen jatkamme instanssin luomista. Tarvittavat tiedot täytyy määritellä request-tyylisiksi, niiden endpointit sekä mitä dataa tulee mukana. CARA muuttaa tiedot JSON-muotoon. Aloitetaan ensin luomalla XML:n alkuosuus, jossa on restart ja deploy. Voisimme määritellä tarvittavia muuttujia seuraavalla tavalla ${esimerkki}, jos meillä olisi paljon erilaisia instansseja ja muuttujia. Toiseen XML-tiedostoon voidaan laittaa viittaukset siihen, mitkä muuttujat saavat mitkäkin arvot. Teemme esimerkin myös tästä tapauksesta hieman myöhemmin.

```
<?xml version="1.0"?>
<automaatio>
 <requests class="common">
  <request name="restart">
   <endpoint>reverseproxy/test_instance</endpoint>
   <method>PUT</method>
   <data>{"operation": "restart"}</data>
  </request>
  <request name="deploy">
   <endpoint>pending_changes/deploy</endpoint>
   <method>GET</method>
   <data></data>
  </request>
 </requests>
```

Esimerkki 26. Instanssin luontia varten perus-XML.

Näillä komennoilla määrittelemme requestin, joka sisältää, mitä tehdään, minne tehdään, mikä metodi sekä mitä dataa operaatio sisältää. Datan voimme määrittää samalla tavalla kuin JSON:ssa eli avain-arvo-parilla. Seuraavaksi teemme instanssin luomiselle tarvittavat requestit ja instanssin konfiguraatiot.

Määrittelemme instanssin perustiedot data-settiin, jossa on avain-arvo-pareja, joista osataan luoda oikeilla tiedoilla instance.

```
 <requests class="reverseproxy" deploy="yes" restart="yes">
  <request name="initialise">
   <endpoint>reverseproxy</endpoint>
   <method>POST</method>
```

```
    <data>{ "inst_name": "test_instance", "host":"localhost", "listening_port":"8080", "domain": "Default",
"admin_id": "käyttäjätunnus", "admin_pwd": "salasana", "ssl_yn": "yes", "ssl_port": "636", "key_file":
"keyfile.kdb", "http_yn": "yes", "http_port": "80", "https_yn": "no", "https_port": "8443", "nw_inter-
face_yn": "yes", "ip_address":"127.0.0.1" }</data>
   </request>
 </requests>
```

Esimerkki 27. Instanssin tiedot, joilla se luodaan.

Seuraavaksi määritellään esimerkki, miten instanssiin voidaan lisätä tarkempia tietoja XML:n sisälle. Tässäkin tehdään request, jossa määritellään, minne asetetaan ja millä metodilla sekä mikä data laitetaan mukaan. Samalla tavalla pystyisimme lisäämään kaikki tarvittavat tiedot.

```
  <requests class="server" deploy="no" restart="no">
   <request name="server-name">
    <endpoint>reverseproxy/test_instance/configuration/stanza/server/entry_name/server-name</end-
point>
    <method>PUT</method>
    <data>{ "value": "webseald-web1/test_instance" }</data>
   </request>
 </requests>
```

Esimerkki 28. Tiedon lisääminen instanssinluonti-XML:ään.

Teimme aiemmin JSON:in avulla TFIM-klusterin. Nyt teemme saman mutta XML:n avulla.

```
<requests class="tfim-cluster" deploy="no" restart="no">
   <request name="tfim-cluster-init">
    <endpoint>reverseproxy/test_instance/configuration/stanza/tfim-cluster:fimCluster/</endpoint>
    <method>POST</method>
    <data></data>
   </request>
   <request name="tfim-cluster-entry">
    <endpoint>reverseproxy/test_instance/configuration/stanza/tfim-cluster:fimCluster/en-
try_name/v1</endpoint>
    <method>POST</method>
```

```
    <data>{ entries: [ ["server","9, TFIM_server_address ["ssl-keyfile-stash","keyfile.sth"], ["ssl-keyfile","keyfile.kdb"], ["basic-auth-user", "käyttäjätunnus"], ["basic-auth-passwd", "salasana"], ["handle-pool-size","10"], ["handle-idle-timeout","30"], ["timeout","30"] ]}</data>
   </request>
  </requests>
</automaatio>
```

Esimerkki 29. TFIM -kusterin lisääminen instanssiin.

Nyt olemme näyttäneet, miten voisimme luoda instanssin ja liittää siihen tietoja CARA:n avulla XML-muotoisella datalla. Seuraavaksi luomme esimerkin junctionin lisäyksestä. Ensiksi määritellään restart ja deploy ja seuraavaksi päästään käsiksi itse junctioniin.

```
<?xml version="1.0"?>
<automaatio>
 <requests class="common">
  <request name="restart">
   <endpoint>reverseproxy/test_instance</endpoint>
   <method>PUT</method>
   <data>{"operation": "restart"}</data>
  </request>
  <request name="deploy">
   <endpoint>pending_changes/deploy</endpoint>
   <method>GET</method>
   <data/>
  </request>
 </requests>
```

Esimerkki 30. Deploy ja restart junctionin luomiseen.

Pystymme asettamaan kaikki tarvittavat tiedot myös XML:ään aivan kuten JSON:iin. Seuraavassa esimerkissä laitamme junctioniin tarvittavat tiedot.

```
 <requests class="init" deploy="yes" restart="no">
  <request name="web-host-name">
   <endpoint>reverseproxy/test_instance/configuration/stanza/server/entry_name/web-host-name</endpoint>
   <method>POST</method>
```

```
    <data>{ "value": "test_junction" }</data>
   </request>
  </requests>
<requests class="junctions" deploy="no" restart="no">
   <request name="junction">
    <endpoint>reverseproxy/test_instance/junctions</endpoint>
    <method>POST</method>
    <data>{ "server_hostname":"webseald-WEB1", "junction_point":"test_junction", "junction_type":"tcp", "junction_hard_limit":"0", "junction_soft_limit":"0", "basic_auth_mode":"filter", "client_ip_http":"yes", "tfim_sso":"yes", "request_encoding":"utf8_uri", "virtual_hostname":"test_junction", "case_sensitive_url":"yes", "server_port":"80", "http_port":"80”</data>
   </request>
  </requests>
```

Esimerkki 31. Junctioniin tarvittavat tiedot.

Voimme liittää myös pdadmin-komentoja näihin XML-tietoihin kuten seuraavasta esimerkistä käy ilmi.

```
 <requests class="pdadmin" deploy="no" restart="no" runon="policyhost">
   <request name="set-rules">
    <endpoint>isam/pdadmin</endpoint>
    <method>POST</method>
    <data>{"admin_id": "käyttäjätunnus", "admin_pwd": "salasana", "commands": [
"acl attach /WebSEAL/ webseald-WEB1/test_instance/@test_junction test-ACL”,
"pop attach /WebSEAL/webseald-WEB1/test_instance/@test_junction test-POP",
"authzrule attach /WebSEAL/webseald-WEB1/test_instance/@test_junction test-AuthzRule"] }</data>
   </request>
  </requests>
</automaatio>
```

Esimerkki 32. pdadmin-komentojen lisääminen XML:ään.

Näin olemme luoneet esimerkki junctionin ja liittäneet sen instanssiimme. Olemme myös liittäneet erilaisia tietoja junctioniin XML:n avulla.

4.2.2 Jatkokehitys

Voisimme myös määritellä muuttujia ja niiden arvoja, jotta uudelleenkäytettävyys ja ylläpidettävyys paranevat. Tässä luvussa teemme muutaman muuttujan ja katsomme, miten niitä voisi käyttää hyödyksi RA:ssa. Luomme ensin XML-muotoisen datan, jossa on määritelty muutama muuttuja sekä mitä XML-tiedostoa kutsutaan, johon muuttujat osoittavat. Sen jälkeen asetamme kyseisiä arvoja edellä luomiimme XML-esimerkkeihin ja näemme, miten muuttujia voidaan asettaa.

Luomme seuraavaksi esimerkin, joka käyttää hyödyksi toista XML-tiedostoa. Tässä esimerkkikoodissa asetamme muuttujille arvot. Nämä tulevat tarpeeseen, kun luomme seuraavia esimerkkejä. Voimme määrittää toiseen XML-tiedostoon, mitkä XML-tiedostot luetaan ja missä järjestyksessä kuten seuraava esimerkki osoittaa.

```
<?xml version="1.0"?>
<automaatio>
 <templates>
  <instance>
   <create>create.xml</create>
  </instance>
  <virtualhost>
   <create> virtualhost/create.xml</create>
  </virtualhost>
 </templates>
```

Esimerkki 33. Instanssin ja junctionin luontia varten tarvittavat tiedostot.

Seuraavaksi määrittelemme muutaman muuttujan, joita voidaan käyttää edellä luoduissa XML-tiedostoissa. Teemme esimerkin myöhemmin, missä osoitetaan, miten seuraavan, esimerkin muuttujia voidaan käyttää edellisissä XML-tiedostoissa hyödyksi.

```
 <properties>
   <property name="proxyhost.name">WEB1</property>
   <property name="object_name_prefix">test-webseald-</property>
 </properties>
```

Esimerkki 34. Muuttujien määrittelyä.

Pystymme määrittelemään tietoja instanssiin, kuten seuraavassa esimerkissä osoitetaan.

```
 <instances>
  <instance name="test_instance" description="description">
   <line name="test_instance" host="line1" ip_address="127.0.0.1" listening_port="8080"/>
  </instance>
 </instances>
</automaatio>
```

Esimerkki 35. Instanssin tietojen määrittely muuttujiksi.

Olemme nyt määritelleet erilaisia tietoja yhteen XML-tiedostoon, josta voimme antaa arvoja muuttujina edellä luotuihin esimerkkeihin vain pienillä muutoksilla. Seuraavaksi käymme läpi muutokset sekä miten ne auttavat automaatiota.

Aloitamme instanssin muuttujista ja muokkaamme esimerkkiä 27 sen verran, että se ottaa vastaan muuttujia. Tämä auttaa todella paljon uudelleen käytettävyyttä sekä sitä, että voidaan luoda monta instanssia samalla kertaa. Pystymme siis määrittelemään monta erilaista instanssia ja sen tiedot esimerkin 35 mukaisesti. Tiedot osataan käydä läpi yksi kohta kerrallaan. Seuraavasta esimerkistä käy ilmi, miten muuttujia asetetaan XML-tiedostoon. Tämä on suhteellisen helppo ja yksinkertainen prosessi.

```
<requests class="reverseproxy" deploy="yes" restart="yes">
  <request name="initialise">
   <endpoint>reverseproxy</endpoint>
   <method>POST</method>
   <data>{ "inst_name": "${instance.name}", "host":"${host.name}", "listening_port":"${listening_port}",
"domain": "Default", "admin_id": "${käyttäjätunnus}", "admin_pwd": "${salasana}", "ssl_yn": "yes",
"ssl_port": "636", "key_file": "${keyfile}", "http_yn": "yes", "http_port": "80", "https_yn": "no",
"https_port": "8443", "nw_interface_yn": "yes", "ip_address":"${ip_address}" }</data>
  </request>
 </requests>
```

Esimerkki 36. Instanssin muuttujien asettaminen XML-tiedostoon.

Tämän jälkeen voimme tehdä samat muutamalle tiedolle, mitä määrittelimme aiemmin esimerkissä 28, pystymme määrittelemään muuttujia myös endpoint-osoitteisiin.

```
<requests class="server" deploy="no" restart="no">
  <request name="server-name">
   <endpoint>reverseproxy/${instance.name}/configuration/stanza/server/entry_name/server-name</endpoint>
   <method>PUT</method>
   <data>{ "value": "webseald-web1/${instance.name}" }</data>
  </request>
 </requests>
```

Esimerkki 37. Tietojen asettaminen muuttujien avulla.

Laitamme junctionin tietoja myös muuttujiin samalla tavalla kuin aiemmin laitoimme muihin tiedostoihin. Se käy seuraavan esimerkin mukaisesti.

```
<requests class="junctions" deploy="no" restart="no">
  <request name="junction">
   <endpoint>reverseproxy/${instance.name}/junctions</endpoint>
   <method>POST</method>
   <data>{ "server_hostname":"${ServerName}", "junction_point":"${Junction}", "junction_type":"tcp", "junction_hard_limit":"0", "junction_soft_limit":"0", "basic_auth_mode":"filter", "client_ip_http":"yes", "tfim_sso":"yes", "request_encoding":"utf8_uri", "virtual_hostname":"${JunctionAddress}", "case_sensitive_url":"yes", "server_port":"80", "http_port":"80"}</data>
  </request>
 </requests>
```

Esimerkki 37. Junction tietojen muokkaaminen muuttujiksi.

Liitämme vielä esimerkin, miten määrittelemme muuttujia myös pdadmin-kutsuille. Tämä auttaa liittämään kaikki tarvittavat säännöt, joita ISAM sitten osaa toteuttaa.

```
<requests class="pdadmin" deploy="no" restart="no" runon="policyhost">
  <request name="set-rules">
```

```
    <endpoint>isam/pdadmin</endpoint>
    <method>POST</method>
    <data>{"admin_id": "käyttäjätunnus", "admin_pwd": "salasana", "commands": [
"acl attach /WebSEAL/ ${object_name}${instance.name}/@${JunctionName} test-ACL”,
"pop attach /WebSEAL/${object_name}${instance.name}/@${JunctionName} junction test-POP",
"authzrule attach /WebSEAL/${object_name}${instance.name}/@${JunctionName} test-AuthzRule"]
}</data>
  </request>
 </requests>
</automaatio>
```

Esimerkki 38. pdadmin-komennot muuttujien kanssa.

Näin olemme luoneet tarvittavat esimerkit ja skriptit, joita automaatio vaatii toimiakseen tehokkaasti. Automaation edistämiseksi kaikkia skriptejä ja esimerkkejä voi jatkokehittää sellaisiksi, että ne toimivat itselleen parhaalla tavalla. Edellä luotujen JSON- ja XML-skriptien pohjalta voitaisiin luoda myös solmukohtien ja instanssien poistamiseen tarvittavat XML-muotoiset CARA:lle annettavat skriptit.

Kaikki edellä luodut JSON- ja XML-pohjaiset tiedot ovat käyttövalmiita ja niiden avulla voidaan toteuttaa automaatiota. Automaation kehittämiseen kannattaa panostaa ja, kuten huomaamme, sen tekemisessä on paljon työtä. On päätös kumpaan lähestymistapaan tahansa, niin niillä on omat hyötynsä. XML-pohjaiseen tietoon voimme määritellä muuttujia, jotka auttavat ylläpidettävyydessä. Sitä on myös paljon siistimpi ylläpitää koska tiedot ovat vain muutamassa tiedostossa.

Automaatio ratkaisee monta ongelmaa, joita tavallisissa ympäristöissä saattaa ilmentyä. Sen avulla pystytään luomaan ja muokkaamaan tietoja ja tiedostoja vain muutamalla pienellä muutoksella. Automaatio vähentää todella paljon inhimillisiä virheitä. Tämä tehostaa huomattavasti toimintaa ja virhemarginaalit pienenevät

5 Yhteenveto

Insinöörityössä tutustuttiin ISAM:n toimintaan ja siihen liittyvien yleisten osien automatisointiin sekä pohdittiin automaation hyötyjä näitä tehdessä. Tietoturvan näkökulmasta

yksi sovellus, jota on helppo hallita sekä konfiguroida monipuolisesti on saanut suurta suosiota yritysten keskuudessa. ISAM tarjoaa yhdessä paketissa kaiken tarpeellisen ja siihen pystytään myös itse lisäämään toiminnallisuutta esimerkiksi java-koodin avulla.

Työssä perehdyttiin erityisesti ISAM:n yleisimpiin toimintoihin. Koska ISAM on IBM;n tuote löytyy koulutusmateriaalia paljon eritasoisille osaajille. Helpoin tapa aloittaa on kuitenkin, jos ISAM on asennettu ja sitä päästään konfiguroimaan sekä luoman instansseja ja tutkimaan sen toimintaa laajemmin. Uusina asioina on opittu muun muassa instanssin luomiseen liittyviä asioita, uusia metodikutsuja sekä automaation kehityksessä käytettyjen XML-tietoja ja niiden luomista.

Työn aiheeksi valittiin ISAM:n automaatio, koska haluttiin tutustua, miten sitä pystytään automatisoimaan sekä mitä kaikkea ISAM pitää sisällään. Tarkoituksena oli kehittää skriptit ja luoda tarvittavat JSON-muotoiset datatiedot sekä tutustua, miten automaation pystyisi toteuttamaan XML-tiedon avulla käyttäen CARA:a. Kehityksen aikana opittiin runsaasti, mitä kaikkea ISAM pitää sisällään sekä missä muodossa ja minkälaisilla kutsuilla tietoja voidaan välittää, jotta saadaan haluttu lopputulos. Automaation luonti vaati paljon aikaa sekä kärsivällisyyttä, koska piti opetella todella paljon uutta ja pohtia, mitä voidaan laittaa ja minkä muotoisessa tiedossa. Tällaisia kokonaisia automaatioskriptejä ei oltu aiemmin toteutettu, joten esimerkkejä ei ollut tarjolla. Tämä vaati aikaa opetella kokonaisuuden hallintaa ja mitä kaikkia osa-alueita pitää ottaa huomioon. Oppimista helpotti, kun löytyi paljon apuja muutamalta henkilöltä, jotka tekevät ISAM:n kanssa töitä, IBM:n sivuilta sekä heidän kehittäjien luomasta ISAM-oppikirjoista.

Insinöörityön tuloksena kehitettiin JSON-muotoiset data-tiedostot, shell-skriptit, jotka käyttävät näitä hyödykseen sekä XML-pohjaista automaatiota CARA:lle. Tarkoituksena oli kehittää instanssien luomiselle sekä hallinnalle käyttöä ja ylläpidettävyyttä helpottavia skriptejä. Instanssien luominen ja ylläpitäminen helpottuvat huomattavasti, kun pystytään luomaan ne automaattisesti. Työlle asetetut tavoitteet toteutuivat hienosti ja skriptejä on päästy kokeilemaan ja kehittämään lisää.

Lähteet

1. IBM Security Access Manager. Verkkodokumentti. <http://www-03.ibm.com/software/products/fi/access-mgr> Luettu 10.02.2017.

2. Tieturvan investoinnit nyt ja tulevaisuudessa. Verkkodokumentti. <https://www.cybersecurityintelligence.com/blog/the-brave-new-world-of-cyber-security-362.html> Luettu 17.02.2017.

3. ISAM opas. Verkkodokumentti. <http://www.ibm.com/support/knowledgecenter/SSPREK_7.0.0/com.ibm.isam.doc_70/ameb_webseal_guide/reference/ref_srvr_tsk_cmd_jct.html> Luettu 06.03.2017.

4. ISAM opas. Verkkodokumentti. <https://www.ibm.com/support/knowledgecenter/SSPREK_7.0.0/com.ibm.isam.doc_70/ameb_webseal_guide/reference/ref_srvr_tsk_rmv.html> Luettu 10.03.2017.

5. Pdadmin komentoja. Verkkodokumentti. <https://www.ibm.com/support/knowledgecenter/SSXJVR_2.0.0/com.ibm.sso.doc/cmdref/concept/con_pdadmin_cmd.html> Luettu 13.03.2017.

6. Shane Weeden. ISAM konfiguraatiota. Verkkodokumentti. <https://www.ibm.com/blogs/sweeden/an-introduction-to-automated-configuration-in-ibm-security-access-manager-v9/> Luettu 13.03.2017.

7. https://www.ca.com/us/products/ca-release-automation.html?intcmp=headernav

8. ISAM REST-rajapinta kuvaus. Verkkodokumentti. <https://www.google.fi/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjw9N3P1tbSAhUL-DywKHYo5BMoQFggIMAE&url=https%3A%2F%2Fwww.ibm.com%2Fdeveloperworks%2Fcommunity%2Fforums%2Fajax%2Fdownload%2Fbf3e76cf-86fe-46e2-8ec1-cb5c36a0ecd2%2Ff91772e3-1e76-43e3-9676-df28810467ef%2FISAM9-Basic%2520Administration%2520using%2520ReST-TAPI_v1.1.pdf&usg=AFQjCNEx2IkxGzvKwZ3BZGIIb26HtvNkxA> Luettu 18.03.2017

9. IBM Federation Cookbook. Verkkodokumentti. <https://www.ibm.com/developerworks/community/wikis/form/anonymous/api/wiki/bf636498-bc5a-442b-a6e5-3c799035ba5b/page/82b4335f-8526-4db2-904c-e5976ab9766e/attachment/30fefce8-b84f-4eaf-8b19-da90ed5c4c33/media/ISAM9%20FederationCookbook%2020160503.pdf> Luettu 13.03.2017.

10. ISAM instanssin luontia. Verkkodokumentti. <https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W746177d414b9_4c5f_9095_5b8657ff8e9d/page/IBM

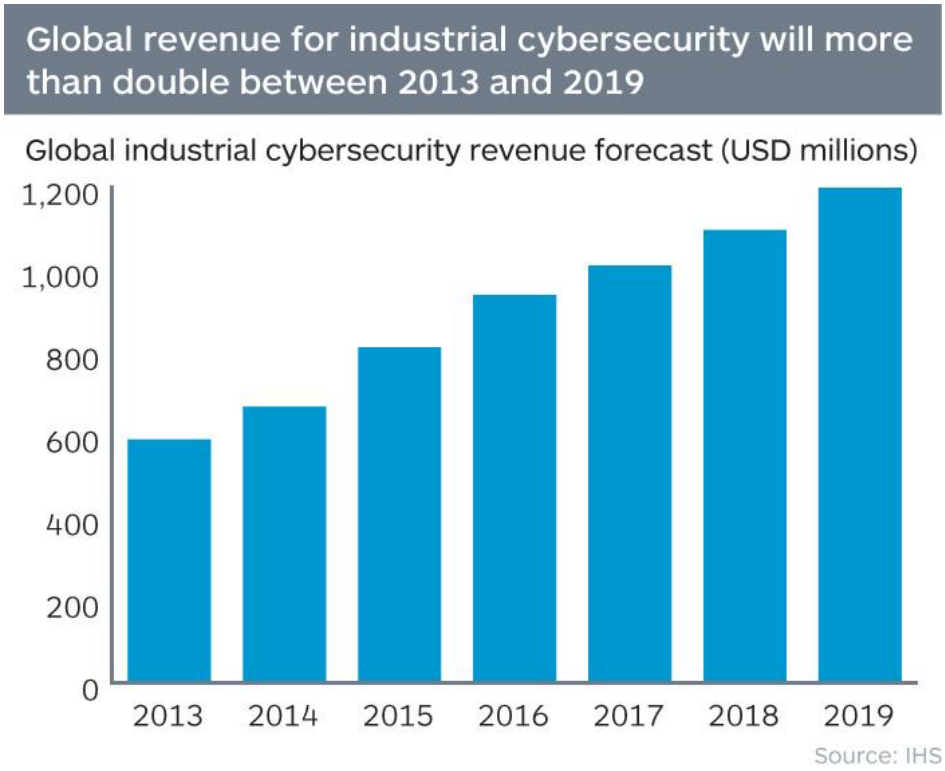
%20Security%20Access%20Manager%20(ISAM)%20Reverse%20Proxy%20Scenario> Luettu 20.03.2017.

11. CA Release Automation lyhyesti. Verkkodokumentti. <https://www.ca.com/content/dam/ca/us/files/data-sheet/ca-release-automation.pdf> Luettu 20.03.2017.

12. Tietoruvan suunnittelua. Verkkodokumentti <https://www.tietosuoja-lehti.fi/index.php?mid=2&pid=32&aid=3042> Luettu 15.04.2017.

Liite 1. Tietoturvan merkitys nykyään ja tulevaisuudessa

Tietoturva on nykyään kasvava alue ohjelmistokehityksessä (2), koska on ollut paljon tieturvahyökkäyksiä. Näistä tapauksista uutisoidaan laajasti ja ne merkitsevät yrityksille maineen sekä rahan menetystä. Kuten voimme todeta seuraavasta kuvasta (2), tietoturvaan panostetaan koko ajan entistä enemmän ja se tulee vain kasvamaan tulevaisuudessa.



Liite 2. create_and_configure.sh skripti kokonaisuudessaan

Valmis skripti kokonaisuudessaan joka käyttää hyödykseen JSON pohjaista dataa. Tätä on helppo muokata yksittäisen instanssin tarpeisiin ja antaa hyvän pohjan skriptien kehittämiselle.

```
echo 'create reverse proxy'
./rest_cmd.sh reverseproxy POST revproxy_testinst.json
./rest_cmd.sh pending_changes/deploy GET empty
./rest_cmd.sh reverseproxy/testinst PUT restart.json
echo 'set server-name'
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/server-name PUT server-name.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/worker-threads PUT worker-threads.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/request-body-max-read PUT request-body-max-read.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/request-max-cache PUT request-max-cache.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/utf8-url-support-enabled PUT utf8-url-support-enabled.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/suppress-server-identity PUT val_yes.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/suppress-backend-server-identity PUT val_yes.json
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/use-http-only-cookies PUT val_yes.json
echo 'set web-host-name'
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/server/entry_name/v1 POST web-host-name_testinst.json
echo 'junction stanza'
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/junction/entry_name/match-vhj-first PUT val_no.json
echo 'configure TFIM cluster'
./rest_cmd.sh reverseproxy/testinst/configuration/stanza/tfim-cluster:fimCluster/entry_name/v1 POST fimcluster.json
echo configure test1 TFIM SSO settings
```

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/tfimsso:test1/entry_name/v1 POST tfimsso_test1.json

echo update EAI trigger URLs

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai-trigger-urls/entry_name/v1 POST triggers_testinst.json

echo acnt-mgt stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/acnt-mgt/entry_name/enable-local-response-redirect PUT val_yes.json

echo local-response-redirect-uri stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/local-response-redirect/entry_name/local-response-redirect-uri PUT local-response-redirect-uri.json

echo local-response-redirect:test1 stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/local-response-redirect:test1/entry_name/local-response-redirect-uri PUT local-response-redirect-uri_test1.json

echo local-response-macros stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/local-response-macros/entry_name/v1 POST macros.json

echo authentication-levels stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/authentication-levels/entry_name/v1 POST authentication-levels.json

echo eai stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-auth PUT val_both.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-user-id-header PUT eai-user-id-header.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-redir-url-priority PUT val_yes.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-pac-header PUT eai-pac-header.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/eai/entry_name/eai-redir-url-header PUT eai-redir-url-header.json

echo system-environment-variables stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/system-environment-variables/entry_name/LANG PUT lang.json

echo session stanza

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/session/entry_name/user-session-ids PUT val_yes.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/session/entry_name/use-same-session PUT val_yes.json

./rest_cmd.sh reverseproxy/testinst/configuration/stanza/session/entry_name/shared-domain-cookie PUT val_yes.json

echo deploy changes and restart reverse proxy instance

./rest_cmd.sh pending_changes/deploy GET empty

./rest_cmd.sh reverseproxy/testinst PUT restart.json

echo test1 junction

./rest_cmd.sh reverseproxy/testinst/junctions POST junction_test1.json

echo /LRR junction

./rest_cmd.sh reverseproxy/testinst/junctions POST junction_LRR.json

echo set rules

./rest_cmd.sh isam/pdadmin POST pdadmin.json