Eero Hirvonen

Vahva autentikointi BIG-IP Access Policy Managerilla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

25.2.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Eero Hirvonen Vahva autentikointi BIG-IP Access Policy Managerilla 47 sivua + 5 liitettä 25.2.2013
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja(t)	Yliopettaja Janne Salonen Tekninen asiantuntija Turo Siira

Insinöörityössä tarkasteltiin erilaisia autentikointimenetelmiä, esiteltiin F5 Networksin BIG-IP Application Delivery Controller -laite ja sen tuomia etuja autentikointiin. Lisäksi luotiin katsaus tämän hetken suurimpaan ongelmaan päätelaite- ja käyttäjäntunnistuksessa, BYOD:iin.

Teknisessä osuudessa rakennettiin DCC Oy:lle testiympäristö, jossa voidaan esitellä Google Authenticator -autentikointia ja sähköpostilla lähetettävää kertakäyttösalasanaa sekä tehtiin Proof of Concept TUPAS-tunnistautumisesta. Toteutuksessa käytettiin F5 Networksin BIG-IP-laitteita ja Access Policy Manager -moduulia. Apuna hyödynnettiin erilaisia virtuaalipalvelimia.

Testiympäristöllä osoitettiin ulkoisen autentikointijärjestelmän tuomia etuja. Ulkoisen autentikointijärjestelmän avulla vähennettiin ylläpidettävien komponenttien määrää ja mahdollistettiin saman autentikointimekanismin käyttäminen useassa eri sovelluksessa ilman merkittäviä muutoksia sovellukseen. Eri autentikointimekanismien yhdistämisen avulla voitiin tuottaa vahva autentikointi.

Toteutusten avulla varmistettiin Access Policy Managerin monipuolisuus ja sen mahdollisuudet toteuttaa erilaisia autentikointijärjestelmiä. Yleisesti haastavaksi koettu TUPAS-tunnistautuminen pystyttiin rakentamaan yhden virtuaaliserverin päälle, vaikkei sitä vielä pystytty liittämään alkuperäisen suunnitelman mukaan APM:ään.

Testiympäristön avulla DCC Oy pystyy esittelemään osaamistaan BIG-IP Access Policy Managerin parissa ja näin vahvistaa asemaansa tietoliikenteen ja verkkojen erikoisosaajana.

Avainsanat	Autentikointi, BIG-IP, TUPAS, kertakäyttösalasana, Access Policy Manager

Author(s) Title Number of Pages Date	Eero Hirvonen Strong Authentication Using BIG-IP Access Policy Manager 47 pages + 5 appendices 25 February 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Data networks
Instructor(s)	Janne Salonen, Principal Lecturer Turo Siira, Technical Expert

This thesis focuses on different authentication methods, takes a closer look at F5 Networks' BIG-IP Application Delivery Controller systems and what benefits it brings to authentication. In addition the thesis reflects on today's biggest challenge in user and end device authentication, BYOD.

A test environment to showcase different authentication methods was built for DCC Oy. The authentication methods include such methods as Google Authenticator, One Time Password using email. A Proof of Concept was also made for bank credential authentication, TUPAS. The test environment was built using F5 Networks' BIG-IP with Access Policy Manager module. Virtual machines were used where necessary.

The purpose of the test environment was to show the benefits of moving the authentication from the application to an external authentication proxy. Using an external authentication proxy reduced the number of maintained components and made it possible to use the same authentication mechanism for multiple applications without having to make changes in the application itself. Strong authentication can be achieved by combining different authentication mechanisms.

BIG-IP and Access Policy Manager proved to be flexible solutions for using different authentication mechanisms. A Proof of Concept of TUPAS authentication was implemented successfully. This proved that it is possible to implement challenging authentication mechanisms on F5 Networks' BIG-IP platform.

With the use of their new test environment DCC Oy can showcase their knowledge of the BIG-IP Access Policy Manager and strengthen their position as experts on data networking.

Keywords	Authentication, BIG-IP, TUPAS, One Time Password, Access Policy Manager

Sisällys

Lyhenteet

1 Johdanto 1

2 Autentikointi yleisesti 2

2.1 Autentikointimenetelmiä 3
2.1.1 Käyttäjähakemistot 3
2.1.2 Kaksitasoinen autentikointi 3
2.1.3 Varmenteet 5
2.1.4 Single Sign-On 6

3 BIG-IP ja Access Policy Manager 6

4 Käytännön toteutukset 11

4.1 Google Authenticator 12
4.1.1 Toiminta 12
4.1.2 Active Directory ja Access Policy Manager 13
4.1.3 Kertakäyttösalana- ja tunnuksen luonti - iRule 14
4.1.4 Vuokaavio 16
4.1.5 Virtuaaliserverit 19

4.2 Email OTP 24
4.2.1 Toiminta 24
4.2.2 Kertakäyttösalasana-iRule ja sähköposti-skripti 25
4.2.3 Liittäminen Access Policyyn ja SNMP-trap 27
4.2.4 Viimeistely 28

4.3 TUPAS-tunnistautuminen 31
4.3.1 Toiminta 32
4.3.2 Toteutus Access Policy Managerin kanssa 33
4.3.3 Toteutus ilman Access Policy Manageria 36

5 BYOD 41

5.1 Uhat ja ongelmat 42

5.2 APM ja BYOD 43

6 Yhteenveto 43

Liitteet

Liite 1. Google_auth_verify_apm - iRule

Liite 2. Google_auth_soft_token_gen - iRule

Liite 3. OTP-iRule

Liite 4. Email-skripti

Liite 5. TUPAS-iRule

Lyhenteet

AAA	Authentication, authorization and accounting, tunnistus- ja pääsynhallinta-protokolla.
AD	Active Directory, käyttäjähakemisto.
ADC	Application Delivery Controller, laite joka hoitaa kuormanjakoa.
AFM	Advanced Firewall Manager, BIG-IP:n palomuurimoduuli.
APM	Access Policy Manager, BIG-IP:n pääsynhallintamoduuli.
ASM	Application Security Manager, BIG-IP:n sovelluspalomuurimoduuli.
BYOD	Bring Your Own Device, omien päätelaitteden käyttäminen työntekoon.
CA	Certificate Authority, taho joka varmentaa sähköisiä varmenteita.
DMZ	Demilitarisoitu alue, turva-alue sisäverkon ja julkisen verkon välillä.
IdP	Identity Provider, tunnistaa käyttäjän SAML:ssa.
LTM	Local Traffic Manager, BIG-IP:n kuormanjakomoduuli.
MAM	Mobile Application Management, mobiilisovellusten hallinta.
MDM	Mobile Device Management, mobiililaitteiden hallinta.
NTLM	NT Lan Manager, Microsoftin protokolla, jota käytettään muun muassa autentikointiin.
NTP	Network Time Protocol, verkon kellopalvelu.
OTP	One Time Password, kertakäyttösalasana.
SAML	Security Assertion Markup Language, autentikointiprotokolla.
SNAT	Secure Network Address Translation, yksi suuntainen IP-osoitteenmuutos.
SNMP	Simple Network Management Protocol, protokolla jolla voidaan hallita laitteitta IP-verkoissa.
SP	Service Provider, pyytää käyttäjäntunnistusta SAML:ssa ja kuljettaa tietoa eteenpäin.
SSL	Secure Sockets Layer, tekniikka, jolla salataan selaimen ja web-palvelimen välinen liikenne.
SSO	Single Sign-On, kirjaudutaan automaattisesti useampaan palveluun.
TFA	Two Factor Authentication, kaksitasoinen autentikointi.
TMOS	Traffic Management Operation System, BIG-IP:n mikrokernel.
TOTP	Time based one time password, aikaan perustuva kertakäyttösalasana.
VDI	Virtual Desktop Interface, virtuaalityöpöytä.
WOM	WAN Optimization Module, BIG-IP:n WAN-yhteyksien optimointi -moduuli.

1 Johdanto

Yritysten palveluiden siirtyessä kasvavissa määrin verkkoon on tärkeää pystyä autentikoimaan käyttäjä luotettavasti. Riittävän tunnistautumisen ansioista käyttäjälle voidaan taata tietoturvaa ja riittävät käyttöoikeudet palveluun. Yritys puolestaan pystyy yksilöllistämään palvelujaan tunnistuksen perusteella. Lisäksi autentikointi ja sen lokittaminen lisäävät palveluiden ja niiden käyttämän datan tietoturvaa.

Sovellusten käyttöliittymät ovat tyypillisesti HTTP-selainpohjaisia. Tällä saavutetaan riippumattomuus päätelaitteesta ja myös sovellusalustasta sekä helpotetaan julkaisua erityyppisille käyttäjille. Käyttäjäntunnistus voidaan myös helpommin irrottaa sovelluksista.

Käyttäjäntunnistusmenetelmiä on useita erilaisia. Ne voivat vaihdella yksinkertaisista käyttäjätunnuksen ja salasanan sisältävistä käyttäjähakemistopohjaisista menetelmistä monimutkaisiin kertakäyttösalasanoihin, jotka toimitetaan käyttäjälle esimerkiksi SMS-viestillä. Erilaisia autentikointimenetelmiä voidaan myös yhdistellä. Näin tekemällä vahvistetaan autentikointiprosessia, luodaan ja varmistetaan vahva tietoturva. Varmenteiden avulla voidaan tunnistaa palvelu tai käytetty laite.

Teknologian kehitys tuo uusia haasteita ja käyttäjän ja päätelaitteen tunnistamisen merkitys korostuu entisestään. Älypuhelimien ja tablettien yleistyessä kovaa vauhtia, pitää pääsynhallintajärjestelmien pystyä tunnistamaan nämä ja takaamaan haluttu toimivuus ja tietoturva. BYOD, Bring Your Own Device, eli omien päätelaitteiden käyttäminen pakottaa yritykset miettimään omia toimintatapojaan, kun kontrolli verkkoon liittyvistä laitteista ei välttämättä ole perinteisen mallin mukaan yrityksen IT-organisaatiolla.

Itse sovellukseen ei välttämättä tarvitse koodata erilaisia autentikontimenetelmiä, jos käyttäjäntunnistus hoidetaan erillisellä laitteella. Tällöin laite toimii niin sanottuna pääsynhallinta- tai käyttäjäntunnistusproxyna. Autentikoinnin ulkoistaminen yksinkertaistaa sovelluksen koodia ja parantaa sen toimivuutta, kun sovelluksen ei tarvitse huolehtia käyttäjäntunnistuksesta. Parhaimmillaan erillinen autentikoinitilaite voi poistaa tarpeen koodata käyttäjäntunnistus useisiin sovelluksiin ja mahdollistaa kertakirjautumisen.

Tässä insinöörityössä perehdytään erilaisiin käyttäjäntunnistusmenetelmiin, toteutetaan niitä käyttäen F5 Networksin BIG-IP Access Policy Manageria. Lisäksi pohditaan BYOD-ilmiötä ja sen tuomia mahdollisuuksia ja uhkia, sekä miten Access Policy Manager voi auttaa yritystä tämän BYOD-politiikan kanssa.

2 Autentikointi yleisesti

Autentikoinnilla tarkoitetaan käyttäjäntunnistusta sovellusten ja palveluiden käyttöä varten. Sovellukset pystyvät tarjoamaan kohdistettuja palveluita tunnistetuille käyttäjille. Käyttäjäntunnistuksen siirtäminen ulkoiseen laitteeseen poistaa tarpeen koodata autentikointikoodi itse sovellukseen. Tämä helpottaa sovelluksen toteutusta ja mahdollistaa joustavien ja useiden erilaisten autentikointimenetelmien käyttämisen.

Vahvan autentikoinnin määritelmä vaihtelee puhujasta riippuen. Esimerkiksi amerikkalaiset mieltävät kaksitasoisen autentikoinnin, Two-Factor Authentication (TFA), osaksi vahvaa autentikointia. Suomen valtiohallinnon tietoturvallisuuden johtoryhmä VAHTI puolestaan listaa seuraavia menetelmiä esimerkkeinä vahvasta tunnistautumisesta:

- toimikortti
- token-pohjainen tunnistekortti
- puhelimen mobiilivarmenne
- tunnetut ja luotettaviksi todetut kansalliset tunnistuspalvelut. [1, s. 48.]

Yleisesti voidaan kuitenkin todeta, että vahvassa autentikoinnissa käyttäjältä vaaditaan jokin toinenkin tieto käyttäjätunnuksen ja salasanan lisäksi. RSA määrittelee vahvan autentikoinnin seuraavin perustein. Vahvan autentikoinnin tulee sisältää kaksi kolmesta vaatimuksesta. Nämä vaatimukset ovat jokin tieto kuten salasana, jotain omistettua kuten pankkikortti ja jotain käyttäjälle yksilöllistä kuten sormenjälki. [2; 3; 4.]

2.1 Autentikointimenetelmiä

Käyttäjä voidaan tunnistaa käyttäen perinteistä käyttäjätunnus ja salasana -menetelmää tai käyttäen vahvempia menetelmiä, esimerkiksi kertakäyttösalasanoja. Se millainen autentikointi sovellukseen ja palveluun halutaan, riippuu palveluntarjoajan asettamista vaatimuksista. Tietyt palvelut, esimerkiksi verkkopankit, vaativat tehokkaampaa käyttäjäntunnistusta kuin esimerkiksi valokuvaharrastajien keskustelupalsta.

2.1.1 Käyttäjähakemistot

Perinteisen käyttäjätunnus ja salasana -tunnistautumisen perustana on tietokanta tai tiedosto, johon käyttäjien tunnukset ja salasanat tallennetaan. Heikoimmillaan kyseessä on tekstitiedosto, johon tiedot on tallennettu selväkielisinä. Yleisin vaihtoehto lienee käyttäjähakemisto, johon käyttäjät luodaan ja ryhmitellään ja jossa salasanat eivät ole selväkielisiä. Tällaisiä ovat esimerkiksi Microsoftin Active Directory ja LDAP -ratkaisut sekä radius-palvelimet. Valtiovarainministeriö antaa omat suosituksensa hyvästä salasanapolitiikasta. Nämä suositukset sisältävät muun muassa seuraavia kohtia:

- Salasanan pituus pitää olla vähintään 10 merkkiä.
- Salasanassa pitää olla merkkejä vähintään kolmesta luokasta (pienet kirjaimet, isot kirjaimet, numerot, erikoismerkit).
- Salasanan enimmäisikä on 90 päivää ja vähimmäisikä 1 päivä.
- Salasana ei saa olla sama kuin viisi aikaisempaa salasanaa.
- Salasana lukitaan usean epäonnistuneen yrityksen jälkeen ja vapautetaan ylläpidon toimesta. [5.]

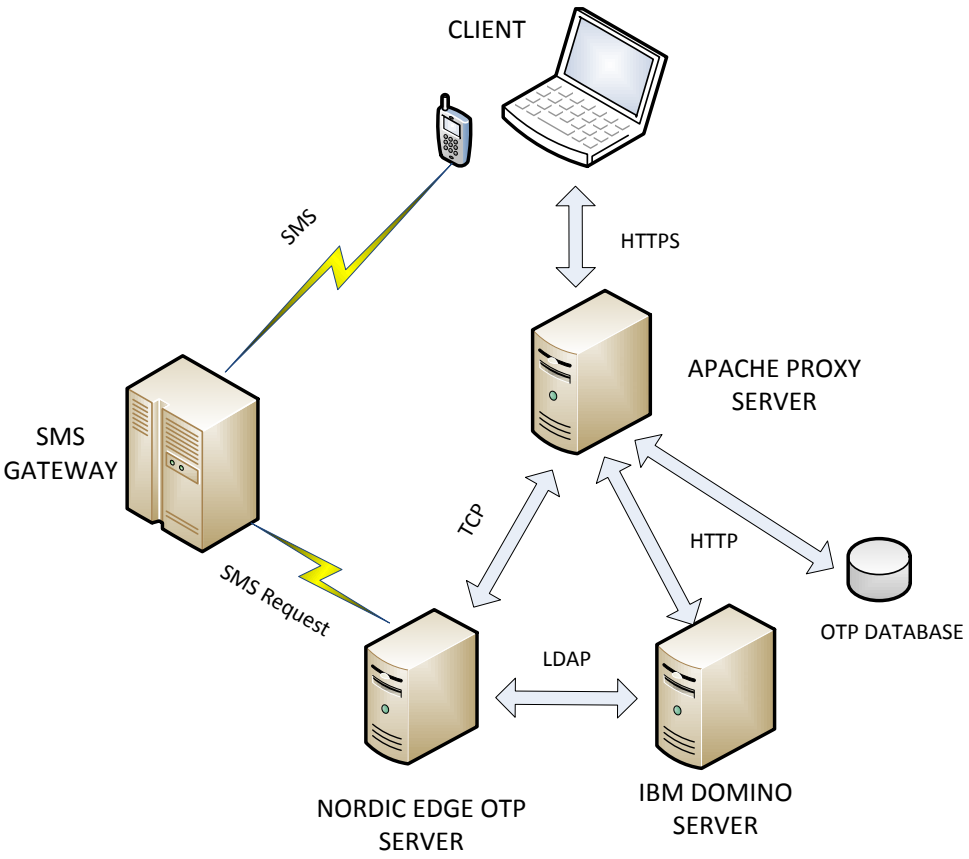
2.1.2 Kaksitasoinen autentikointi

Kaksitasoisessa autentikoinnissa käyttäjälle annetaan tunnuksen ja salasanan lisäksi jokin ylimääräinen tieto, jota käytetään henkilöllisyyden varmistamiseen. Yleisin vaihtoehto on kertakäyttösalasana, One Time Password (OTP). Kertakäyttösalasanan toteu-

tuksia on monia erilaisia, esimerkiksi tunnistautumishetkellä luotuja yksittäisiä tokeneita taikka ennaltamäärättyjä kertakäyttösalasanalistoja.

RSA-avaimet ovat esimerkki token-pohjaisesta vahvasta autentikoinnista. Siinä käyttäjällä on erillinen laite, token-avain, josta hän saa lyhyen aikaa voimassaolevan kertakäyttösalasanan. Toinen esimerkki vastaavasta on Google Authenticator, jossa käyttäjen älypuhelimessa on sovellus, joka laskee hänelle kertakäyttösalasanan. Kertakäyttösalasana voidaan myös toimittaa käyttäjille esimerkiksi SMS-viestillä taikka sähköpostilla. Kaksitasoisessa autentikoinnissa on kuitenkin omat haasteensa. SMS-viestillä lähetettäviä kertakäyttösalasanoja ei aina voida toimittaa perille esimerkiksi ulkomaalaisille käyttäjille. Lisäksi viestien lähettäminen maksaa ja ulkoisten palveluiden käyttäminen tuo omat lisenssikustannuksensa. Fyysiset token-avaimet kuten RSA:n avainperät voivat kadota tai särkyä.

Perinteiset tavat toteuttaa SMS-viestillä lähetettävä kertakäyttösalasana vaativat useita erillisiä komponentteja toimiakseen. Käyttäjähakemistoon tarvitaan usein erillinen komponentti, jonka avulla luodaan kertakäyttöinen salasana. Lisäksi voidaan tarvita erilaisia tietokantoja käyttäjistä, sekä web proxy -palvelimia ja yhdyskäytäviä, joilla luodaan kertakäyttösalasana, joiden kautta autentikointiliikenne ohjataan ja jotka lähettävät kertakäyttösalasanan käyttäjille ja lopulta tarkastavat takaisinsyötetyn salasanan oikeellisuuden.



Kuva 1 Nordic Edge SMS -ympäristö kertakäyttösalasanan käsittelyyn

2.1.3 Varmenteet

Varmenteiden avulla voidaan tunnistaa käyttäjä ja palvelu. Varmenteet ovat kolmannen osapuolen digitaalisesti allekirjoittamia osoituksia siitä, että kyseinen julkinen avain kuuluu tietylle avaimen omistajalle. Niitä myöntävät varmentajat (CA). Tunnistautumiseen käytettävät varmenteet voidaan jakaa kahteen pääryhmään, käyttäjävarmenteisiin ja palvelinvarmenteisiin. Palvelinvarmenteet asennetaan palveluntarjoajan palvelimiin ja niiden avulla käyttäjä varmistaa olevansa oikeassa palvelussa huijauspalvelun sijaan. Käyttäjävarmenne puolestaan kertoo palvelulle että käyttäjän laite, ja mahdollisesti käyttäjä on varmennettu. Myös itse laite voidaan varmentaa laitevarmenteella. [6.]

Varmenteet eivät kuitenkaan yksinään ole turvallisin mahdollinen tunnistautumismenetelmä. Esimerkiksi tietokoneeseen asennettava käyttäjävarmenne kertoo palvelulle ainoastaan, että kyseessä on henkilön X tietokone. Pelkän varmenteen avulla palvelu ei voi olla varma, että juuri henkilö X käyttää kyseistä konetta sillä hetkellä.

Laatuvarmenne on varmenne, joka täyttää Suomen sähköisen allekirjoituslain vaatimukset. Ja sen on myöntänyt lain vaatimukset täyttänyt varmentaja. Laatuvarmenteita valvoo Viestintävirasto ja Suomessa on tällä hetkellä vain yksi virallinen laatuvarmenteiden varmentaja, Väestörekisterikeskus. [7; 8.]

Mobiilivarmenne on DNA:n, Elisan ja Soneran kehittämä varmenne matkapuhelimiin. Se on matkapuhelimen SIM-korttiin asennettu käyttäjävarmenne, joka sisältää PIN-koodilla suojatun yksityisen avaimen. Tämän avaimen perusteella palveluntarjoaja pystyy varmistamaan mobiilivarmennetta käyttävän henkilön henkilöllisyyden luotettavasti. Mobiilivarmennepalvelu vahva sähköinen tunnistautumispalvelu, sillä se täyttää laissa määritetyt sähköisen tunnistautumisen kriteerit. [9, s. 12.]

2.1.4 Single Sign-On

Kertakirjautumisella, Single Sign-On (SSO), tarkoitetaan mekanismia, jonka avulla käyttäjä tunnistetaan useampaan palveluun kerralla. Toisin sanoen käyttäjä kirjautuu sisään haluamaansa palveluun. Käyttäjän siirtyessä toiseen palveluun autentikointijärjestelmä osaa siirtää jo aikaisemmin syötetyt tunnukset uudelle palvelulle. Käyttäjälle tämä näkyy saumattomana siirtymisenä palvelusta toiseen. Windows-ympäristöissä tietyt palvelut osaavat katsoa käyttäjätunnukset jo käyttäjän koneesta itsestään. Yleisimmät SSO-mekanismit käyttävät NTLM- tai Kerberos-protokollia kirjautumistietojen siirtämiseen ja istuntojen yllpitämiseen. [10.]

Security Assertion Markup Language, SAML, on XML-pohjainen protokolla, jolla siirretään autorisaatio- ja autentikointitietoa koneelta toiselle. SAML:n avulla voidaan toteuttaa joustavia SSO-ratkaisuja myös eri palveluntarjoajien välillä. SAML-protokolla määrittelee niin sanotun Service Providerin (SP) ja Identity Providerin (IdP). Toisin sanottuna SAML-laite pystyy itse autentikoimaan käyttäjän tai siirtämään jo autentikoidun käyttäjän tiedon eteenpäin. XML-pohjaisuuden ansiosta SAML-laite on monipuolinen ja helposti muokattavissa sekä helppo integroida erilaisiin rajapintoihin. [11.]

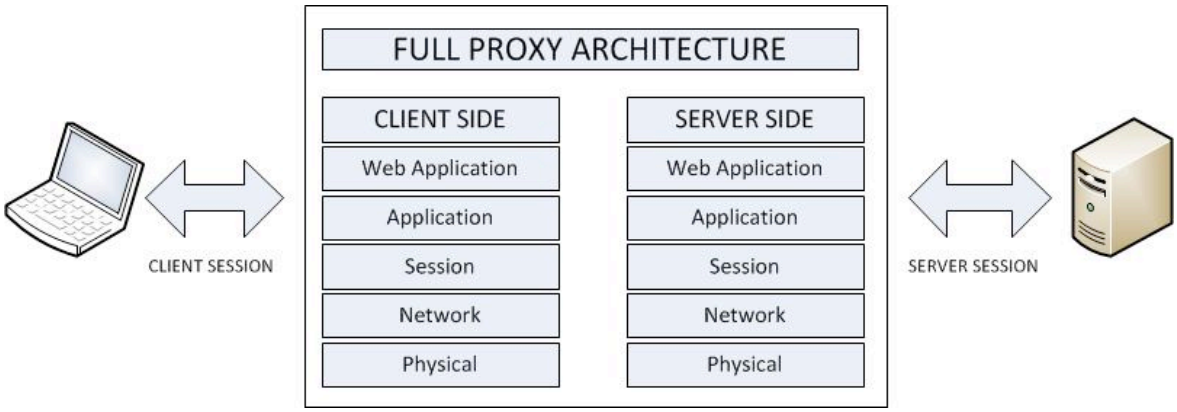
3 BIG-IP ja Access Policy Manager

F5 Networksin BIG-IP on niin sanottu ADC-järjestelmä (Application Delivery Controller). ADC-laitteisto sijaitsee tyypillisesti palveluiden edessä datakeskuksissa. Niillä varmiste-

taan korkea käytettävyys, kuormajako ja skaalautuvuus, parannetaan suorituskykyä, sovellusten turvallisuutta ja voidaan hoitaa pääsynvalvontaa. BIG-IP perustuu TMOS Full Proxy -arkkitehtuuriin (Traffic Management Operation System). Laitteet toimivat Full Proxyna sovelluspalvelimien edessä. BIG-IP-laitteisiin pystytään liittämään lisäominaisuuksia erikseen lisenssoitavilla moduuleilla. Moduulien saatavuus riippuu laitteen mallista. Tässä insinöörityössä käytetään pääasiallisesti virtuaalisia BIG-IP-koneita.

ADC-järjestelmät sijoitetaan perinteisesti niin kutsuttuun demilitarisoituun alueeseen (DMZ) datakeskuksissa. Toisin sanoen ADC-järjestelmät sijaitsevat palveluiden edessä, mutta palomuurien takana. Ensimmäisen sukupolven järjestelmät toimivat pääasiassa pelkästään kuormanjakajina ja sovelluskiihdyttiminä, mutta nykyään ADC-järjestelmät ovat kehittyneet ja pystyvät suorittamaan useita erilaisia tehtäviä. F5 Networksin BIG-IP-tuotteiden arkkitehtuuri mahdollistaa perinteisten kuormanjakotoimintojen lisäksi muun muassa käyttäjiltä sisääntulevan liikenteen tarkastelun ja käsittelyn, ennen kuin liikenne ohjataan halutulle palvelimelle ja palvelulle. BIG-IP-tuotteet huolehtivat myös tietoturvasta, eikä niitä ole välttämätöntä laittaa palomuurin taakse, kuten perinteisiä ADC-järjestelmiä. [12.]

BIG-IP-tuotteet rakentuvat TMOS Full Proxy -arkkitehtuurin ympärille. Kyseinen arkkitehtuuri perustuu eri protokollien syvälliseen ymmärtämiseen. Full Proxy -laite keskustelee erikseen käyttäjän ja palvelimen kanssa, eikä ole riippuvainen toisesta. Full Proxy -laite terminoi käyttäjän yhteyden ja avaa täysin uuden yhteyden palvelimelle. Tämä mahdollistaa liikenteen käsittelyn kumpaankin suuntaan erikseen ja jokaisen yhteyden optimoinnin aina sovellustasolle asti. [13, s. 4; 14.]

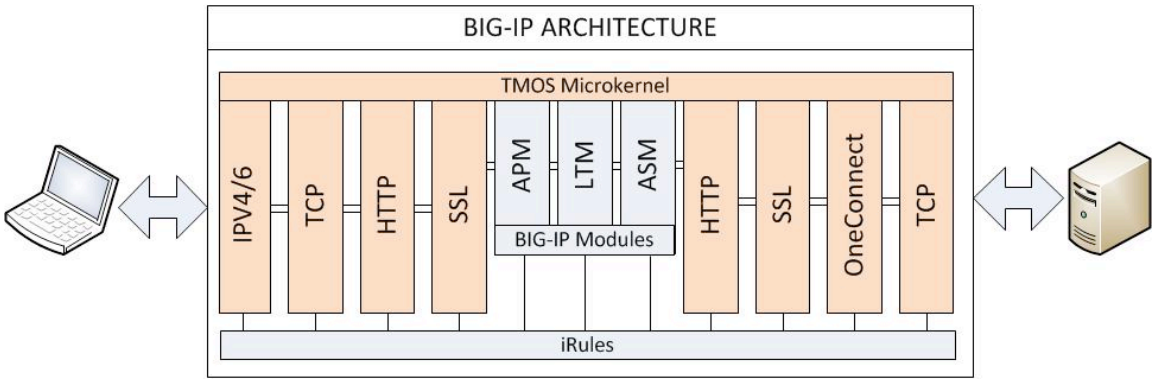


Kuva 2 Full Proxy

BIG-IP-laitteen toiminta ja moduulit voidaan jakaa muutamaan pääryhmään. Ensimmäisenä tulee palveluiden skaalautuvuus ja korkea käytettävyys. Perinteiset kuormanjakotoiminnot ovat oleellinen osa BIG-IP-laitteita ja Local Traffic Manager -moduuli (LTM) hoitaa näitä toimintoja. BIG-IP:n takana olevien palveluiden monistaminen ja kuormanjako niille mahdollistavat skaalautuvuuden ja korkean käytettävyyden. Itse BIG-IP-laitteiden korkea käytettävyys toteutetaan kahdentamalla järjestelmä. Käytännössä tämä tarkoittaa kahden BIG-IP-laitteen muodostamaa paria, joiden konfiguraatiotiedostot on synkronoitu keskenään. Yleisin käytäntö on asettaa toinen laitteista aktiivitilaan ja toinen stand-by-tilaan. Aktiivisen laitteen vioittuessa valmiustilassa toimiva laite siirtyy aktiiviseksi ja ottaa vioittuneen laitteen aseman. Tämä vaihdos ei näy loppukäyttäjälle.

Toisena pääryhmänä voidaan pitää resurssien ja tietoliikenteen optimointia. Web Accelerator -moduulilla voidaan kiihdyttää web-pohjaista liikennettä. Moduuli pyrkii muunmuassa hyödyntämään selainten välimuistia ja lataamaan käyttäjälle vain muuttuneet objektit. WAN Optimization Module (WOM) puolestaan pyrkii kiihdyttämään kahden päätepisteen välistä liikennettä vähentämällä siirrettävän datan määrää esimerkiksi kompressoinnilla ja deduplikoinnilla. Resurssien optimointia tapahtuu myös SSL-purun ja TCP-optimoinnin muodossa.

Tietoturvasta vastaavat kolme moduulia, Advanced Firewall Module (AFM), Application Security Module (ASM) ja Access Policy Manager (APM). BIG-IP-laitteet ovat ICSA-sertifioituja palomuureja ja AFM on palomuuritoimintojen hallintakäyttöliittymä. ASM puolestaan vastaa sovelluspuolen tietoturvasta. APM on täysiverinen SSL VPN -moduuli, joka hoitaa SSL VPN-ominaisuuksien lisäksi päätelaite- ja käyttäjätunnistusta, autentikointia ja pääsynhallintaa. Access Policy Manager ja sen toiminnat ovat oleellinen osa tätä insinöörityötä.

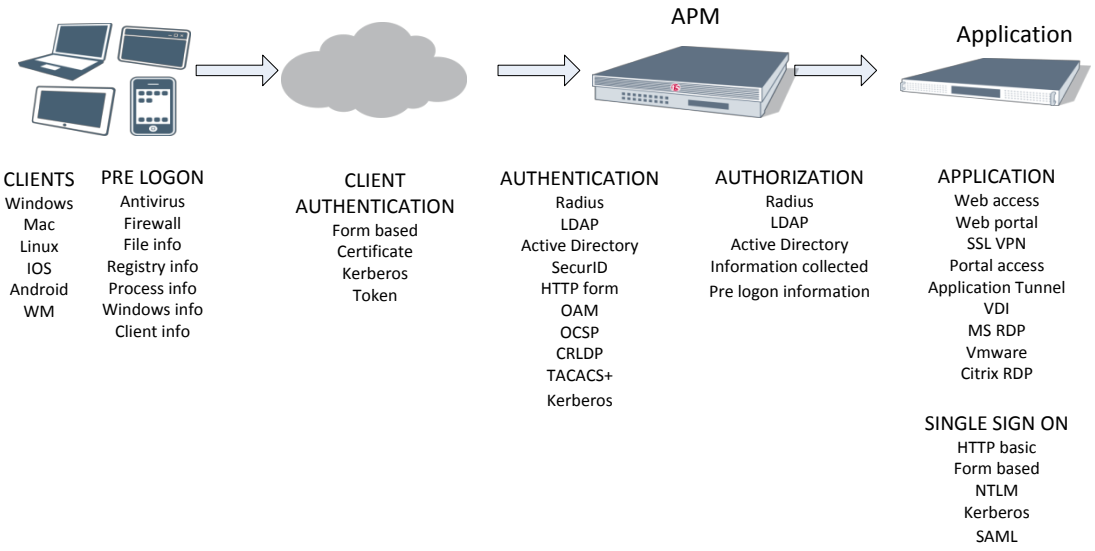


Kuva 3 BIG-IP -arkkitehtuuri

BIG-IP-järjestelmän ominaisuuksia ja toiminnallisuutta voidaan laajentaa tapahtumapohjaisella iRule-skriptauskielellä, jonka avulla voidaan hallita ja muokata IP-pohjaista liikennettä ja sen sisältämää dataa. iRuleilla tyypillisesti myös tehdään älykästä liikenteenohjausta, luodaan uusia toimintoja ja laajennetaan ADC-laitteiston ominaisuuksia sekä paikataan sovelluksissa havaittuja puutteita ja ongelmia. Tämä skriptauskieli on TMOS-mikrokernelin ominaisuus ja se on yhteensopiva kaikkien BIG-IP-laitteiden ja moduulien kanssa. [15.]

Access Policy Manager

Access Policy Manager (APM) on BIG-IP:n erikseen lisenssoitava pääsynhallintamoduuli, jolla voidaan toteuttaa joustavia ja turvallisia etäyhteysratkaisuja ja käyttäjän sekä päätelaitteen tunnistuksia. APM:n avulla voidaan myös käyttäjät ohjata käyttäjätunnistuksen perusteella halutuille palvelimille. Sen avulla voidaan myös julkaista käyttäjäntunnistuksen takana olevia kuormanjaettuja HTTP-sovelluksia joko suoraan tai käyttäen web-portaalia. APM:llä voidaan myös avata pääsy haluttuihin verkkoihin.



Kuva 4 APM-toimintoja

Oheinen kuva selvittää hyvin Access Policy Managerin toimintoja. APM:n avulla voidaan tehdä erilaisia käyttäjä- ja päätelaitetarkistuksia, esimerkiksi tarkistaa käytetyn laitteen käyttöjärjestelmä. Sen avulla voidaan suorittaa erilaisia tarkistuksia myös ennen sisäänkirjautumista, voidaan tarkistaa laitteesta löytyviä virustorjuntaohjelmia ja palomuureja ja edellyttää riittävän tuoretta virustietokantaa.

Client-puolen autentikoinnissa tuetaan lomakepohjaista autentikointia, Kerberosta, erilaisia token-pohjaisia ratkaisuja sekä client-sertifikaatteja. Lisäksi ohjelmistoversiosta 11.3 lähtien on tuki client-puolen NTLM-autentikoinnille ja SAML-autentikoinnille. Perinteisiä palveluiden toteuttamia autentikointimenetelmiä tuetaan laajalti. Käyttäjähakemistoon pohjautuvat ratkaisut kuten radius, LDAP ja Active Directory ovat tuettuja. Myös erilaiset HTTP Form -pohjaiset autentikoinnit on tuettu. Access Policy Managerin avulla pystytään myös tuottamaan kertakäyttösalasanoja ja varmistamaan, että ne ovat paikkansapitäviä.

Käyttäjäntunnistuksen ja pääsynhallinnan lisäksi APM toimii myös SSL VPN -laitteena. Sen avulla voidaan toteuttaa erilaisia etäyhteysratkaisuja. Käyttäjälle voidaan avata SSL VPN -tunneli haluttuun verkkoon tai käyttäjälle voidaan tarjota tietty ohjelma SSL VPN -tunnelissa. Käyttäjän voidaan myös antaa valita haluamansa resurssit luomalla portaali, jonne halutut resurssit asetetaan tai käyttäjällä voi olla vain oikeudet ajaa esimerkiksi Remote Desktopia, jolloin se voidaan avata suoraan hänelle.

Lisäksi APM tukee laajaa valikoimaa erilaisia VDI-ratkaisuja. Tuki löytyy muun muassa Citrix-, MS RDP- ja VMware -ympäristöihin. Näihin voidaan myös suorittaa erilaisia Single Sign-On -järjestelmiä. Käyttäjän aikaisemmin käyttämiä tunnuksia voidaan viedä eteenpäin muille palveluille esimerkiksi Kerberoksen tai SAML:n avulla.

APM on helppokäyttöinen ja selkeä. Riippuen mitä sen avulla halutaan toteuttaa, tarvitsee käyttäjän määritellä AAA-serverit, mahdolliset VPN-tunnelit, SSO-ominaisuudet ja muut jaettavat resurssit. Näiden pohjalta luodaan pääsynhallintasäännöt. Tässä käytetään apuna Visual Policy Editoria, jolla tehdään pääsynhallinnasta vuokaavio. Tästä on helppo tarkastella pääsynhallintasääntöjen kulkua, ja vuokaaviota voidaan muokata helposti. Jo luotuun policyyn voidaan lisätä osia ja tapahtumia. Turhiksi todettuja kohtia voidaan poistaa ilman, että policya tarvitsee luoda uudestaan.

Access Policy Manager tukee iRule-skriptiohjelmointia, joita voidaan käynnistää suoraan halutuissa kohdissa access policyn vuokaaviota. iRulejen avulla voidaan muun muassa toteuttaa kertakäyttösalasanojen luominen ja pääsynhallintaan tarvittava lokitus.

4 Käytännön toteutukset

Tässä insinöörityössä toteutetaan kolme erilaista autentikointimekanismia. Ensimmäisenä toteutetaan ohjelmisto-token-pohjainen Google Authenticator. Tämän jälkeen tulee sähköpostilla lähetettävä kertakäyttösalasana ja viimeisenä tehdään Proof of Concept - toteutus TUPAS-tunnistautumisesta. Kaikki autentikointijärjestelmät toteutetaan hyödyntämällä F5 Networksin BIG-IP-laitteita ja Access Policy Manager -moduulia.

Testiympäristönä käytetään DCC Oy:n ESXi-palvelinta. Palvelimelle asennetaan virtuaaliversio BIG-IP:sta ohjelmistoversiolla 11.2.1. Lisäksi palvelimella on Windows Server 2008 R2 -palvelin, jossa toimii DHCP-, DNS-, NTP- ja Active Directory -palvelut. Tarpeen vaatiessa voidaan luoda lisää virtuaalikoneita. Lisäapuna käytettiin BIG-IP-laitteiden käyttäjäyhteisön verkkofoorumia, DevCentralia (devcentral.f5.com).

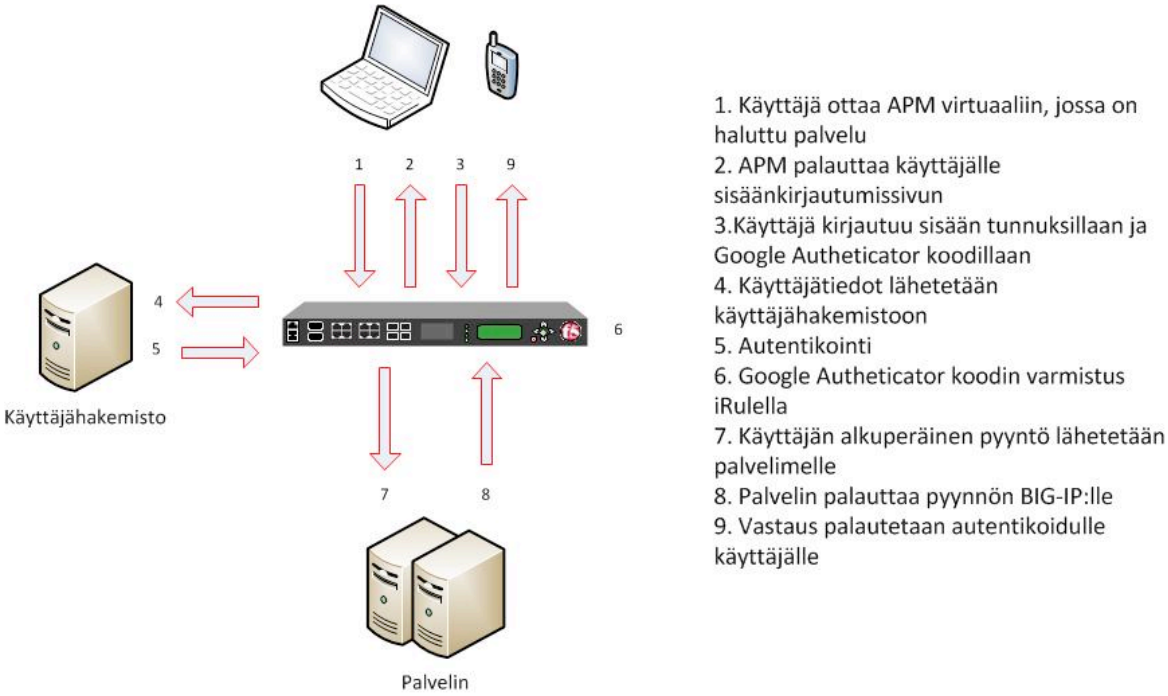
4.1 Google Authenticator

Google Authenticator on Googlen kehittämä ilmainen kertakäyttösalasanasovellus älypuhelimiin. Sen avulla voidaan luoda aikaan perustuvia kertakäyttöisiä salasanoja (TOTP). Tämä salasana lasketaan luomalla token HMAC-SHA1 -algoritmilla. Tokenin laskennassa käytetään käyttäjän ja autentikointijärjestelmän välillä jaettua salaista avainta sekä 30 sekunnin välein tarkistettavaa UNIX-aikaa (Unix time epoch). Käyttäjän älypuhelimen ja BIG-IP-järjestelmän kellojen tulee olla riittävän lähellä toisiaan tämän aikavälin sisällä. [16.]

Google Authenticator ratkaisua kokeiltiin DCC Oy:n testiympäristössä. Se pystytettiin virtuaaliseen BIG-IP-koneeseen, jossa oli Access Policy Manager -moduuli. Lisäksi toteutuksessa käytettiin Windows Server 2008 -palvelimen pyörittämää Active Directoryä. APM hoitaa autentikoinnin Active Directorya vastaan, sekä määrittää milloin Google Authenticatorin tarvitsema aikariippuva kertakäyttösalasana (TOTP) lasketaan. Salasanan generointi hoidettiin iRule-ohjelmoinnilla. Demon pystyttämisessä hyödynnettiin F5 Networksin käyttäjäyhteisöstä löytyvää ohjetta. [17.]

4.1.1 Toiminta

Käyttäjä ottaa yhteyden virtuaaliserveriin, joka on liitetty Google Authenticator -access policyyn. Käyttälle annetaan sisäänkirjautumissivu, johon hän syöttää käyttäjätunnuksensa ja salasanansa ja myös kertakäyttöisen koodinsa puhelimestaan. APM tarkistaa ja autentikoi käyttäjän Active Directory -palvelua vasten. Samalla BIG-IP laskee Google Authenticatorin käyttämän kertakäyttösalasanan ja tarkistaa, että se vastaa käyttäjän syöttämää koodia. Autentikoinnin onnistuessa käyttäjän alkuperäinen pyyntö ohjataan halutulle palvelulle ja tämä palvelu lähettää vastauksensa BIG-IP:lle, joka puolestaan viestittää vastauksen käyttäjälle.

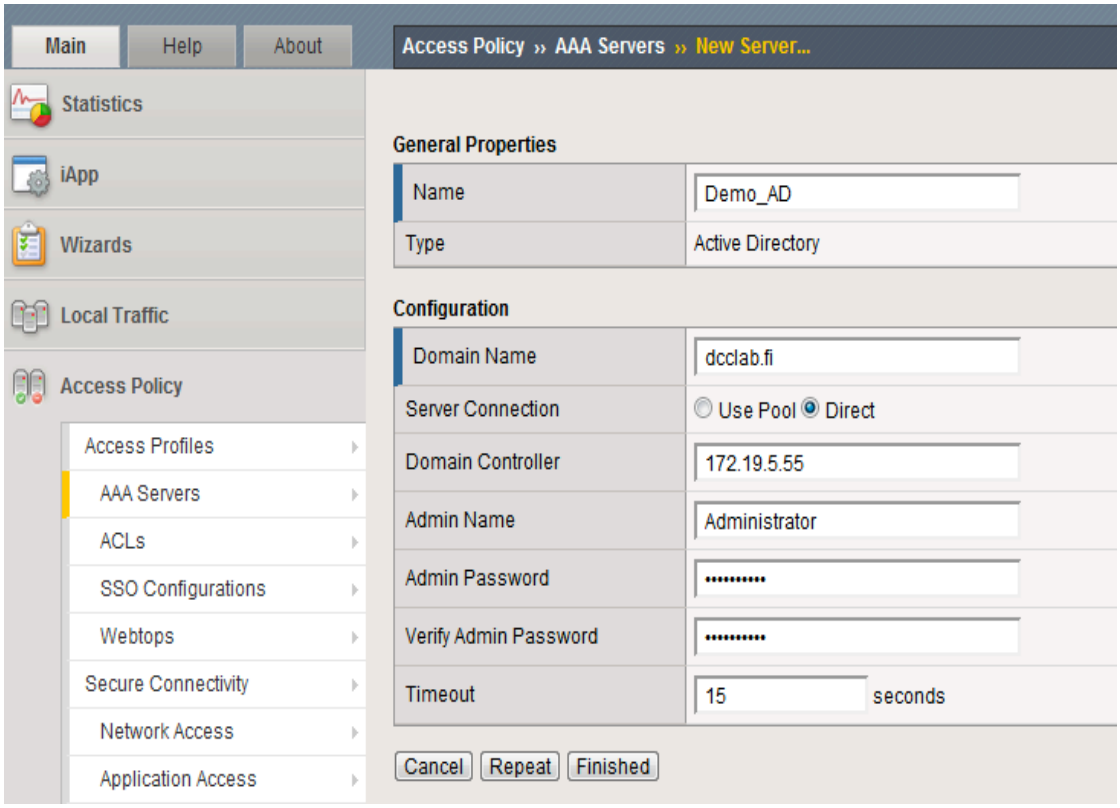


Kuva 5 Google Authenticatorin toiminta

Google Authenticatorin pystyttäminen voidaan jakaa neljään osaan. Nämä osat ovat Active Directory -palvelun määrittäminen Access Policy Managerissa, Google Authenticator kertakäyttösalasananlaskenta-iRule ja tunnustenluonti-iRule, vuokaavion liittäminen ja muokkaaminen sekä virtuaaliservereiden luominen.

4.1.2 Active Directory ja Access Policy Manager

Ensimmäisenä määritellään käyttäjähakemisto, jota vastaan APM autentikoi käyttäjän. DCC Oy:llä käyttäjähakemistona on testiympäristössä Windows Server 2008 -pohjainen Active Directory. Se määriteltiin BIG-IP APM:ään Access Policy -välilehden AAA Servers -kohdan kautta. Active Directory palvelimelle annetaan nimi, jonka avulla se tunnistetaan Access Policyissä. Tämän jälkeen määritellään domain, domain controller sekä admin-tunnukset.



Kuva 6 Active Directoryn määritys APM:ssä

Koska DCC Oy:n testiympäristö rakentuu pitkälti Active Directoryn varaan, käytetään Domain Controllerin määrittämiseen IP-osoitetta nimen Fully Qualified Domain Namen sijaan. Näin varmistetaan jatkuva yhteys palveluun, vaikka DNS-asetuksia muutettaisiinkin.

4.1.3 Kertakäyttösalana- ja tunnuksen luonti - iRule

Active Directoryn määrittämisen jälkeen luodaan iRule, joka laskee käyttäjälle ajasta riippuvan kertakäyttösalasanan. Kyseinen iRule löytyy F5 Networksin käyttäjäyhteisöstä, devcentralista. BIG-IP-laitteeseen luodaan uusi *google_auth_verify_apm*-iRule ja DevCentralista löytyvä koodipätkä kopioitiin siihen. Koodiesimerkkiin 1 on poimittu oleellisin kohta iRulesta. Kokonaisuudessaan iRule löytyy liitteistä (liite 1).

```
when ACCESS_POLICY_AGENT_EVENT {
    if { [ACCESS::policy agent_id] eq "ga_code_verify" } {
        ### Google Authenticator verification settings ###

        # lock the user out after x attempts for a period of x se-
conds
        set static::lockout_attempts 3
        set static::lockout_period 30

        # logon page session variable name for code attempt form
field
        set static::ga_code_form_field "ga_code_attempt"

        # key (shared secret) storage method: ldap, ad, or datagroup
        set static::ga_key_storage "datagroup"

        # LDAP attribute for key if storing in LDAP (optional)
        set static::ga_key_ldap_attr "google_auth_key"

        # Active Directory attribute for key if storing in AD (op-
tional)
        set static::ga_key_ad_attr "google_auth_key"

        # datagroup name if storing key in a datagroup (optional)
        set static::ga_key_dg "google_auth_keys"
```

Esimerkkikoodi 1 Google Authenticatorin muuttujat

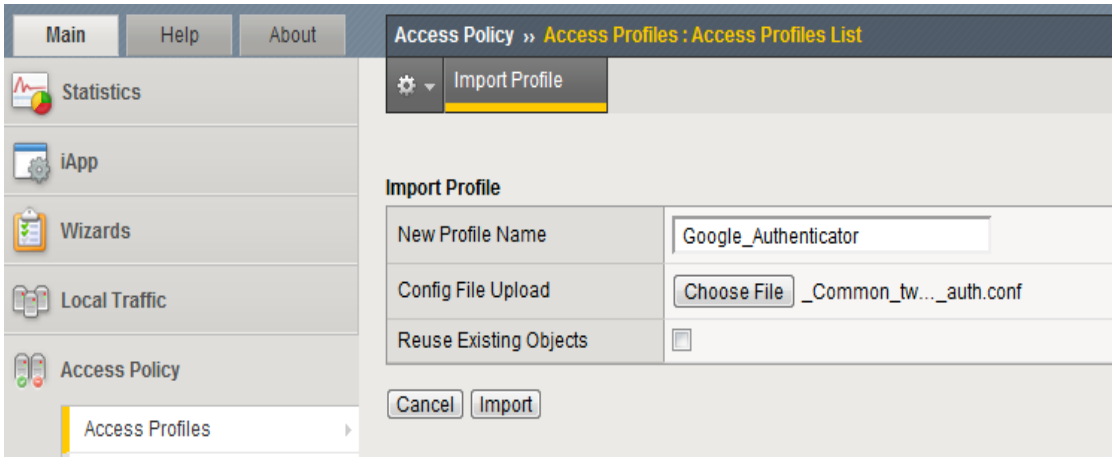
Google Authenticatorin pää-iRulessa on seitsemän muuttujaa, joita voidaan muokata halutunlaisiksi. *Lockout_attempts* ja *lockout_period* määrittelevät, kuinka monta kertaa käyttäjä voi koettaa kirjautua virheellisesti sisään ennen kuin hänen tunnuksensa lukitaan tietyksi aikaa. *Ga_code_form_field* määrittää muuttujan nimen ja arvon, johon TOTP syötetään sisäänkirjautumissivulla. Näiden muuttujien oletusarvot sopivat testiympäristöömme, joten niihin ei koskettu. Toiminnan kannalta tärkein muuttuja on *ga_key_storage*. Se määrittää, minne käyttäjän jaettu avain on tallennettu. Vaihtoehtoja ovat LDAP, Active Directory tai datagroup. Datagroup on BIG-IP-laitteen sisäinen taulukko, jonne voidaan tallentaa arvopareja. Testiympäristöön valittiin datagroup. *Ga_key_ldap_attr*, *ga_key_ad_attr* ja *ga_key_dg* määrittävät, minkä niminen attribuutti taikka datasäilö sisältää käyttäjän ja hänen jaetun avaimensa. Loppu iRulesta hoitaa TOTP-laskemisen.

Samalla luodaan toinen iRule, *Google_Auth_Soft_Token_Gen*. Tämä iRule ei ole osa varsinaista autentikointijärjestelmää, vaan pikemminkin aputyökalu, jonka avulla luodaan käyttäjille mahdollisuus käyttää Google Authenticator -sovellusta. iRulen avulla käyttäjälle tarjotaan verkkosivu, jossa käyttäjille luodaan tunnus Google Authenticatoria

varten. Käyttäjä luodaan muotoon *username@domain*. Kun tämä on tehty, ilmoittaa sivu käyttäjän jaetun avaimen, joka tullaan myöhemmin liittämään niin älypuhelimeen kuin BIG-IP-laitteeseen. Tuotantoympäristössä tulee tarkkaan miettiä, kenellä on oikeus käyttää kyseistä iRulea. iRule löytyy kokonaisuudessaan liitteistä.

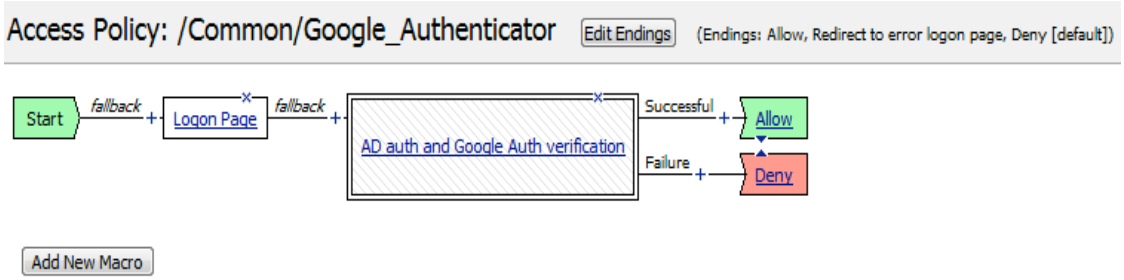
4.1.4 Vuokaavio

F5 devcentralista löytyy valmis pohja Google Authenticator – access policyn vuokaavioolle. Kyseinen Access Policy liitetään BIG-IP-laitteeseen. Liittäminen tapahtuu BIG-IP:n Access Policy -välilehden Access Profiles -kohdassa. Samalla luotava pääsynhallintaprofiili nimettiin nimellä *Google_Authenticator*.



Kuva 7 Access Policyn liittäminen

Tätä testiympäristöä varten vuokaaviota hieman yksinkertaistettaan. Siitä poistettaan muun muassa Landing URI Check, koska DCC Oy:n testiympäristössä ei ole tarvetta suorittaa kyseistä tarkistusta.

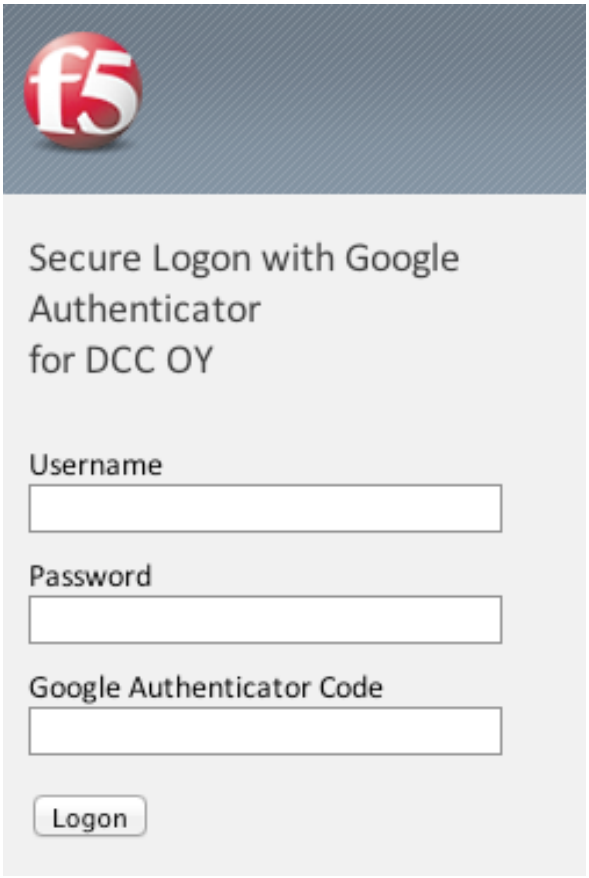


Kuva 8 Google Authenticator -vuokaavio

Vuokaavio koostuu Logon Page -sisäänkirjautumissivusta, johon käyttäjä syöttää käyttäjätunnuksensa ja salasanansa. Lisäksi hänen täytyy syöttää myös Google Authenticatorin kertakäyttösalasana. Sisäänkirjautumissivua voi muokata haluamakseen. Siihen lisätään kenttä TOTP-salasanalle. Kyseiseen kenttään syötetty arvo tallennetaan nimellä *ga_code_attempt*. Aikaisemmin luotu *google_auth_verify_apm*-iRule vertailee tätä arvoa itse laskemaansa arvoon.

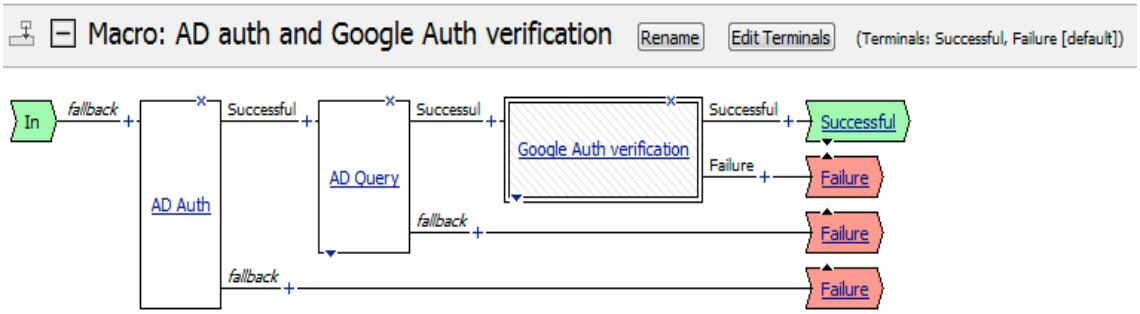
	Type	Post Variable Name	Session Variable Name	Read Only
1	text	username	username	No
2	password	password	password	No
3	text	ga_code_attempt	ga_code_attempt	No
4	none	field4	field4	No
5	none	field5	field5	No

Kuva 9 Sisäänkirjautumissivun asetukset



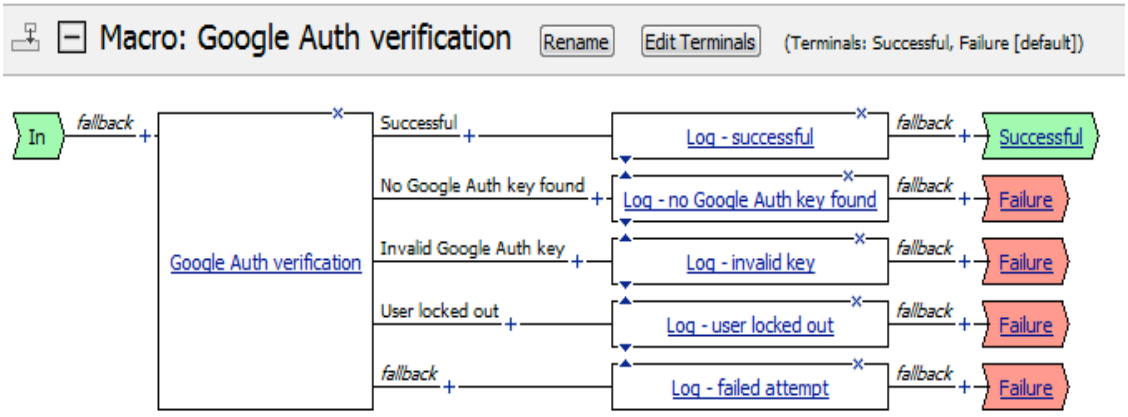
Kuva 10 Sisäänkirjautumissivu

Käyttäjän syötettyä tunnuksensa siirrytään AD auth and Google Auth verification -kohtaan, joka on itse asiassa makro, joka suorittaa käyttäjän autentikoinnin Active Directorya vastaan sekä kysyy AD:lta, mihin ryhmiin käyttäjä kuuluu. Autentikoinnin ja kyselyn onnistuessa siirrytään *Google Auth verification* -makroon. Makrolla voidaan koota osa vuokaaviota yhdeksi rakennuspalikaksi, jota voidaan käyttää useassa eri kohdassa vuokaaviota.



Kuva 11 Autentikointimakro

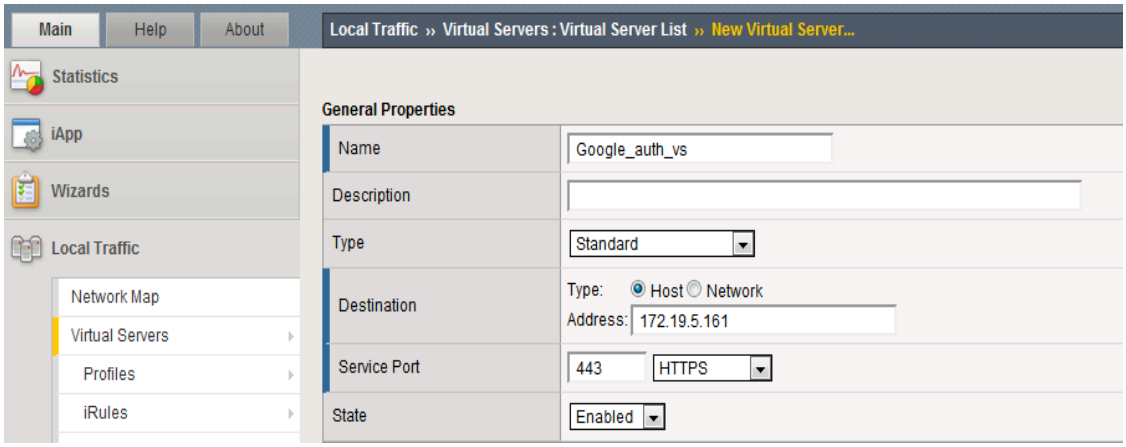
Google Auth verification -kohdassa käynnistetään *google_auth_verify_apm*-iRule ja tarkistetaan käyttäjän syöttämä TOTP. Tarkistuksen tuloksen perusteella APM lokittaa tietoja sisäänkirjautumisyrityksestä. Lokeista saadaan tieto, puuttuiko käyttäjältä Google Authenticatorin vaatima jaettu avain vai oliko syötetty TOTP väärä. Usean epäonnistuneen sisäänkirjautumisyrityksen jälkeen käyttäjän tunnukset lukitaan ennalta määrätyksi ajaksi.



Kuva 12 Kertakäyttäsalasanan tarkistus -makro

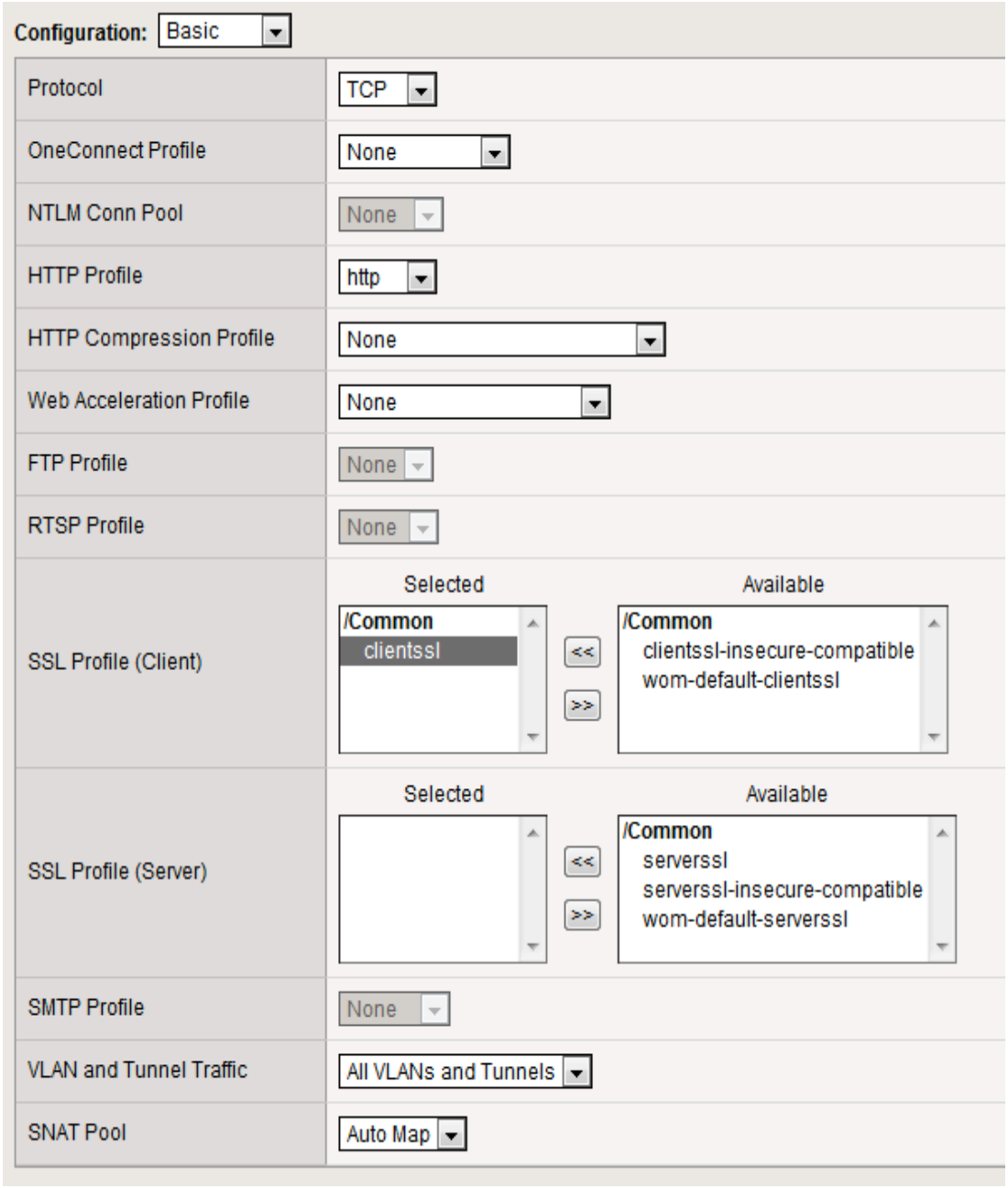
4.1.5 Virtuaaliserverit

Ennen kuin Access Policyn vuokaaviota voidaan kokeilla toiminnassa, tarvitaan virtuaaliserveri, johon Access Profile liitetään. Tämä virtuaaliserveri näkyy käyttäjälle kohdepalveluna. Virtuaaliserverin luominen tapahtuu Local Traffic -välilehdeltä. Perusominaisuuksissa määritellään virtuaaliserverin nimi, tyyppi, IP-osoite ja -portti. Portiksi APM vaatii HTTPS-portin (443).



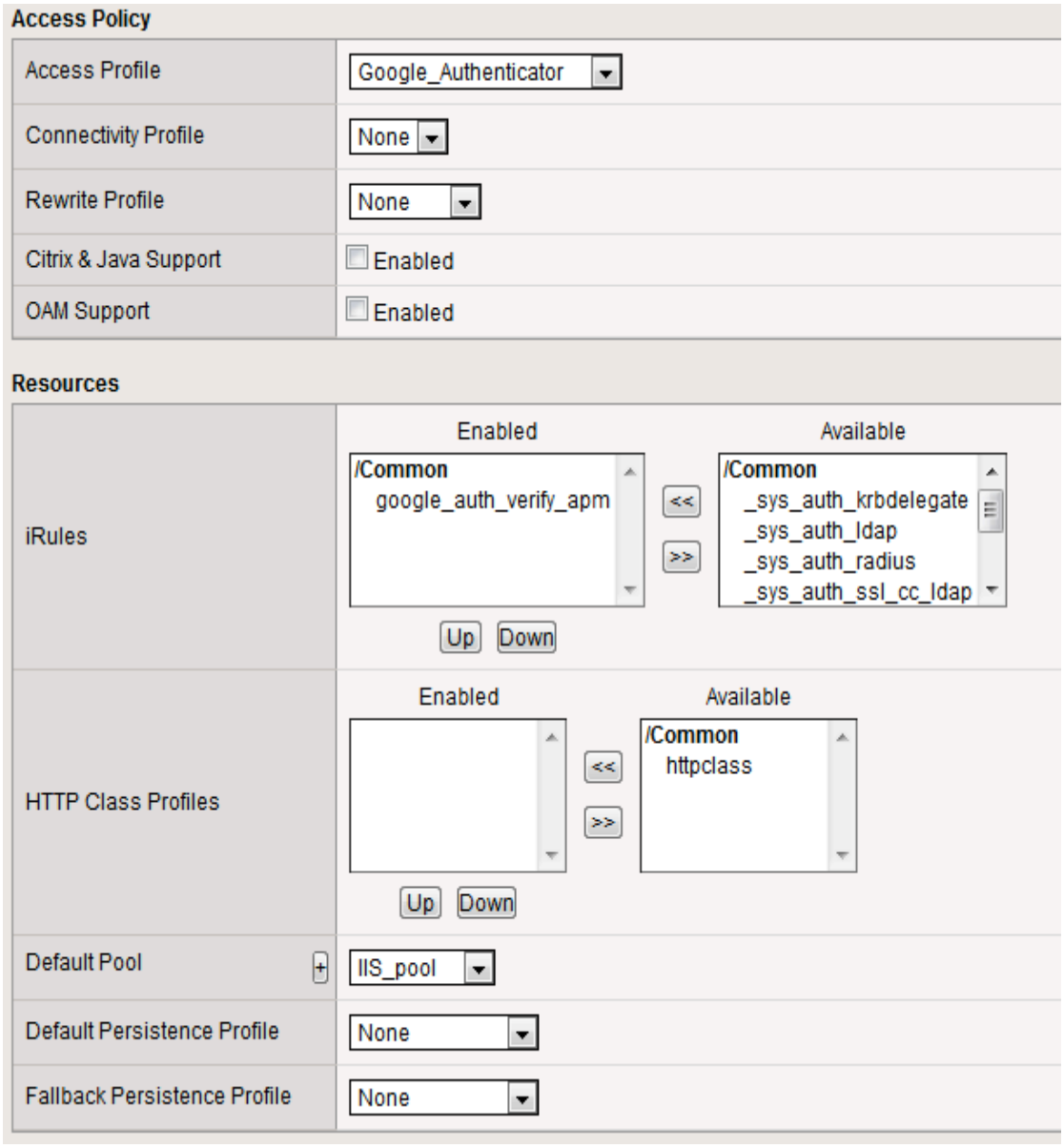
Kuva 13 Virtuaaliserverin yleiset asetukset

Virtuaaliserverin toimintaprotokollaksi määritellään TCP. Siihen valittiin http-profiili käyttöön. Koska APM vaati virtuaaliserveriltä HTTPS-salauksen, pitää siihen myös liittää geneerinen SSL-profiili. Viimeisenä kytketään päälle SNAT, eli Secure Network Address Translation, yksisuuntainen NAT. SNATin käyttäminen johtuu DCC Oy:n verkonrakenteesta, eikä ole kaikissa ympäristöissä pakollinen.



Kuva 14 Lisää virtuaaliserverin asetuksia

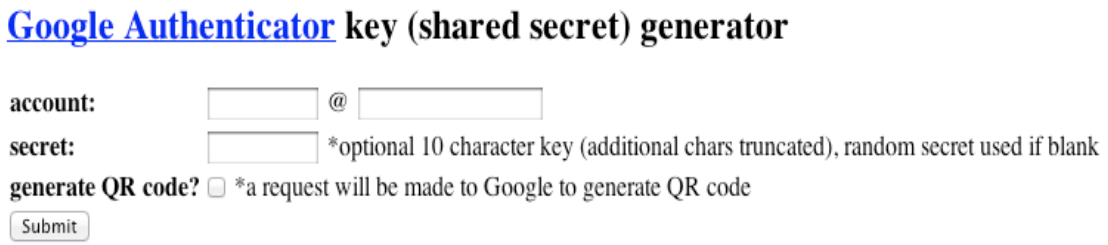
Viimeiseksi virtuaaliserveriin liitetään Google Authenticator -profiili, sekä määritellään virtuaaliserverille resursseja. Tässä tapauksessa resursseja ovat aikaisemmin luotu *google_auth_verify_apm*-iRule ja *IIS_pool*, jossa sijaitsee IIS-web-palvelu, joka on tämän esimerkin kohdepalvelu, johon käyttäjä autentikoidaan.



Kuva 15 Virtuaaliserverin resurssit

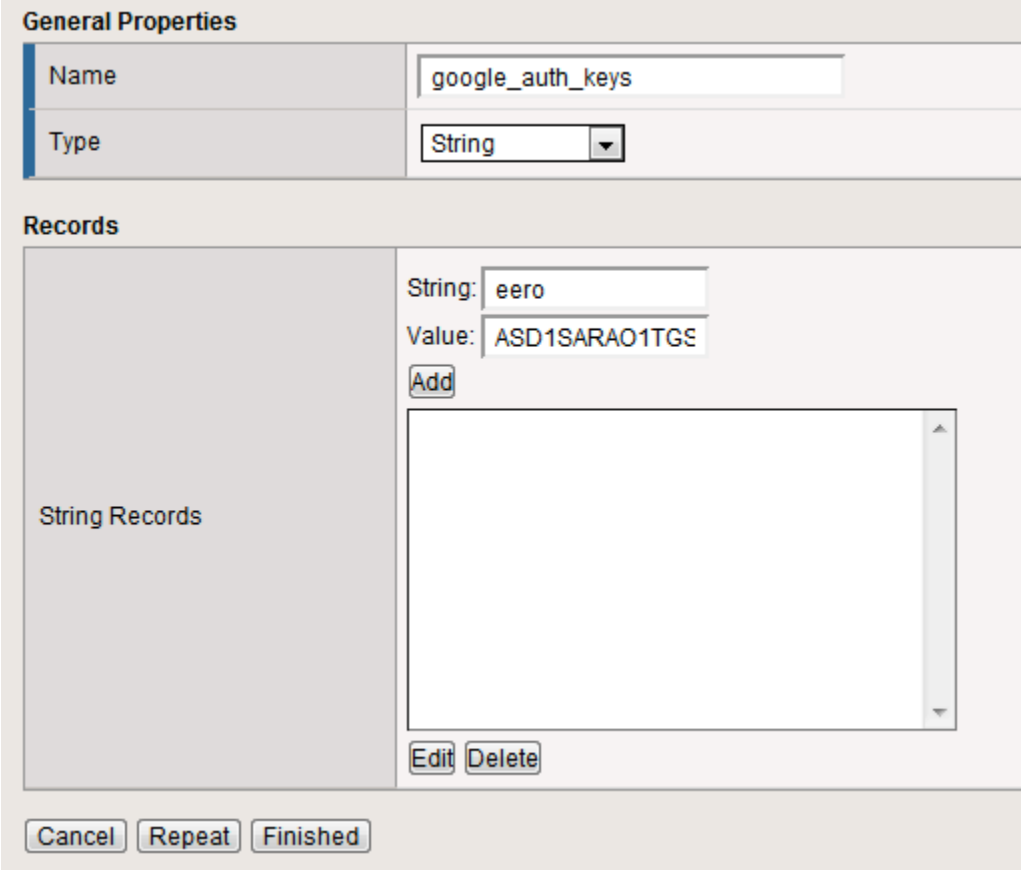
Aikaisemmin luotua *Google_Auth_Soft_Token_Gen*-iRulea varten tarvitaan toinen virtuaaliserveri, jonka avulla luodaan Google Authenticatorin vaatima jaettu avain. Tämän virtuaaliserverin luominen tapahtuu samalla tavalla kuin aikasemmankin, muutamalla poikkeuksella. Tähän ei tarvita HTTPS-tukea, joten portiksi jätettiin pelkkä *HTTP:80*. Tämän takia myös SSL-profiili jätetään pois. Resursseihin valitaan pelkästään *Google_Auth_Soft_Token_Gen*-iRule. Kohdepalvelua ei tarvita, koska iRule luo, laskee ja palauttaa käyttäjälle jaetun avaimen yksinkertaisena web-sivuna.

Tämä virtuaaliserveri ei varsinaisesti ole osa autentikointijärjestelmää, vaan sen avulla luodaan käyttäjille tunnukset Google Authenticator -sovellukseen. Sen ei tarvitse sijaita samassa verkossa, eikä edes samassa BIG-IP-laitteessa kun varsinaisen Google Authenticator -virtuaaliserverin. Jaettu avain voidaan toki myös keksiä ja sopia käyttäjän ja BIG-IP-laitteen ylläpidon kesken muutenkin.



Kuva 16 iRulella generoitu HTML-sivu jaetun avaimen luontiin

Ennen kuin testiympäristö on valmis käytettäväksi pitää vielä luoda BIG-IP:hen data group, jonne käyttäjien jaetut avaimet tallennetaan. Data groupin nimeksi määritellään aikaisemmasta iRulesta löytyvä *google_auth_keys* ja tyypiksi valittaan String. Käyttäjät lisätään laittamalla String-kohtaan käyttäjän Active Directory tunnus ilman domain tunnistetta. Value-kohtaan laitetaan *Google_Auth_Soft_Token_Gen*-iRulella virtuaaliserverissä luotu jaettu avain. Lisäksi puhelimeen ladataan Google Authenticator -sovellus ja siihen lisätään tunnukset muodossa *tunnus@domain* ja jaettu avain. Tämän jälkeen testiympäristö on valmis käytettäväksi.



Kuva 17 Datagroup

4.2 Email OTP

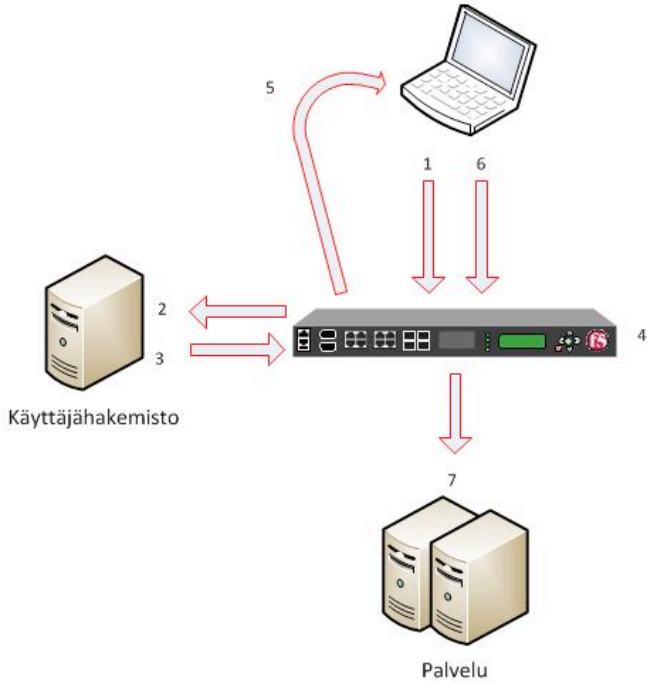
Vahvassa autentikoinnissa kertakäyttösalasanat ovat oleellisessa osassa. DCC Oy testaa sähköpostiin lähetettävää kertakäyttöistä salasanaa Active Directory autentikoinnin yhteydessä. Testiympäristö rakennetaan käyttämällä Windows Server 2008 -palvelinta Active Directorynä. Autentikoinnin, kertakäyttösalasanan luonnin ja lähetyksen hoitaa BIG-IP ja sen Access Policy Manager -moduuli.

4.2.1 Toiminta

Toiminta perustuu Active Directoryyn tallennettuun sähköpostiosoitteeseen. Käyttäjä syöttää APM:llä luotuun sisäänkirjautumissivuun Active Directory tunnuksensa, jonka jälkeen APM autentikoi käyttäjän, luo hänelle kertakäyttöisen salasanan, poimii käyttäjän sähköpostiosoitteen AD:stä ja lähettää kertakäyttösalasanan hänelle. Käyttäjä syöt-

tää saamansa kertakäyttösalasanan, APM tarkistaa sen olevan oikea ja näin käyttäjä on autentikoitu kahdesti ja hänet päästetään sisään palveluun.

1. Käyttäjä ottaa APM virtuaaliin, jossa on haluttu palvelu ja syöttää käyttäjätunnuksensa
2. APM suorittaa autentikoinnin käyttäjähakemiston kanssa ja tarkistaa, että käyttäjältä löytyy sähköpostiosoite
3.Käyttäjähakemisto palauttaa sähköpostiosoitteen APM:lle
4. APM käynnistää iRulen, joka luo kertakäyttösalasanan ja tallentaa sen muuttujaan ja lokittaa tiedon salasanan luonnista. SNMP trap käynnistää skriptin, jonka avulla salasana toimitetaan käyttäjälle
5. Kertakäyttösalasana lähetetään käyttäjälle
6. Käyttäjä syöttää kertakäyttösalasanan ja APM tarkistaa sen olevan sama kuin aikaisemmin muuttujaan tallennettu
7. Käyttäjä ohjataan hänen pyytämäänsä palveluun



Kuva 18 Email OTP:n toiminta

4.2.2 Kertakäyttösalasana-iRule ja sähköposti-skripti

Testiympäristössä hyödynnettiin jo aikaisemmin luotua Google Authenticator -ympäristöä pohjana. Molemmat demot hyödyntävät samaa virtuaaliserveriä, jossa sisäänkirjautumissivu sijaitsee. Käyttäjän ottaessa yhteyden virtuaaliserveriin selaimen kautta hänelle annetaan valikko, josta hän valitsee, haluaako hän autentikoida itsensä Google Authenticatorin kautta vai sähköpostitse.

Access Policylle käytetään jo aikaisemmin määriteltyä virtuaaliserveriä, samoin Active Directoryn tiedot on jo täytetty aikaisemman Google Authenticatorin luonnin yhteydessä. Ensimmäisenä suunnitellaan iRule, joka luo kertakäyttösalasanan ja tallentaa kirjautujan tiedot lokiin.

```
when ACCESS_POLICY_AGENT_EVENT {
   if { [ACCESS::policy agent_id] eq "otp_verify_email" } {
            binary scan [b64encode [CRYPTO::keygen -alg random -len
128]] i otp
            set otp [string range $otp 0 3]
            ACCESS::session data set session.user.otp $otp
            set    mobile    [ACCESS::session    data    get    "ses-
sion.ad.last.attr.mobile"]
            set    mail    [ACCESS::session    data    get    "ses-
sion.ad.last.attr.mail"]
            set  logstring  "mail,[ACCESS::session  data  get  "ses-
sion.ad.last.attr.mail"],otp,$otp,mobile,$mobile"
            log   local0.alert   "Event   [ACCESS::policy   agent_id]
$logstring"
   }
}
```

Esimerkkikoodi 2 Kertakäyttösalananan luominen iRulella

Binary scan ja *set otp* -kohdat luovat ja asettavat kertakäyttösalasanan muuttujaan *$otp*. Tämän jälkeen iRule tallentaa kyseisen muuttujan aukiolevaan sessioon. Lisäksi iRule hakee Active Directorystä käyttäjän puhelinnumeron ja sähköpostiosoitteen hyödyntämällä *ACCESS::session data get* -funktiota. Viimeisenä käyttäjän tiedot kirjoitetaan BIG-IP:n sisäiseen lokiin, josta niitä voidaan myöhemmin hakea. BIG-IP:n iRulet eivät itsessään voi lähettää sähköpostiviestejä, joten tarvitaan Unix-skripti, joka hoitaa sähköpostiviestin lähettämisen. Unix-skriptit ajetaan TMOS-mikrokernelin ulkopuolella BIG-IP:n hallintajärjestelmänä toimivassa CentOS:ssä.

```
#!/bin/bash

tail -n0 -f /var/log/ltm | while read line
do
    var2=`echo $line | grep otp | awk -F'[,]' '{ print $2 }'`
    var3=`echo $line | grep otp | awk -F'[,]' '{ print $3 }'`
    var4=`echo $line | grep otp | awk -F'[,]' '{ print $4 }'`
    if [ "$var3" = "otp" -a -n "$var4" ]; then
       echo Sending pin $var4 to $var2
       echo One Time Password is $var4 | mail -s $var4 $var2
    fi
done
```

Esimerkkikoodi 3 Mail-skripti

Kyseinen skripti lukee BIG-IP:n ltm-lokia, jonne aikaisemmin luotu iRule kirjoittaa käyttäjätiedot. Lokitiedoista poimitaan käyttäjän sähköpostiosoite ja kertakäyttösalasana. *Echo One Time Password is $var4* | *mail -s $var4 $var2* -komennolla salasana lähetetään oikealle henkilölle.

Ennen kuin sähköpostiviestit toimivat, pitää BIG-IP:ssä konfiguroida ssmtp. Tämä tapahtuu muokkaamalla */etc/ssmtp/ssmtp.conf*-tiedostoa. *Mailhub*=-kohtaan lisättiin DCC Oy:n internetoperaattorin tarjoama sähköpostipalvelin. Sähköpostiviestien toimivuus testattiin Unixin *mail* -komennolla. [18.]

4.2.3 Liittäminen Access Policyyn ja SNMP-trap

Testiympäristön tärkeimmät osat ovat valmiita. Viimeinen vaihe on liittää aikaisempaan Access Policyyn haara email-salasanan käyttöä varten. Ennen tätä kuitenkin hiotaan vielä järjestelmän toteutusta. Unix-skripti pitää asettaa käsin päälle, eikä se käynnisty automaattisesti esimerkiksi uudelleenkäynnistyksen jälkeen. Lisäksi se kuluttaa järjestelmän prosessointitehoa ollessaan kokoajan päällä. Tämän korjaamiseksi luodaan SNMP-trap, joka käynnistää skriptin vain ja ainoastaan, kun sitä tarvitaan.

BIG-IP:ssä on helppo luoda omia SNMP-trapeja, sillä ne kirjoitetaan suoraan */config/user_alert.conf*-tiedostoon. SNMP-demonia ei tarvitse käynnistää käsin uudelleen. Se havaitsee, mikäli johonkin sen konfiguraatiotiedostoon tulee muokkauksia, ja käynnistää itsensä uudelleen automaattisesti.

Tiedostoon lisätään seuraavanlainen trap

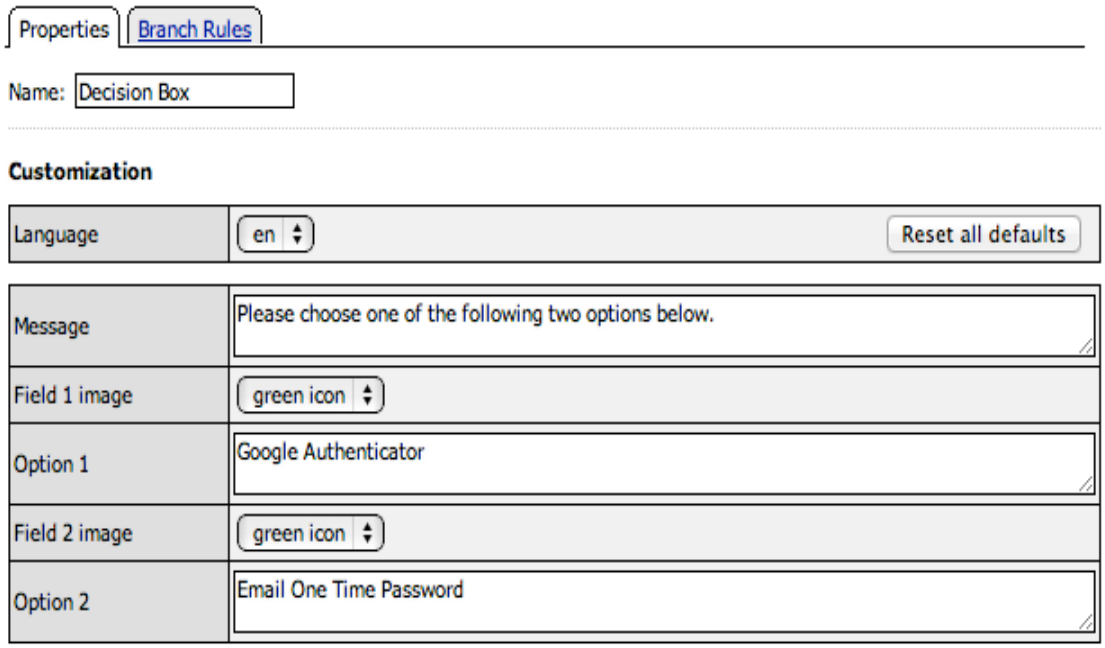
```
alert OTP_SCRIPT "Event otp_verify_email" {
            exec command ="/var/log/otp_script.sh";
}
```

Esimerkkikoodi 4 SNMP-trap

Kyseinen trap käynnistää *otp_script.sh*-skriptin aina kun lokitiedostoihin kirjoitetaan rivi, joka sisältää lauseen "*Event otp_verify_email*".

4.2.4 Viimeistely

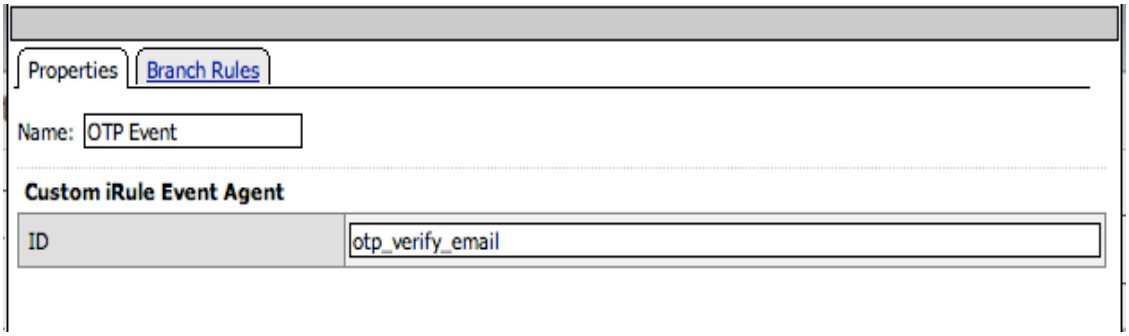
Viimeinen osio aloitetaan lisäämällä *Google Authenticator* -vuokaavioon Decision Box -niminen toiminto, josta käyttäjä valitsee haluamansa autentikoinnin. Toinen haaroista ohjaa käyttäjän Google Authenticatoriin ja toinen sähköpostiautentikointiin.



Kuva 19 Desicion Box -asetukset

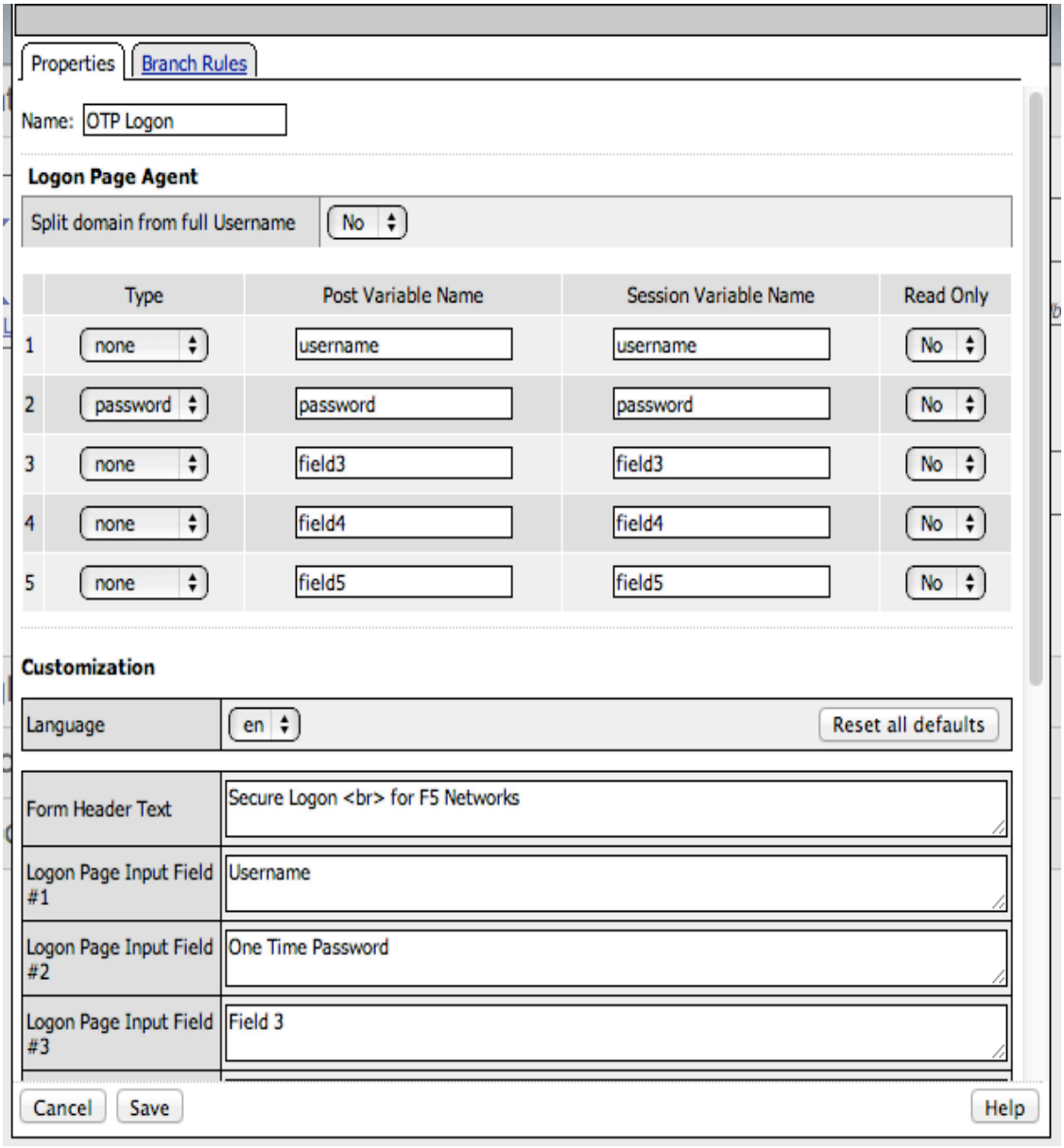
Tämän jälkeen luodaan normaali sisäänkirjautumissivu, jossa käyttäjä antaa käyttäjätunnuksensa. Nämä tunnukset tarkistetaan Active Directorysta, ja samalla APM tarkistaa, että käyttäjällä on määritelty sähköpostiosoite. Sähköpostiosoitteen tarkistus toteutetaan seuraavanlaisella kyselyllä *expr {[mcget {session.ad.last.attr.mail}] != ""}*

Mikäli autentikointi onnistuu ja sähköpostiosoite löytyvät, siirrytään iRule event -tapahtumaan. Tässä kohdassa käynnistetään aikaisemmin luomamme iRule. Jotta oikea iRule käynnistettäisiin, määriteltiin siihen tunniste *opt_verify_email*. Sama tunniste löytyy niin iRulesta, kuin access policyyn laitetusta iRule event -tapahtumasta.

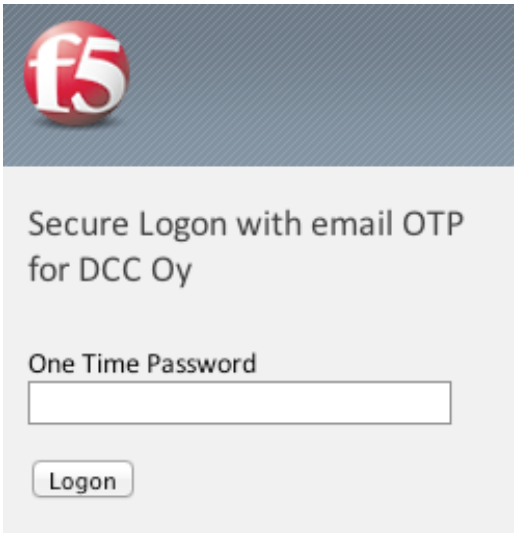


Kuva 20 iRule event

Kun kyseinen iRule käynnistyy, tulee lokiin merkintä, jonka BIG-IP:n SNMP-alerter huomaa ja käynnistää unix-skriptin, jolla salasana lähetetään käyttäjälle. Tämän jälkeen käyttäjä ohjataan uuteen sisäänkirjautumissivuun, jossa on vain yksi syöttökenttä kertakäyttösalasanalle. Kertakäyttösalasanan tarkistus tehdään vertailemalla juuri syötettyä salasanaa aikaisemmin iRulen sessiomuuttujiin tallentamaa salasanaa. Tämä tapahtuu komennolla *expr { [mcget {session.user.otp}] == [mcget -secure {session.logon.last.password}] } .*

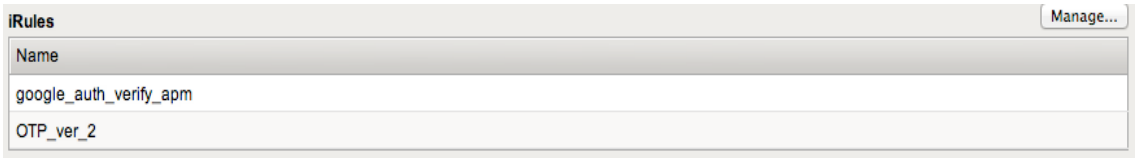


Kuva 21 Kertakäyttösalasanan syöttäminen



Kuva 22 Kertakäyttösalasanan syöttämissivu

Viimeisenä iRule liitetään virtuaaliserveriin ja tämän jälkeen email kertakäyttösalasana on toiminnassa.



Kuva 23 Resurssit

4.3 TUPAS-tunnistautuminen

TUPAS-tunnistautumisessa käyttäjä käyttäjä autentikoidaan hänen pankkitunnuksiaan vastaan. Autentikoinnin hoitaa pankki, ja se perustuu HTTP-lomakkeeseen (HTTP-Form), jonka palveluntarjoaja lähettää pankille. Lomakkeen kentät ovat tarkkaan määriteltyjä Finanssialan keskusliiton toimesta. Lisäksi Finassialan keskusliitto asettaa muita vaatimuksia TUPAS-tunnistautumiselle, esimerkiksi kaiken liikenteen tulee olla SSL-salattua. [19, s. 4-8.]

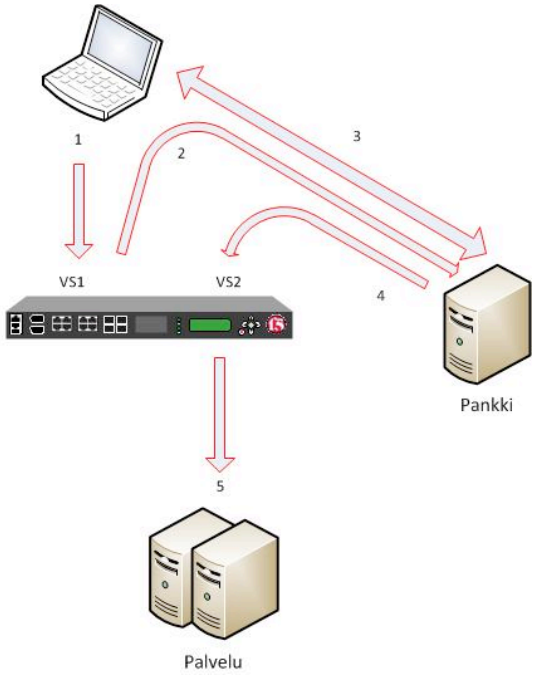
FORM-TIETORYHMÄ			
Kenttä	**Tiedon nimi**	**Pituus**	**Huomautus**
1. Sanomatyyppi	A01Y_ACTION_ID	3 - 4	Vakio, "701"
2. Versio	A01Y_VERS	4	Esim. "0002"
3. Palveluntarjoaja	A01Y_RCVID	10 -15	Asiakastunnus
4. Palvelun kieli	A01Y_LANGCODE	2	ISO 639:n mukainen tunnus: FI = Suomi SV = Ruotsi EN = Englanti
5. Pyynnön yksilöinti	A01Y_STAMP	20	Vvvvkkpphhmmssxxxxxx
6. Yksilöintitiedon tyyppi	A01Y_IDTYPE	2	ks. liite 1
7. Paluuosoite	A01Y_RETLINK	199	OK paluuosoite tunnisteelle
8. Peruuta-osoite	A01Y_CANLINK	199	Paluuosoite peruutuksessa
9. Hylätty-osoite	A01Y_REJLINK	199	Paluuosoite virhetilanteessa
10. Avainversio	A01Y_KEYVERS	4	Avaimen sukupolvitieto
11. Algoritmi	A01Y_ALG	2	01 = MD5 02 = SHA-1 03 = SHA-256
12. Tarkiste	A01Y_MAC	32 - 64	Pyynnön turvatarkiste

Kuva 24 TUPAS-lomakkeen kenttien määritelmät

4.3.1 Toiminta

Tarkoituksena on tutkia mahdollisuutta toteuttaa TUPAS-tunnistautuminen BIG-IP:n ja Access Policy Managerin avulla. Ideana on pystyttää neljä virtuaaliserveriä, joissa sijaitsevat lomakkeen lähetys-, onnistuneet autentikointi-, peruutetut autentikointi- ja epäonnistuneet autentikointipalvelut. Lisäksi tarkoituksena on poimia pankilta tulevasta paluusanomasta tiedot ja laskea niistä tarkistussumma ja verrata sitä pankista saatuun tarkistussummaan. Näin voidaan välttää niin sanotut man in the middle -hyökkäykset. Useimmilla pankeilla on jaossa tunnukset, jotka mahdollistavat TUPAS-tunnistautumisen testaamisen. Tähän demoon valitttiin Nordean tunnukset.

1. Käyttäjä ottaa yhteyden TUPAS-autentikoinit virtuaaliserveriin (VS1)
2. VS1 muodostaa TUPAS-lomakkeen ja lähettää sen pankille käyttäjän selaimen kautta
3. Pankki suorittaa tunnistautumisen käyttäjän syöttämillä tunnuksilla
4. Pankki ohjaa onnistuneen tunnistautumisen palvelun virtuaaliserveriin (VS2)
5. VS2 ohjaa käyttäjän haluttuun palveluun



Kuva 25 TUPAS-toiminta

4.3.2 Toteutus Access Policy Managerin kanssa

Testiympäristö aloitettiin luomalla HTTPS-virtuaaliserveri, johon tarkoituksena on luoda iRule, joka lähettää TUPAS-lomakkeen Nordean testipalveluun. iRulen käynnistystapahtumaksi valitaan HTTP-pyyntö (*when_ACCESS_SESSION_STARTED*). Tämän jälkeen määritellään Tupasin vaatimaan HTTP-lomakkeen arvot muuttujiin, joista niitä on helppo käsitellä. iRuleen pystyy syöttämään HTML-koodia, jolla HTTP-lomake tehdään. Tämä HTML-koodi laitetaan *ACCESS::respond 200 content* -tapahtuman sisään.

TUPAS-tunnistuspyynnön (HTTP-Form) kasaaminen aloitetaan määrittelemällä lomakkeen metodi ja toiminto. Tässä testissä metodina on POST ja toimintona on Nordean tarjoaman tupastunnistautumispalvelun osoite (https://solo3.nordea.fi/cgi-bin/SOLO3011). Lomakkeen ensimmäiseen kenttään syötetään sanoman tyyppi, joka on 701. Se on vakio kaikissa TUPAS-tunnistautumisissa. Toiseen kenttään syötetään pankkikohtainen tunnistuspyyntö-sanoman versionumero. Nordean kohdalla tämä on 0002.

Kolmannessa kentässä on pankin palveluntarjoajalle antama asiakasnumero. Testiympäristössä tämä on 87654321. Neljänteen kenttään määritellään palvelun kieli. Tässä tapuksessa FI eli Suomi. Viides kenttä on puolestaan palveluntarjoajan antama tunnis-

te yksittäiselle tunnistuspyynnölle. Tämä tunniste voi olla viite, asiakasnumero tai yhdistelmä päivämäärästä, kellonajasta ja juoksevasta tunnuksesta sekä viitteestä. Testiympäristöön valittiin aikaleima. iRulessa tarvittava aikaleima saadaan seuraavilla komennoilla:

```
set aika [clock seconds]

set tunnus [clock format $aika -format {%Y%m%d%H%M%S} ]
```

Esimerkkikoodi 5 Aikaleiman luominen iRulella

TUPAS-lomakkeen kuudennessa kentässä on yksilöintitiedon tyyppi, eli tieto jonka palveluntarjoaja tunnistautuvasta henkilöstä haluaa. Vaihtoehtoina ovat salattu perustunnus, selväkielinen perustunnus ja selväkielinen typistetty tunnus. Nordean testissä käytetään arvoa 02 eli selväkielinen perustunnus.

Kentissä seitsemän, kahdeksan ja yhdeksän määritetään osoitteet, joihin tunnistettava asiakas ohjataan tunnistautumisen onnistuessa, asiakkaan keskeyttäessä tunnistautumisen ja tunnistautumisen epäonnistuessa. Nämä osoitteet tehdään BIG-IP:n sisäisillä virtuaaliservereillä.

Kentässä kymmenen on MAC-tarkisteen laskennassa käytetyn avaimen versio, arvo 0001. Kenttä 11 kertoo tarkisteen salaukseen käytetyn algoritmin, vaihtoehtoina ovat MD5, SHA-1 ja SHA-256. Vuonna 2012 siirryttiin pelkästään SHA-256-salaukseen, joten sitä käytettään myös Nordean testiympäristössä. Viimeiseen kenttään lasketaan MAC-tarkiste. Tämä tarkiste koostuu aikaisemmin määritellyistä kentistä 1-11. Niiden arvot luetaan peräkkäin erottaen ne &-merkillä ja loppuun lisätään vielä pankin ja palveluntarjoajan välinen tarkisteavain. Tämä merkkijono salataan käytetyllä algoritmilla. Nordea käyttää tarkistevaimena merkkijonoa LEHTI. Lisäksi saadun tarkisteen kirjaimet muutetaan isoiksi kirjaimiksi. iRulessa SHA-256-salaus toteutetaan seuraavilla komennoilla:

```
binary scan [sha256 $tarkiste] H* sha_tarkiste

set sha_tarkiste [string toupper $sha_tarkiste]
```

Esimerkkikoodi 6 SHA-256-tarkisteen luominen

Lomakkeeseen lisätään vielä submit-painike, joka lähettää lomakkeen Nordean palveluun. [20, s. 7, 13.]

Paluusanoma tulee myös lomakkeena takaisin, mutta samalla sen sisältämät arvot kirjoitetaan takaisinohjaussivun URL:iin query-stringinä. Tässä vaiheessa suoritettiin ensimmäiset testit, jotka menivät läpi. Nordealta saadaan vastaus tunnuksilla 123456 ja salasanalla 1111. Seuraava vaihe on luoda virtuaaliserveri, jonne asiakas ohjataan tunnistautumisen onnistuessa. Pohjana toimii perinteinen HTTPS-virtuaaliserveri. Serveriin liitetään iRule, joka poimii URL:sta query-stringin arvot ja lokittaa ne. Loppuun lisätään if-lause, jossa lasketaan alkuperäisillä arvoilla MAC-tarkiste ja vertaillaan sitä pankin palauttamaan MAC-tarkisteeseen.

Access Policy Manageriin TUPAS-tunnistautuminen on tarkoitus liittää asettamalla virtuaaliserverit osaksi TUPAS Access Policya. Kaikki iRulet asetaan käynnistymään halutuissa kohdissa vuokaaviota. Lomakkeen luonti -iRuleen lisättiin komento, jolla poimittiin Session ID. Tämä Session ID lisätään selaimen kekseihin, kun pankki palauttaa käyttäjän aikaisemmin määritettyyn onnistuneen tunnistautumisen jatko-osoitteeseen. Tämän avulla pyritään palaamaan samaan APM-sessioon kuin TUPAS-lomaketta luodessa. Ongelmaksi kuitenkin ilmenee se, että APM tunnistaa keksin lisäämisen ja ilmoittaa, että sessio on vielä voimassa. Toisena ongelmana on Visual Policy Editorissa tiettyyn kohtaan palaaminen. Niillä kerroilla kun päästiin takaisin samaan sessioon, palattiin kuitenkin vuokaavion alkuun. Projektin tavoitteiden ja aikarajoitteiden takia APM osuudesta TUPAS-tunnistautumisessa luovuttiin ja tilalle kehitettiin yhdessä BIG-IP:n perus-LTM virtuaaliserverissä toimiva ratkaisu.

4.3.3 Toteutus ilman Access Policy Manageria

Yhteen virtuaaliserveriin siirtyminen tarkoittaa kaikkien käytettyjen iRulejen yhdistämistä yhteen iRuleen. Käytännössä skriptin pitää pystyä tunnistamaan missä vaiheessa TUPAS-tunnistautumisprosessia käyttäjä on. Tämä toteutetaan käyttämällä switch-komentoa ja tarkastelemalla käyttäjän URL:lia.

iRulen tapahtumaksi voidaan nyt valita *when_HTTP_REQUEST*. *Switch -glob [HTTP::uri]* -komennolla tarkastellaan URL:lia kun HTTP-pyyntö tulee. Tämän avulla määritetään esimerkiksi jokaiselle pankille oma sivu, jossa luodaan kunkin pankin omilla tiedoilla TUPAS-lomake. Lomake toteutetaan *HTTP::respond 200 content* -komennon sisään, jolloin BIG-IP tulkitsee kyseisen komennon sisään kirjoitetut komennot HTML-koodina.

```
when HTTP_REQUEST {
    switch -glob [HTTP::uri] {
        /nordea/ {
             log local0. "URI: [HTTP::uri]
             log local0. "Nordea TUPAS"

             set aika [clock seconds]
             set tunnus [clock format $aika -format {%Y%m%d%H%M%S} ]

       set mac "[class lookup sanomatyyppi nordea]&[class lookup
       versio nordea]&[class lookup asiakastunnus nordea]&[class
       lookup kieli nordea]&$tunnus&[class lookup yksilotieto
       nordea]&[class lookup jatko nordea]&[class lookup peruutus
       nordea]&[class lookup virhe nordea]&[class lookup avainversio
       nordea]&[class lookup alg nordea]&[class lookup tarkiste
       nordea]&"

       binary scan [sha256 $mac] H* sha_mac
       set sha_mac [string toupper $sha_mac]

             HTTP::respond 200 content "
                  <html>

                  <head>
                  <title>Testi</title>
                  </head>

                  <body>
                  Tassa on form
                  <FORM METHOD=\"POST\" ACTION=\"[class lookup oso-
ite nordea]\">
                  <INPUT NAME=\"A01Y_ACTION_ID\" TYPE=\"hidden\"
VALUE=\"[class lookup sanomatyyppi nordea]\">
                  <INPUT NAME=\"A01Y_VERS\"  TYPE=\"hidden\" VAL-
UE=\"[class lookup versio nordea]\">
                  <INPUT NAME=\"A01Y_RCVID\" TYPE=\"hidden\" VAL-
UE=\"[class lookup asiakastunnus nordea]\">
                  <INPUT NAME=\"A01Y_LANGCODE\" TYPE=\"hidden\" VAL-
UE=\"[class lookup kieli nordea]\">
                  <INPUT NAME=\"A01Y_STAMP\" TYPE=\"hidden\" VAL-
UE=\"$tunnus\">
                  <INPUT NAME=\"A01Y_IDTYPE\" TYPE=\"hidden\" VAL-
UE=\"[class lookup yksilotieto nordea]\">
                  <INPUT NAME=\"A01Y_RETLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup jatko nordea]\">
                  <INPUT NAME=\"A01Y_CANLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup peruutus nordea]\">
                  <INPUT NAME=\"A01Y_REJLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup virhe nordea]\">
                  <INPUT NAME=\"A01Y_KEYVERS\" TYPE=\"hidden\" VAL-
UE=\"[class lookup avainversio nordea]\">
                  <INPUT NAME=\"A01Y_ALG\" TYPE=\"hidden\" VAL-
UE=\"[class lookup alg nordea]\">
                  <INPUT NAME=\"A01Y_MAC\" TYPE=\"hidden\" VAL-
UE=\"$sha_mac\">
                  <input type=\"submit\" />
                  </FORM>

                  </body>

                  </html>
               "
 }
```

Esimerkkikoodi 7 TUPAS-lomake Nordean verkkopankkiin

TUPAS-lomakkeen arvot on tallennettu datagroup-tiedostoihin, joista ne luetaan class lookup -komennolla. Esimerkiksi *class lookup asiakastunnus nordea* etsii nordea-nimisestä datagroupista arvon muuttujalle asiakastunnus. Arvot sijoitettiin datagroupeihin, jotta niitä ei tarvitse koodata iRuleen ja niitä on helpompi käsitellä datagroupeissa. Ainoat muuttujat, jotka luodaan iRulessa, ovat tunnus, jolla istunto tunnistetaan, ja MAC-tarkiste, joka pitää laskea. Kaikki muut arvot ovat kiinteitä, joten ne voivat sijaita datagroupeissa. Koodiesimerkissä 7 on esimerkki Nordean lomakkeenlähetys-sivusta. Vastaavat koodinpätkät ja datagroupit luotiin myös Aktialle, Danskebankille ja Osuuspankille.

Pankkikohtaisten lomakeiden luonnin jälkeen määritellään, miten toimitaan, kun TUPAS-tunnistautuminen keskeytetään tai siinä tapahtuu virhe. Tämän Proof of Concept -projektin kannalta niille ei luotu omia palautesivuja, vaan riittää että kyseisistä tapahtumista kirjoitetaan maininnat lokiin.

```
/canceled/ {
   log local0. "Tunnistautuminen keskeytettiin"
   log local0. "URI: [HTTP::uri]"
}

/rejected/ {
   log local0. "Tunnistautumisessa virhe"
   log local0. "URI: [HTTP::uri]"
}
```

Esimerkkikoodi 8 Epäonnistunut autentikointi - ja keskeytyshaara

Tunnistautumisen onnistuessa käyttäjä palautetaan */auth_ok/** -haaraan, jossa tähden paikalle pankki sijoittaa query stringin, josta löytyvät pankin palauttaman lomakkeen arvot. iRule parsii nämä arvot ja lokittaa ne.

```
/auth_ok/* {
            log local0. "URI: [HTTP::uri]"

            log local0. "----------------------"
            log local0. "URI Information"
            log local0. "----------------------"
            log local0. "HTTP::uri: [HTTP::uri]"
            log local0. "----------------------"
            log local0. "Path Information"
            log local0. "----------------------"
            log local0. "HTTP::path: [HTTP::path]"
            set depth [URI::path [HTTP::uri] depth]

           for {set i 1} {$i <= $depth} {incr i} {
              set dir [URI::path [HTTP::uri] $i $i]
              log local0. "dir\[$i\]: $dir"
            }

            log local0. "Basename: [URI::basename [HTTP::uri]]"
            log local0. "----------------------"
            log local0. "Query Information"
            log local0. "----------------------"
            log local0. "HTTP::query: [HTTP::query]"

            set namevals [split [HTTP::query] "&"]
            for {set i 0} {$i < [llength $namevals]} {incr i} {
              set params [split [lindex $namevals $i] "="]
              set pnum [expr $i+1]
              log local0. "Param\[$pnum\]: [lindex $params 0]"
              log local0. "Value\[$pnum\]: [URI::query [HTTP::uri]
[lindex $params 0]]"

              set value$pnum [URI::query [HTTP::uri] [lindex $params
0]]
            }
}
```

Esimerkkikoodi 9 Onnistunut autetikointi -haara

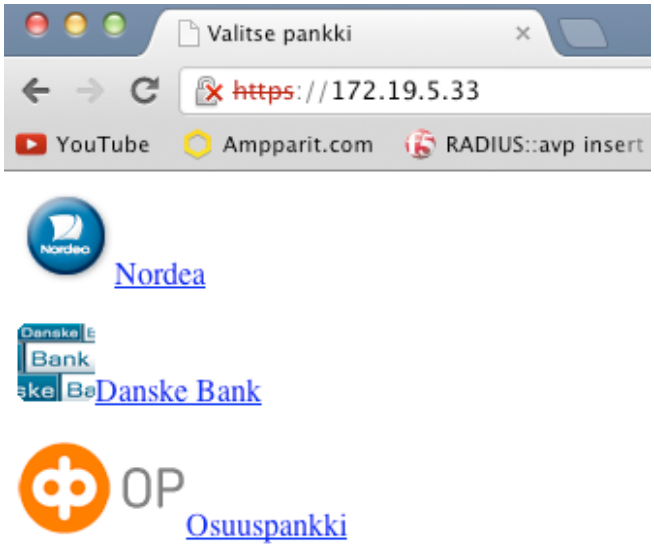
Viimeisimpänä luodaan default-haara, jonne käyttäjä ohjataan niissä tapauksissa, kun käyttäjän käyttämä URL ei vastaa mitään switch-määritystä. Tähän default-haaraan luodaan HTML-sivu, jossa käyttäjä saa valita haluamansa pankin. HTML-sivun luontiin haluttiin testata hieman erilaista toteutustapaa. Sivun lähdekoodi ja sen sisältämät kuvat (pankkien logot) tallennetaan BIG-IP:n kovalevylle iFile-muodossa. iFile on tiedosto, jota voidaan kutsua iRuleilla. Tämän käyttäminen mahdollistaa vaativampienkin verkkosivujen toteutuksen virtuaaliserverissä iRulejen avulla.

```
default {
                log local0. "Default"
                log local0. "URI: [HTTP::uri]"
                if {[HTTP::uri] eq "/"}{
                     HTTP::respond 200 content [ifile get
pankin_valinta]
                  }
                elseif {[HTTP::uri] eq "/nordea_logo.gif"}{
                     HTTP::respond 200 content [ifile get nordea]
                }
                elseif {[HTTP::uri] eq "/db_logo.gif"}{
                     HTTP::respond 200 content [ifile get dans-
ke_bank]
                }
                elseif {[HTTP::uri] eq "/op-logo.gif"}{
                     HTTP::respond 200 content [ifile get op]
                }

          }
```

Esimerkkikoodi 10 Pankin valintahaara

Ensimmäisenä iRulessa haetaan pankin_valinta-niminen iFile. Tämä tiedosto sisältää HTML-koodin, jossa on linkit jokaisen pankin TUPAS-lomakkeeseen HTML-viittaukset pankkien logoihin. Seuraavaksi haetaan jokaisen pankin logo BIG-IP:n kovalevyltä erikseen. Tiedostot luetaan HTML-kääntäjän avulla käyttämällä jo aiemmin esiteltyä *HTTP::respond 200 content* -komentoa.



Kuva 26 iFile-tiedostojen avulla luotu HTML-sivu

5 BYOD

Nykypäivänä teknologia kehittyy kovaa vauhtia ja uusia päätelaitteita tuodaan markkinoille entistä nopeammin. Älypuhelimien ja tablettien yleistymisen myötä yritysten tulee miettiä omat päätelaite- ja niiden tunnistuspolitiikkansa uudestaan. Puhutaan niin sanotusta BYOD:sta, Bring Your Own Device, eli työntekijät käyttävät omia päätelaitteitaan työntekoon. Näyttää siltä, että BYOD on tullut jäädäkseen ja IT-organisaatioiden tulee sopeutua siihen. Tarvitaan BYOD-politiikka, jolla sen tuomia uhkia vältetään ja mahdollisuuksia voidaan hyödyntää. BYOD-käsite ja se, millainen BYOD-politiikan tulisi olla, ovat vielä yleisesti varsin epäselviä. Vastaavasti BYOD-politiikkan toteutusmekanismit ja apuvälineet ovat vielä kehitysvaiheessa. Lisäksi ei ole juurikaan BYODia kattavia standardeja.

BYOD-politiikan toteutus vaatii huolellista suunnittelua ja toteutusta. Heikohko toteutus saattaa aiheuttaa muun muassa tietoturvaongelmia yritykselle. Huolellisesti suunniteltuna se saattaa puolestaan tehostaa työntekijöiden tuottavuutta ja vähentää yrityksen IT-kustannuksia. Toteutuksen tulisi sisältää kirjallinen osuus, jossa sovitaan käytännöistä, mitä omilla päätelaitteilla saa tehdä yrityksen verkoissa ja miten toimitaan esimerkiksi älypuhelimen kadotessa. Lisäksi toteutukseen tarvitaan myös tekninen puoli. Omille laitteille pitää pystyä tuomaan kaikki tarvittavat resurssit ja käyttöoikeudet yrityksen verkosta. Käyttäjien käyttöoikeuksia pitäisi pystyä hallitsemaan ja toimintaa seuraamaan samalla tavalla kuin kaikkia muitakin laitteita yrityksen verkossa. Lisäksi uudet päätelaitteet saattavat aiheuttaa muutoksia yrityksen tietoliikenneinfrastruktuurissa. Epämääräistä on vielä, kuka ja kuinka BYOD-laitteita hallitaan, mitä ohjelmia ja mitä dataa niissä ajetaan sekä mihin ja kuinka vedetään raja käyttäjän ja organisaation välille.

BYOD on osa tulevaisuutta ja omien päätelaitteiden merkitys, ja määrä tulevat kasvamaan huomattavasti seuraavina vuosina. Juniper Reasearch arvioi BYOD-laitteiden määrän tuplaantuvan vuoteen 2014 mennessä. Gartner puolestaan pitää BYOD-laitteiden tuloa yritysmaailmaan yhtä merkittävänä muutoksena yritysten päätelaitekantaan kuin PC-koneiden yleistymistä. [21; 22.]

5.1 Uhat ja ongelmat

Omien päätelaitteiden käyttö työntekoon asettaa tiettyjä uhkia niin yritykselle kuin työntekijälle. Ongelmia aiheuttaa päätelaitteiden laaja kirjo. Yritysten resurssien ja tietoliikenneinfrastruktuurin pitää tukea yhä useampaa erilaista päätelaitetta, jotka eivät välttämättä ole yrityksen hallinnassa. Pahimmillaan yritysten sovelluksia joudutaan koodaamaan uudestaan tukemaan erilaisia mobiilikäyttöjärjestelmiä ja vanhempaa infrastruktuuria joudutaan uusimaan.

Yritysten kannalta eräs suurimmista uhista on yrityksen tiedon tallentuminen työntekijöiden omiin päätelaitteisiin ja mahdollisuus käsitellä dataa älypuhelimien kautta. Kun yrityksen tietoja pystytytään käsittelemään ja tallentamaan älypuhelimiin ja tabletteihin, on yrityksen vaikea valvoa, etteivät tiedot poistu yrityksestä ja päädy väärin käsiin. Ongelmia aiheuttaa myös se, miten ohjelmia käytetään. Pystyykö BYOD-laite ajamaan sovelluksen itse vai tarvitaanko BYOD-laitteita varten VDI-ratkaisu, jossa sovellus ajetaan. BYOD-laitteet tuovat myös haasteita tietoturvalle. Sisältävätkö BYOD-laitteet haittaohjelmia, kuinka haittaohjelmat tunnistetaan ja miten niiltä suojaudutaan, ovat oleellisia kysymyksiä tietoturvapuolella.

Infrastruktuuri saattaa kokea muutoksia, kun mobiililaitteita tuodaan yrityksen verkkoon. Pitää varmistaa, että BYOD-laitteiden käyttöön tarkoitetut resurssit ja sovellukset tukevat kyseisiä laitteita. Lisäksi verkkoliikenteen määrä kasvaa, kun työntekijät liittävät älypuhelimensa yrityksen verkkoon. Tämä näkyy erityisesti langattomien verkkojen liikenteessä. Datakeskus puolella erilaiset VDI-ratkaisut (Virtual Desktop Infrastructure) nousevat esille. Tietoturvaratkaisut täytyy mahdollisesti miettiä uusiksi. [23; 24.]

Infoworld.com:n lokakuussa 2012 julkaiseman artikkelin mukaan työntekijöiden suurin huolenaihe on heidän yksityisyytensä. Suurin osa tämän hetkisistä BYOD-ratkaisuista älypuhelimille ja tabletti-laitteille perustuu laitteisiin ladattavaan sovellukseen. Nämä MDM-sovellukset (Mobile Device Management) pystyvät usein seuraamaan laitteen sijaintia GPS-yhteyden kautta niin työaikoina kuin niiden ulkopuolella. Laitteiden tiedot voidaan pyyhkiä etäältä esimerkiksi, kun työntekijän älypuhelin hukkuu. Laitteen muistin tyhjentäminen etäältä poistaa useimmiten myös käyttäjän laitteeseen tallentaman yksityistiedon, kuten tekstiviestit. Lisäksi MDM-sovelluksilla pystytään myös tarkastelemaan, mitä sovelluksia laitteisiin asennetaan. Ne antavat mahdollisuuden tarkastella

laitteen sisältämiä tietoja. Näiden toimenpiteiden laillisuus on herättänyt keskustelua. [25.]

Työntekijöiden kannalta myös uhkana voidaan pitää mahdollisesti kasvavia kuluja, mikäli yrityksen BYOD-politiikka pakottaa työntekijän itse hankkimaan käyttämänsä laitteet. Lisäksi mobiililaitteissa tämä saattaa näkyä kasvavina matkapuhelinlaskuina, kun työntekijän dataliikenne matkapuhelimessa kasvaa. Myös mahdolliset huoltokulut saattavat kasautua työntekijän hoidettavaksi. [23.]

5.2 APM ja BYOD

BIG-IP ja APM istuvat tyypillisesti välittömästi palveluiden edessä, jossa ne pystyvät valvomaan liikennettä ja hallitsemaan palveluiden ja datan käyttöä. BIG-IP voi tunnistaa käyttäjän, BYOD-laitteen ja ymmärtää, mihin niitä kulloinkin käytetään ja sallia niiden käytön yrityksen määrittelemän BYOD-politiikan mukaisesti. BIG-IP:n avulla voidaan antaa tai estää pääsy yrityksen resursseihin. Samalla resurssienkäytön lokitus voidaan suorittaa BIG-IP:n avulla. Näin BIG-IP ja APM täydentävät yrityksen BYOD-politiikan toteutusta.

BIG-IP ja Access Policy Manager voivat olla osa BYOD-politiikan toteutusta. Tällä hetkellä BIG-IP:n ja APM:n avulla voidaan pääasiassa toteuttaa päätelaitteen ja käyttäjän pääsynhallintaa haluttuihin sovelluksiin. Työntekijä voi ilmoittaa käyttävänsä omaa päätelaitettaan, jolloin IT-ylläpito voi APM:n avulla tunnistaa ja valvoa kyseisen laitteen käyttöä. Tulevaisuudessa BYOD-puolelle tuodaan uusia mahdollisuuksia. Jatkossa BIG-IP:n avulla voidaan myös hallinoida BYOD-laitteissa ajettavia yrityksen omia sovelluksia ja dataa. BYOD laitehallinnasta (Mobile Device Management, MDM) ollaan siirtymässä kohti BYOD sovellustenhallintaa (Mobile Application Management, MAM), jossa yritys voi hallita BYOD-laitteissa olevia yrityksen omia sovelluksia ja dataa. Ne pidetään erillään, eikä yrityksellä ole pääsyä käyttäjän sovelluksiin tai dataan.

6 Yhteenveto

Erilaiset sovellukset ja palvelut vaativat erilaisia käyttäjäntunnistusmenetelmiä. Palveluntarjoajat päättävät, millaisen autentikoinnin he haluavat palvelulleen. Bisneskriittiset

palvelut, kuten verkkopankit, vaativat tehokkaamman ja turvallisemman käyttäjäntunnistamisprosessin kuin esimerkiksi keskustelupalstat. Tässä insinöörityössä tarkastellaan yleisimpiä autentikointitapoja, esitellään F5 Networksin BIG-IP ADC-laite ja sen tuomia etuja autentikointiin. Lisäksi pohditaan tämän hetken suurinta ongelmaa päätelaite- ja käyttäjäntunnistuksessa, BYOD:ia.

Insinöörityön teknisessä osuudessa rakennettiin DCC Oy:lle testiympäristö, jossa voidaan esitellä Google Authenticator -autentikointia ja sähköpostiin lähetettävää kertakäyttösalasanaa sekä tehtiin Proof of Concept TUPAS-tunnistautumisesta. Toteutuksessa käytettiin F5 Networksin BIG-IP -laitteita ja Access Policy Manager -moduulia.

Teknisen osuuden pohjalta voidaan todeta, että erilaisten autentikointimenetelmien toteuttaminen ja ylläpito ovat helppoa BIG-IP APM:n avulla. Laitteiston avulla voidaan toteuttaa erilaisia TFA-ratkaisuja ja yhdistellä niitä toisiinsa, kuten Google Authenticatorin ja Email OTP:n yhdistämisestä samaan pääsynhallintasäännöstöön liittämisestä voidaan huomata. Lisäksi BIG-IP taipuu monimutkaisempiinkin toteutuksiin, esimerkiksi TUPAS-tunnistautumiseen.

Autentikoinnin siirtäminen ADC-laitteelle luo useita etuja ja helpotuksia ylläpidolle. Näkyvin hyöty on keskitetty autentikointi. Samaa ADC-laitetta voidaan käyttää autentikointiproxyna usealle eri sovellukselle ja palvelulle. Samalla poistuu tarve koodata autentikointi erikseen jokaiseen sovellukseen. Eri autentikointimenetelmien ylläpito helpottuu, kun ne sijaitsevat yhdessä laitteessa ja eri menetelmiä voidaan yhdistellä toisiinsa. Samaan palveluun voidaan tarvittaessa kohdistaa erilaisia autentikointimenetelmiä riippuen käyttäjäryhmästä, päätelaitteesta tai siitä mistä sitä käytetään.

Parhaimmillaan autentikoinnin ulkoistamisella päästään eroon useista komponenteista. Esimerkiksi käyttämällä BIG-IP APM:ää Nordic Edgen SMS kertakäyttösalasanan kanssa voidaan APM:llä korvata muun muassa OTP-palvelinkomponentti, Apache-palvelin ja tietokanta OTP-käyttäjistä. Nordic Edgeltä tarvitaan vain SMS gateway. Tässä tapauksessa APM hoitaa kertakäyttösalasanan luomisen ja tarkistuksen sekä käyttäjän autentikoinnin. Koska kaikki liikenne kulkee palveluihin ADC-laitteiden lävitse, ei myöskään tarvita erillistä liikenteen ohjausta tai reititystä OPT-autentikointikomponenteille. Useista irrallisista komponenteista luopuminen luonnollisesti vähentää ylläpidettävien laitteiden määrä ja helpottaa vianselvitystä.

Lähteet

1 Teknisen ICT-ympäristön tietoturvataso-ohje. 2012. Verkkodokumentti. <https://www.vahtiohje.fi/c/document_library/get_file?uuid=5a273c6e-2935-4bbf-a4c6-f00e0f878db5&groupId=10128&groupId=10229>. Luettu 13.12.2012.

2 Two-factor authentication. Verkkodokumentti. <http://en.wikipedia.org/wiki/Two-factor_authentication>. Luettu 2.11.2012.

3 Strong authentication. Verkkodokumentti. <http://www.rsa.com/glossary/default.asp?id=1080>. Luettu 2.12.2012.

4 Laki sähköisestä tunnistautumisesta ja sähköisestä allekirjoittamisesta. 2009. Verkkodokumentti. <http://www.finlex.fi/fi/laki/alkup/2009/2009617>. Luettu 29.11.2012.

5 Korhonen, Eeva. 2010. Sisäverkon ohje, tunnistautuminen. Verkkodokumentti. <https://www.vahtiohje.fi/web/guest/tunnistautuminen;jsessionid=1ECB34973152927372961AE4A523B173E580CBEAA60008549951CB8B6FF46C51A060B90319AEC6D0236BDA>. Luettu 2.1.2013.

6 Varmenne. 2007. Verkkodokumentti. Viestintäkeskus. <http://www.ficora.fi/index/palvelut/palvelutaiheittain/tietoturva/pki/varmenne.html>. Luettu 18.11.2012.

7 Laatuvarmenne. Verkkodokumentti. <http://fineid.fi/default.aspx?docid=4090&site=9&id=293>. Luettu 18.11.2012.

8 Lakisähköisistä allekirjoituksista. 2003. Verkkodokumentti. <http://www.finlex.fi/fi/laki/alkup/2003/20030014> Luettu 18.11.2012.

9 Mobiiliasiointivarmenne, varmennepolitiikka. 2011. Verkkodokumentti. <http://mobiilivarmenne.fi/documents/Mobiiliasiointivarmenne-Varmennepolitiikka.pdf>. Luettu 19.11.2012.

10 Single sign-on. 2012. Verkkodokumentti. <http://en.wikipedia.org/wiki/Single_sign-on>. Luettu 1.12.2012.

11 SAML. Verkkodokumentti. <https://www.pingidentity.com/resource-center/SAML-Tutorials-and-Resources.cfm>. Luettu 1.12.2012.

12 Application Delivery Controller. F5 Networks. Verkkodokumentti. <http://www.f5.com/glossary/application-delivery-controller/>. Luettu 22.12.2012.

13 Salchow Jr, Ken. 2011. TMOS: Redefining the Solution. Verkkodokumentti. <www.f5.com/pdf/white-papers/tmos-wp.pdf>. Luettu 22.12.2012.

14 MacVittie, Lori. 2011. The Full-Proxy Data Center Architecture. Verkkodokumentti. <https://devcentral.f5.com/blogs/us/the-full-proxy-data-center-architecture>. Luettu 22.12.2012.

15 iRule. Verkkodokumentti. <https://devcentral.f5.com/tabid/1082202/Default.aspx>. Luettu 22.12.2012.

16 Google Authenticator. Verkkodokumentti. <http://code.google.com/p/google-authenticator/>. Luettu 3.10.2012.

17 Watkins, George. 2012. Two-Factor Authentication With Google Authenticator And APM. Verkkodokumentti. <https://devcentral.f5.com/Tutorials/TechTips/tabid/63/articleType/ArticleView/articleId/1088525/Two-Factor-Authentication-With-Google-Authenticator-And-APM.aspx>. Luettu 3.10.2012.

18 SOL13180: Configuring the BIG-IP system to deliver locally-generated email messages (11.x). 2011. Verkkodokumentti. <http://support.f5.com/kb/en-us/solutions/public/13000/100/sol13180.html?sr=24072538>. Luettu 15.10.2012.

19 Pankkien TUPAS-tunnistuspalvelu palveluntarjoajille. 2011. Finanssialan Keskusliitto. Verkkodokumentti. <http://www.fkl.fi/teemasivut/sahkoinen_asiointi/Dokumentit/Tupas-varmennepalvelu_v23c.pdf>. Luettu 1.12.2012.

20 E-tunniste, palvelunkuvaus. 2011. Nordea. Verkkodokumentti. <http://www.nordea.fi/sitemod/upload/root/content/nordea_fi_fi/yritysasiakkaat/yhteys_pankkiin/palvelukuvaukset/e_tunniste.pdf>. Luettu 2.12.2012

21 Gartner Says Bring Your Own Device Programs Herald the Most Radical Shift in Enterprise Client Computing Since the Introductuin of the PC. 2012. Verkkodokumentti. <http://www.gartner.com/it/page.jsp?id=2136615>. Luettu 10.1.2013.

22 Brandel, Mary. 2012. Seven BYOD policy essentials. NETWORKWORLD. Verkkodokumentti. <http://www.networkworld.com/news/2012/100112-byod-262932.html>. Luettu 10.1.2013.

23 Burke, Joshua. 2011. Bring Your Own Device: Risks and rewards. TechRepublic. Verkkodokumentti. <http://www.techrepublic.com/blog/tech-manager/bring-your-own-device-risks-and-rewards/7075?tag=content;siu-container>. Luettu 11.1.2013.

24 Lowe, Scott. 2011. BYOD's impact on the data center. TechRepublic. Verkkodokumentti. <http://www.techrepublic.com/blog/networking/byods-impact-on-the-data-center/4948?tag=content;siu-container>. Luettu 11.1.2013.

25 Gruman, Galen. 2012. It's war! BYOD exposes IT's deep distrust of users. Infoworld. Verkkodokumentti. <http://www.infoworld.com/d/consumerization-of-it/its-war-byod-exposes-its-deep-distrust-of-users-202989>. Luettu 11.1.2013.

Google_auth_verify_apm - iRule

```
when ACCESS_POLICY_AGENT_EVENT {
    if { [ACCESS::policy agent_id] eq "ga_code_verify" } {
        ### Google Authenticator verification settings ###

        # lock the user out after x attempts for a period of x seconds
        set static::lockout_attempts 3
        set static::lockout_period 30

        # logon page session variable name for code attempt form field
        set static::ga_code_form_field "ga_code_attempt"

        # key (shared secret) storage method: ldap, ad, or datagroup
        set static::ga_key_storage "datagroup"

        # LDAP attribute for key if storing in LDAP (optional)
        set static::ga_key_ldap_attr "google_auth_key"

        # Active Directory attribute for key if storing in AD (optional)
        set static::ga_key_ad_attr "google_auth_key"

        # datagroup name if storing key in a datagroup (optional)
        set static::ga_key_dg "google_auth_keys"


        ####################################
        ### DO NOT MODIFY BELOW THIS LINE ###
        ####################################

        # set lockout table
        set static::lockout_state_table "[virtual name]_lockout_status"

        # set variables from APM logon page
        set username [ACCESS::session data get session.logon.last.username]
        set ga_code_attempt [ACCESS::session data get ses-
sion.logon.last.$static::ga_code_form_field]

        # retrieve key from specified storage
        set ga_key ""

        switch $static::ga_key_storage {
            ldap {
                set ga_key [ACCESS::session data get ses-
sion.ldap.last.attr.$static::ga_key_ldap_attr]
            }
            ad {
                set ga_key [ACCESS::session data get ses-
sion.ad.last.attr.$static::ga_key_ad_attr]
            }
            datagroup {
                set ga_key [class lookup $username $static::ga_key_dg]
            }
        }

        # increment the number of login attempts for the user
        set prev_attempts [table incr -notouch -subtable $stat-
ic::lockout_state_table $username]
        table timeout -subtable $static::lockout_state_table $username $stat-
ic::lockout_period
```

```
        # verification result value:
        # 0 = successful
        # 1 = failed
        # 2 = no key found
        # 3 = invalid key length
        # 4 = user locked out

        # make sure that the user isn't locked out before calculating GA code
        if { $prev_attempts <= $static::lockout_attempts } {

            # check that a valid key was retrieved, then proceed
            if { [string length $ga_key] == 16 } {
                # begin - Base32 decode to binary

                # Base32 alphabet (see RFC 4648)
                array set static::b32_alphabet {
                    A 0  B 1  C 2  D 3
                    E 4  F 5  G 6  H 7
                    I 8  J 9  K 10 L 11
                    M 12 N 13 O 14 P 15
                    Q 16 R 17 S 18 T 19
                    U 20 V 21 W 22 X 23
                    Y 24 Z 25 2 26 3 27
                    4 28 5 29 6 30 7 31
                }

                set ga_key [string toupper $ga_key]
                set l [string length $ga_key]
                set n 0
                set j 0
                set ga_key_bin ""

                for { set i 0 } { $i < $l } { incr i } {
                    set n [expr $n << 5]
                    set n [expr $n + $static::b32_alphabet([string index
$ga_key $i])]
                    set j [incr j 5]

                    if { $j >= 8 } {
                        set j [incr j -8]
                        append ga_key_bin [format %c [expr ($n & (0xFF << $j))
>> $j]]
                    }
                }

                # end - Base32 decode to binary

                # begin - HMAC-SHA1 calculation of Google Auth token

                set time [binary format W* [expr [clock seconds] / 30]]

                set ipad ""
                set opad ""

                for { set j 0 } { $j < [string length $ga_key_bin] } { incr j
} {
                    binary scan $ga_key_bin @${j}H2 k
                    set o [expr 0x$k ^ 0x5C]
                    set i [expr 0x$k ^ 0x36]
                    append ipad [format %c $i]
                    append opad [format %c $o]
                }

                while { $j < 64 } {
```

```
                    append ipad 6
                    append opad \\
                    incr j
                }

                binary scan [sha1 $opad[sha1 ${ipad}${time}]] H* token

                # end - HMAC-SHA1 calculation of Google Auth hex token

                # begin - extract code from Google Auth hex token

                set offset [expr ([scan [string index $token end] %x] & 0x0F)
<< 1]
                set ga_code [expr (0x[string range $token $offset [expr $off-
set + 7]] & 0x7FFFFFFF) % 1000000]
                set ga_code [format %06d $ga_code]

                # end - extract code from Google Auth hex token

                if { $ga_code_attempt eq $ga_code } {
                    # code verification successful
                    set ga_result 0
                } else {
                    # code verification failed
                    set ga_result 1
                }
            } elseif { [string length $ga_key] > 0 } {
                # invalid key length, greater than 0, but not length not equal
to 16 chars
                set ga_result 3
            } else {
                # could not retrieve user's key
                set ga_result 2
            }
        } else {
            # user locked out due to too many failed attempts
            set ga_result 4
        }

        # set code verification result in session variable
        ACCESS::session data set session.custom.ga_result $ga_result
    }
}


.
```

Google_auth_soft_token_gen - iRule

```
when HTTP_REQUEST {
  set account [URI::query [HTTP::uri] "account"]
  set domain [URI::query [HTTP::uri] "domain"]
  set secret [URI::query [HTTP::uri] "secret"]
  set qr_code [URI::query [HTTP::uri] "qr_code"]

  if { ([HTTP::path] starts_with "/ga_secret_generator") && ($account ne "")
&& ($domain ne "") } {
    if { [string length $secret] <= 10  } {
      set secret [b64encode [md5 [expr rand()]]]
    }

    set secret [string range $secret 0 9]

    array set b32_alphabet_inv {
       0 A  1 B  2 C  3 D
       4 E  5 F  6 G  7 H
       8 I  9 J 10 K 11 L
      12 M 13 N 14 O 15 P
      16 Q 17 R 18 S 19 T
      20 U 21 V 22 W 23 X
      24 Y 25 Z 26 2 27 3
      28 4 29 5 30 6 31 7
    }

    set secret_b32 ""
    set l [string length $secret]
    set n 0
    set j 0

    #      encode      loop      is      outlined      in      RFC      4648
(http://tools.ietf.org/html/rfc4648#page-8)
    for { set i 0 } { $i < $l } { incr i } {
      set n [expr $n << 8]
      set n [expr $n + [scan [string index $secret $i] %c]]
      set j [incr j 8]

      while { $j >= 5 } {
        set j [incr j -5]
        append secret_b32 $b32_alphabet_inv([expr ($n & (0x1F << $j)) >> $j])
      }
```

```
    }

    # pad final input group with zeros to form an integral number of 5-bit
groups, then encode
    if { $j > 0 } { append secret_b32 $b32_alphabet_inv([expr $n << (5 - $j) &
0x1F]) }

    # if the final quantum is not an integral multiple of 40, append "=" pad-
ding
    set pad [expr 8 - [string length $secret_b32] % 8]
    if { ($pad > 0) && ($pad < 8) } { append secret_b32 [string repeat = $pad]
}

    set                                                    ga_qr_code_link
"https://chart.googleapis.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://
totp/"
    append ga_qr_code_link "$account@$domain"
    append ga_qr_code_link "%3Fsecret%3D"
    append ga_qr_code_link $secret_b32


    set ga_secret_http_resp {<html>
  <body>
    <div align="center">
}

    if { $qr_code eq "yes" } {
      append ga_secret_http_resp "      <img src=\"$ga_qr_code_link\" />\n"
    }

    append ga_secret_http_resp "     <p>account: $account@$domain"
    append ga_secret_http_resp "key (secret): $secret_b32</p>\n      </div>\n
</body>\n</html>"

    HTTP::respond 200 content $ga_secret_http_resp
  } else {
    HTTP::respond 200 content {<html>
  <body>
    <h2><a  href="http://goo.gl/edmb2">Google  Authenticator</a>  key  (shared
secret) generator</h2>
    <form action="/ga_secret_generator" method="GET">
      <table cellspacing="1" cellpadding="1" border="0">
        <tr>
          <th align="left">account:</th>
```

```
          <td><input   name="account"   type="text"   size="10">   @   <input
name="domain" type="text" size="20"></td>
        </tr>
        <tr>
          <th align="left">secret:</th>
          <td><input name="secret" type="text" size="10"> *optional 10 charac-
ter key (additional chars truncated), random secret used if blank</td>
        </tr>
        <tr>
          <th align="left">generate QR code?</th>
          <td><input  name="qr_code"  type="checkbox"  value="yes">  *a  request
will be made to Google to generate QR code</td>
        </tr>
      </table>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>}
  }
}
```

OTP - iRule

```
when ACCESS_POLICY_AGENT_EVENT {
   if { [ACCESS::policy agent_id] eq "otp_verify_email" } {
            binary scan [b64encode [CRYPTO::keygen -alg random -len 128]] i
otp
            set otp [string range $otp 0 3]
            ACCESS::session data set session.user.otp $otp
            set mobile [ACCESS::session data get "ses-
sion.ad.last.attr.mobile"]
            set mail [ACCESS::session data get "session.ad.last.attr.mail"]
            set logstring "mail,[ACCESS::session data get "ses-
sion.ad.last.attr.mail"],otp,$otp,mobile,$mobile"
            log local0.alert "Event [ACCESS::policy agent_id] $logstring"
   }
}
```

Email-skripti

```
#!/bin/bash

tail -n0 -f /var/log/ltm | while read line
do
    var2=`echo $line | grep otp | awk -F'[,]' '{ print $2 }'`
    var3=`echo $line | grep otp | awk -F'[,]' '{ print $3 }'`
    var4=`echo $line | grep otp | awk -F'[,]' '{ print $4 }'`
    if [ "$var3" = "otp" -a -n "$var4" ]; then
       echo Sending pin $var4 to $var2
       echo One Time Password is $var4 | mail -s $var4 $var2
    fi
done
```

TUPAS-iRule

```
when HTTP_REQUEST {
     switch -glob [HTTP::uri] {
          /nordea/ {
               log local0. "URI: [HTTP::uri]"
               log local0. "Nordea TUPAS"
               #Tupas-formin muuttujat

               #Kentta 5, palveluntarjoajan tunnistuspyynnˆlle antama yksi-
loiva tunnus
               set aika [clock seconds]
               set tunnus [clock format $aika -format {%Y%m%d%H%M%S} ]

               #Kentta 12, laskettu MAC-tarkiste
               set mac "[class lookup sanomatyyppi nordea]&[class lookup ver-
sio nordea]&[class lookup asiakastunnus nordea]&[class lookup kieli
nordea]&$tunnus&[class lookup yksilotieto nordea]&[class lookup jatko
nordea]&[class lookup peruutus nordea]&[class lookup virhe nordea]&[class
lookup avainversio nordea]&[class lookup alg nordea]&[class lookup tarkiste
nordea]&"
               binary scan [sha256 $mac] H* sha_mac
               set sha_mac [string toupper $sha_mac]


               #Formin luominen html-sivuun
               HTTP::respond 200 content "
                    <html>

                    <head>
                    <title>Testi</title>
                    </head>

                    <body>
                    Tassa on form

                    <FORM METHOD=\"POST\" ACTION=\"[class lookup osoite
nordea]\">
                    <INPUT NAME=\"A01Y_ACTION_ID\" TYPE=\"hidden\" VAL-
UE=\"[class lookup sanomatyyppi nordea]\">
                    <INPUT NAME=\"A01Y_VERS\"  TYPE=\"hidden\" VALUE=\"[class
lookup versio nordea]\">
                    <INPUT NAME=\"A01Y_RCVID\" TYPE=\"hidden\" VALUE=\"[class
lookup asiakastunnus nordea]\">
                    <INPUT NAME=\"A01Y_LANGCODE\" TYPE=\"hidden\" VAL-
UE=\"[class lookup kieli nordea]\">
                    <INPUT NAME=\"A01Y_STAMP\" TYPE=\"hidden\" VAL-
UE=\"$tunnus\">
                    <INPUT NAME=\"A01Y_IDTYPE\" TYPE=\"hidden\" VALUE=\"[class
lookup yksilotieto nordea]\">
                    <INPUT NAME=\"A01Y_RETLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup jatko nordea]\">
                    <INPUT NAME=\"A01Y_CANLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup peruutus nordea]\">
                    <INPUT NAME=\"A01Y_REJLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup virhe nordea]\">
                    <INPUT NAME=\"A01Y_KEYVERS\" TYPE=\"hidden\" VAL-
UE=\"[class lookup avainversio nordea]\">
                    <INPUT NAME=\"A01Y_ALG\" TYPE=\"hidden\" VALUE=\"[class
lookup alg nordea]\">
                    <INPUT NAME=\"A01Y_MAC\" TYPE=\"hidden\" VAL-
UE=\"$sha_mac\">
```

```
                    <input type=\"submit\" />
                    </FORM>

                    </body>

                    </html>
               "
          }
          /aktia/ {
                    log local0. "Aktia TUPAS"
                    log local0. "URI: [HTTP::uri]"
                    #Tupas-formin muuttujat

                    #Kentta 5, palveluntarjoajan tunnistuspyynnˆlle antama
yksiloiva tunnus
                   set aika [clock seconds]
                   set tunnus [clock format $aika -format {%Y%m%d%H%M%S} ]

                    #Kentta 12, laskettu MAC-tarkiste
                    set mac "[class lookup sanomatyyppi aktia]&[class lookup
versio aktia]&[class lookup asiakastunnus aktia]&[class lookup kieli ak-
tia]&$tunnus&[class lookup yksilotieto aktia]&[class lookup jatko ak-
tia]&[class lookup peruutus aktia]&[class lookup virhe aktia]&[class lookup
avainversio aktia]&[class lookup alg aktia]&[class lookup tarkiste aktia]&"
                    binary scan [md5 $mac] H* sha_mac
                    set sha_mac [string toupper $sha_mac]


                    #Formin luominen html-sivuun
                    HTTP::respond 200 content "
                         <html>

                         <head>
                         <title>Testi</title>
                         </head>

                         <body>
                         Tassa on form

                    <FORM METHOD=\"POST\" ACTION=\"[class lookup osoite ak-
tia]\">
                    <INPUT NAME=\"A01Y_ACTION_ID\" TYPE=\"hidden\" VAL-
UE=\"[class lookup sanomatyyppi aktia]\">
                    <INPUT NAME=\"A01Y_VERS\"  TYPE=\"hidden\" VALUE=\"[class
lookup versio aktia]\">
                    <INPUT NAME=\"A01Y_RCVID\" TYPE=\"hidden\" VALUE=\"[class
lookup asiakastunnus aktia]\">
                    <INPUT NAME=\"A01Y_LANGCODE\" TYPE=\"hidden\" VAL-
UE=\"[class lookup kieli aktia]\">
                    <INPUT NAME=\"A01Y_STAMP\" TYPE=\"hidden\" VAL-
UE=\"$tunnus\">
                    <INPUT NAME=\"A01Y_IDTYPE\" TYPE=\"hidden\" VALUE=\"[class
lookup yksilotieto aktia]\">
                    <INPUT NAME=\"A01Y_RETLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup jatko aktia]\">
                    <INPUT NAME=\"A01Y_CANLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup peruutus aktia]\">
                    <INPUT NAME=\"A01Y_REJLINK\" TYPE=\"hidden\" VAL-
UE=\"[class lookup virhe aktia]\">
                    <INPUT NAME=\"A01Y_KEYVERS\" TYPE=\"hidden\" VAL-
UE=\"[class lookup avainversio aktia]\">
                    <INPUT NAME=\"A01Y_ALG\" TYPE=\"hidden\" VALUE=\"[class
lookup alg aktia]\">
                    <INPUT NAME=\"A01Y_MAC\" TYPE=\"hidden\" VAL-
UE=\"$sha_mac\">
```

```
                    <input type=\"submit\" />
                    </FORM>


                         </body>

                         </html>
                    "
          }
          /danskebank/ {
                    log local0. "Dankse Bank TUPAS"
                    log local0. "URI: [HTTP::uri]"
                    #Tupas-formin muuttujat

                    #Pankin tupas-palvelun osoite
                    set osoite
"https://verkkopankki.danskebank.fi/SP/tupaha/TupahaApp"

                    #Kentta 1, sanomatyyppi, vakioarvo 701
                    set sanomatyyppi "701"

                    #Kentta 2, pankkikohtainen versionumero
                    set versio "0003"

                    #Kentta 3, palveluntarjoajan pankkikohtainen asiakastunnus
                    set asiakastunnus "000000000000"

                    #Kentta 4, palvelun kieli
                    set kieli "FI"

                    #Kentta 5, palveluntarjoajan tunnistuspyynnˆlle antama
yksiloiva tunnus
                   set aika [clock seconds]
                   set tunnus [clock format $aika -format {%Y%m%d%H%M%S} ]


                    #Kentta 6, yksilotieto jonka palveluntarjoaja asiakkaas-
taan haluaa
                    set yksilotieto "02"

                    #Kentta 7, OK-tapauksen jatkokohta, vaatii SSL
                    set jatko "https://172.19.5.33/auth_ok/"

                    #Kentta 8, Autentikointi peruutetaan
                    set peruutus "https://172.19.5.33/canceled/"

                    #Kentta 9, tekninen virhe autentikoinnissa
                    set virhe "https://172.19.5.33/rejected/"

                    #Kentta 10, MAC-tarkisteen laskennassa kaytetyn avaimen
versio
                    set avainversio "0001"

                    #Kentta 11, MAC-tarkisteen algoritmi, 01=MD5 02=SHA-1
03=SHA-256
                    set alg "03"

                    #Kentta 12, laskettu MAC-tarkiste
                    set mac "$sanomatyyp-
pi&$versio&$asiakastunnus&$kieli&$tunnus&$yksilotieto&$jatko&$peruutus&$virhe&
$avainver-
sio&$alg&pAwfvWTD7g9etWGNTVR5zCj5EhHt5yuHdLrxQH2BD5gZxk7xBUrfqubtYv8vZvs4&"
                    binary scan [sha256 $mac] H* hex_mac
                    set hex_mac [string toupper $hex_mac]
```

```
                    #Formin luominen html-sivuun
                    HTTP::respond 200 content "
                         <html>

                         <head>
                         <title>Testi</title>
                         </head>

                         <body>
                         Tassa on form

                         <FORM METHOD=\"POST\" ACTION=\"$osoite\">
                         <INPUT NAME=\"A01Y_ACTION_ID\" TYPE=\"hidden\" VAL-
UE=\"$sanomatyyppi\">
                         <INPUT NAME=\"A01Y_VERS\"  TYPE=\"hidden\" VAL-
UE=\"$versio\">
                         <INPUT NAME=\"A01Y_RCVID\" TYPE=\"hidden\" VAL-
UE=\"$asiakastunnus\">
                         <INPUT NAME=\"A01Y_LANGCODE\" TYPE=\"hidden\" VAL-
UE=\"$kieli\">
                         <INPUT NAME=\"A01Y_STAMP\" TYPE=\"hidden\" VAL-
UE=\"$tunnus\">
                         <INPUT NAME=\"A01Y_IDTYPE\" TYPE=\"hidden\" VAL-
UE=\"$yksilotieto\">
                         <INPUT NAME=\"A01Y_RETLINK\" TYPE=\"hidden\" VAL-
UE=\"$jatko\">
                         <INPUT NAME=\"A01Y_CANLINK\" TYPE=\"hidden\" VAL-
UE=\"$peruutus\">
                         <INPUT NAME=\"A01Y_REJLINK\" TYPE=\"hidden\" VAL-
UE=\"$virhe\">
                         <INPUT NAME=\"A01Y_KEYVERS\" TYPE=\"hidden\" VAL-
UE=\"$avainversio\">
                         <INPUT NAME=\"A01Y_ALG\" TYPE=\"hidden\" VAL-
UE=\"$alg\">
                         <INPUT NAME=\"A01Y_MAC\" TYPE=\"hidden\" VAL-
UE=\"$hex_mac\">
                         <input type=\"submit\" />
                         </FORM>

                         </body>

                         </html>
                    "
          }

                     /osuuspankki/ {
                    log local0. "Osuuspankki TUPAS"
                    log local0. "URI: [HTTP::uri]"
                    #Tupas-formin muuttujat

                    #Pankin tupas-palvelun osoite
                    set osoite "https://kultaraha.op.fi/cgi-bin/krcgi"

                    #Kentta 1, sanomatyyppi, vakioarvo 701
                    set sanomatyyppi "701"

                    #Kentta 2, pankkikohtainen versionumero
                    set versio "0003"

                    #Kentta 3, palveluntarjoajan pankkikohtainen asiakastunnus
                    set asiakastunnus "Esittelymyyja"

                    #Kentta 4, palvelun kieli
                    set kieli "FI"
```

```
                    #Kentta 5, palveluntarjoajan tunnistuspyynnˆlle antama
yksiloiva tunnus
                   set aika [clock seconds]
                   set tunnus [clock format $aika -format {%Y%m%d%H%M%S} ]


                    #Kentta 6, yksilotieto jonka palveluntarjoaja asiakkaas-
taan haluaa
                    set yksilotieto "02"

                    #Kentta 7, OK-tapauksen jatkokohta, vaatii SSL
                    set jatko "https://172.19.5.33/auth_ok/"

                    #Kentta 8, Autentikointi peruutetaan
                    set peruutus "https://172.19.5.33/canceled/"

                    #Kentta 9, tekninen virhe autentikoinnissa
                    set virhe "https://172.19.5.33/rejected/"

                    #Kentta 10, MAC-tarkisteen laskennassa kaytetyn avaimen
versio
                    set avainversio "0001"

                    #Kentta 11, MAC-tarkisteen algoritmi, 01=MD5 02=SHA-1
03=SHA-256
                    set alg "01"

                    #Kentta 12, laskettu MAC-tarkiste
                    set mac "$sanomatyyp-
pi&$versio&$asiakastunnus&$kieli&$tunnus&$yksilotieto&$jatko&$peruutus&$virhe&
$avainversio&$alg&Esittelykauppiaansalainentunnus&"
                   binary scan [md5 $mac] H* hex_mac
                   set hex_mac [string toupper $hex_mac]


                    #Formin luominen html-sivuun
                    HTTP::respond 200 content "
                         <html>

                         <head>
                         <title>Testi</title>
                         </head>

                         <body>
                         Tassa on form

                         <FORM METHOD=\"POST\" ACTION=\"$osoite\">
                         <INPUT NAME=\"A01Y_ACTION_ID\" TYPE=\"hidden\" VAL-
UE=\"$sanomatyyppi\">
                         <INPUT NAME=\"A01Y_VERS\"  TYPE=\"hidden\" VAL-
UE=\"$versio\">
                         <INPUT NAME=\"A01Y_RCVID\" TYPE=\"hidden\" VAL-
UE=\"$asiakastunnus\">
                         <INPUT NAME=\"A01Y_LANGCODE\" TYPE=\"hidden\" VAL-
UE=\"$kieli\">
                         <INPUT NAME=\"A01Y_STAMP\" TYPE=\"hidden\" VAL-
UE=\"$tunnus\">
                         <INPUT NAME=\"A01Y_IDTYPE\" TYPE=\"hidden\" VAL-
UE=\"$yksilotieto\">
                         <INPUT NAME=\"A01Y_RETLINK\" TYPE=\"hidden\" VAL-
UE=\"$jatko\">
                         <INPUT NAME=\"A01Y_CANLINK\" TYPE=\"hidden\" VAL-
UE=\"$peruutus\">
```

```
                        <INPUT NAME=\"A01Y_REJLINK\" TYPE=\"hidden\" VAL-
UE=\"$virhe\">
                        <INPUT NAME=\"A01Y_KEYVERS\" TYPE=\"hidden\" VAL-
UE=\"$avainversio\">
                        <INPUT NAME=\"A01Y_ALG\" TYPE=\"hidden\" VAL-
UE=\"$alg\">
                        <INPUT NAME=\"A01Y_MAC\" TYPE=\"hidden\" VAL-
UE=\"$hex_mac\">
                        <input type=\"submit\" />
                        </FORM>

                        </body>

                        </html>
                    "
          }

          /canceled/ {
                   log local0. "Tunnistautuminen keskeytettiin"
                   log local0. "URI: [HTTP::uri]"
          }
          /rejected/ {
                   log local0. "Tunnistautumisessa virhe"
                   log local0. "URI: [HTTP::uri]"
          }
          /auth_ok/* {
            log local0. "URI: [HTTP::uri]"

            log local0. "----------------------"
            log local0. "URI Information"
            log local0. "----------------------"
            log local0. "HTTP::uri: [HTTP::uri]"
            log local0. "----------------------"
            log local0. "Path Information"
            log local0. "----------------------"
            log local0. "HTTP::path: [HTTP::path]"
            set depth [URI::path [HTTP::uri] depth]

           for {set i 1} {$i <= $depth} {incr i} {
              set dir [URI::path [HTTP::uri] $i $i]
              log local0. "dir\[$i\]: $dir"
            }

            log local0. "Basename: [URI::basename [HTTP::uri]]"
            log local0. "----------------------"
            log local0. "Query Information"
            log local0. "----------------------"
            log local0. "HTTP::query: [HTTP::query]"

            set namevals [split [HTTP::query] "&"]
            for {set i 0} {$i < [llength $namevals]} {incr i} {
              set params [split [lindex $namevals $i] "="]
              set pnum [expr $i+1]
              log local0. "Param\[$pnum\]: [lindex $params 0]"
              log local0. "Value\[$pnum\]: [URI::query [HTTP::uri] [lindex
$params 0]]"

              set value$pnum [URI::query [HTTP::uri] [lindex $params 0]]
            }

             set value5 [string map {+ " "} $value5]

             #set tarkiste "$val-
ue1&$value2&$value3&$value4&$value5&$value6&$value7&$value8&$value9&LEHTI&"
             #log local0. "Muuttamaton tarkiste $tarkiste"
```

```
            #binary scan [sha256 $tarkiste] H* sha_tarkiste
            #set sha_tarkiste [string toupper $sha_tarkiste]
            #log local0. "TARKISTE: $sha_tarkiste"
        }
        /favicon.ico {
             #log local0. "Favicon.ico requested"
        }
        default {
             log local0. "Default"
             log local0. "URI: [HTTP::uri]"
             if {[HTTP::uri] eq "/"}{
                  HTTP::respond 200 content [ifile get pankin_valinta]
               }
             elseif {[HTTP::uri] eq "/nordea_logo.gif"}{
                  HTTP::respond 200 content [ifile get nordea]
             }
             elseif {[HTTP::uri] eq "/db_logo.gif"}{
                  HTTP::respond 200 content [ifile get danske_bank]
             }
             elseif {[HTTP::uri] eq "/op-logo.gif"}{
                  HTTP::respond 200 content [ifile get op]
             }

        }
    }
}
```