Testausympäristön luonnin automatisointi AWS-pilvipalveluun

Mikko Korpi

Haaga-Helia ammattikorkeakoulu

Tradenomi

Opinnäytetyö

2023

Tiivistelmä

Tekijä(t)
Mikko Korpi

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Testausympäristön luonnin automatisointi AWS-pilvipalveluun

Sivu- ja liitesivumäärä
36 + 0

Tämä opinnäytetyö tehtiin toimeksiantona CGI Suomi Oy:n kyberturvakeskukselle. Opinnäytetyön lopputuotoksena oli toimiva testausympäristö, joka koostuu kahdesta palvelimesta. Toinen palvelimista toimii testauksen kohteena toimivana palvelimena, kun taas toisella voidaan suorittaa kyberhyökkäystekniikoita. Testausympäristö tullaan toteuttamaan automatisoidusti käyttäen Terraform-automatisaatiokehystä AWS-pilvipalvelussa (eng. Amazon Web Services). AWS on yksi maailman tunnetuimpia pilvipalvelutarjoajia.

Opinnäytetyössä otettiin myös kantaa kyberhyökkäyksen elinkaareen ja perusteltiin harjoitusympäristön tärkeyttä oppimisalustana kohderyhmälle. Kohderyhmänä toimii CGI Suomi Oy:n kyberturvakeskuksen työntekijät.

Toimeksiannossa käytettyjä työkaluja, kuten Ansible, Terraform ja eri skriptit ovat kuvattuina luvussa 4. Viidennessä luvussa käydään läpi toiminnallisen opinnäytetyön empiiristä vaihetta aina suunnittelusta itse tuotokseen. Lopullisen tuotoksen esittelyn jälkeen käydään läpi validointia. Validointi toteutettiin kohderyhmään kuuluvilla henkilöillä. Kuudennessa kappaleessa käydään läpi jatkokehitysideoita.

Toimeksianto oli onnistunut ja jatkokehittämiseen on helppo siirtyä. Jatkokehityskohteita on muun muassa kaiken kattavan lokienkeruujärjestelmän rakentaminen ja integrointi harjoitusympäristöön sekä testauksen kohteena olevien koneiden lisääminen ympäristöön.

Asiasanat
Tietoturva, automatisointi, harjoitusympäristö, Terraform, AWS, infrastruktuuri koodina

Sisällys

1 Johdanto

Tietoturva on kriittinen osa jokaisen yrityksen toimintaa. Yrityksien, jotka tuottavat tietoturvapalveluja muille yrityksille on pystyttävä ylläpitämään henkilöstönsä osaamistasoa korkeana ja kehittämään sitä jatkuvasti. Yksi tapa kehittää henkilöstön tietoturvaosaamista on testausympäristön kautta. testausympäristöllä tarkoitetaan kontrolloitua ympäristöä, missä voidaan evaluoida ja testata järjestelmien tai palveluiden tietoturvaa (Noël 2003, 4).

Tämän opinnäytetyön tavoite on saada täysin automatisoidusti luotua kahden virtuaalikoneen tietoturvatestaus-ympäristö AWS-pilvipalveluun sekä toimiva palvelininfrastruktuuri niiden välille. Työ tehdään toimeksiantona CGI Suomi Oy:n kyberturvakeskukselle.

CGI on yksi maailman suurimmista IT-yrityksistä Suomessa ja koko maailmassa. Suomessa työskentelee noin 3800 työntekijää ja aluetoimistoja on yli kymmenellä paikkakunnalla. Asiakkaina toimii muun muassa eri yritykset sekä julkisen sektorin organisaatiot. (CGI 2023a.)

Testausympäristössä ei ole mitään yrityksen tai kumppaneiden oikeaa tietoa eikä sen hajottamisesta saa koitua haittaa yrityksen toiminnalle. Tämänlainen ympäristö kuuluu olla eriytetty verkosta ja muista komponenteista mitkä toimivat osana yrityksen liiketoimintaa. Testauksissa palvelimet ja niiden käyttöjärjestelmät voivat mennä vikatilaan, joten ympäristön automatisointi on tärkeää. Myös kustannustehokkuus tulee pitää mielessä testausympäristön suunnitteluvaiheessa. Kustannuksia voi kertyä ympäristön tekijän käytetyistä työtunneista sekä sijainnista. Tässä kontekstissa sijainnilla tarkoitetaan sitä, onko palvelimet pilvipalvelussa vai On Premise –ratkaisuna. “On-premises” nimitys tulee siitä jos palvelinkapasiteetti sijaitsee fyysisesti esimerkiksi organisaation omissa tiloissa (Suse, 2023). Kustannustehokkuuden lisäksi tulee huomioida pilvipalvelun tuomat hyödyt, kuten skaalautuvuus ja ylläpidon helpottuminen.

Ympäristön automatisoinnin toteutukselle on useita eri tapoja ja työkaluja. Oikeiden työkalujen ja prosessien tunnistaminen on tärkeä osa suunnittelua. Lopputuotoksen ei pitäisi sisältää juurikaan manuaalisia vaiheita. Tässä toimeksiannossa tullaan käyttämään pääasiassa Terraform-ohjelmistotyökalua, joka soveltuu infrastruktuuri koodina - ympäristön rakentamiseen. Työ rajattiin pienimpään mahdolliseen ympäristön rakentamiseen mikä on järkevää. Tarkoituksena on luoda toimiva todiste infrastruktuuri koodina – konseptista ja miten sitä voidaan hyödyntää kohderyhmän tietoturvataitojen kehittämisessä.

Tässä opinnäytetyössä käytetään lineaarista kehittämismenetelmää jonka vaiheet etenevät rationaalisesti. Lineaarinen malli on kaikkein osuvin kehityksen kannalta sillä päätyökaluksi valittu

Terraform käyttäytyy melko samalla tavoin. Kohderyhmänä toimii CGI Suomen kyberturvakeskuksen työntekijät. Vastuullisuutta silmällä pitäen ympäristö rakennetaan hyviä tietoturvakäytäntöjä seuraten, kuten pelkästään tarvittavien yhteyksien avaaminen, jotta ympäristöön pääsee käsiksi.

2 Kyberturvallisuus ja kyberhyökkäykset

Tässä luvussa käydään läpi, mitä tietoturva on ja miten yksilö voi kehittää tietoturvataitojaan. Tämän lisäksi otetaan katsaus kyberhyökkäyksen elinkaareen, minkä avulla voidaan ymmärtää kyberhyökkäyksiä.

2.1 Tietoturvataitojen kehittäminen

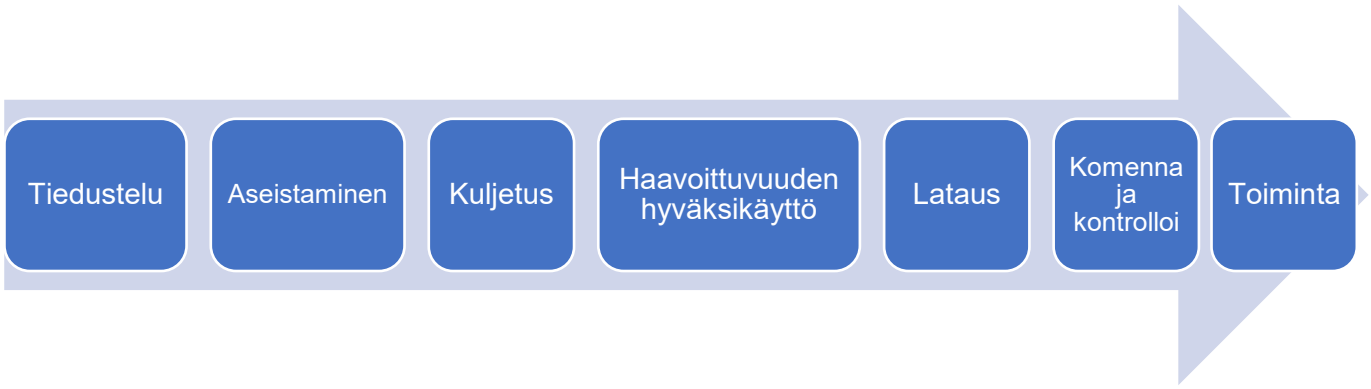
Kyberturvallisuutta on aiemmin pidetty enemmänkin IT-haasteena kuin liiketoimintariskinä. Kyberturvallisuus on osa johtamista, jonka tehtävänä on minimoida organisaation kyberavaruuteen kohdistuvat riskit ja estää mahdolliset tietoturvapoikkeamat. Kyberturvallisuustiedon merkitys tunnustetaan nykyään laajalti, mutta sen soveltamisen tarve riippuu suoraan työvoiman kyberturvallisuuden osaamisesta. Tietoturvataidot voidaan sitoa yhteen yksilön kykyjen, tiedon ja kokemuksen yhdistelmällä ja tarkastella mikä mahdollistaa yksilön suorittamaan tehtävän onnistuneesti. Tietoturva-alalle haluavien ja siellä jo olevien henkilöiden taitovaatimukset muuttuvat tavanomaista nopeammin uusien teknologioiden ja yhteyskunnan nopean digitalisoitumisen myötä. (Blažič 2019, 1-10.)

Tietämystä ja taitoa voi myös hankkia omien harrasteprojektien kautta. Tehokas metodi oppia on erinäisten harjoitusympäristöjen avulla. Nämä harjoitusympäristöt tarjoavat harjoittelumahdollisuuksia tunkeutumistestaajille tai eettisestä hakkeroinnista kiinnostuneille henkilöille. Tästä huolimatta sellaisen puolustavan puolen harjoitusalustoja ei ole, missä simuloitua kyberhyökkäystä voitaisiin tutkia käyttäen järjestelmien lokeja tai lokienkäsittelyjärjestelmiä. Myöskään kustomoidut harjoitukset eivät onnistu valmiissa palveluissa. Oman harjoitusympäristön luominen vaatii monien asioiden huomioimista. Jotkut koulutusohjelmat käyttävät erilaisia työkaluja kulissien takana helpottaakseen asennustehtäviä, mutta tällaisia työkaluja ei julkisteta, joten ne eivät juurikaan hyödytä suurta yleisöä (Beuran, Pham, Tang, Chinen, Tan & Shinoda 2017,157).

Tällä hetkellä tunnetut käsitteet, joita käytetään kyberturvallisuuden rooliprofiilin määritelmissä ovat muun muassa kompetenssi, taidot ja tietämys. Kompetenssilla tarkoitetaan tietoa havaittavien tulosten saavuttamiseksi. Taidot voidaan määritellä kyvyksi käyttää tietotaitoa ja asiantuntemusta tehtävien suorittamiseen ja ongelmanratkaisuun. Tietämys määritellään tiettyyn opiskelu- tai työalaan liittyvien tosiasioiden, periaatteiden, teorioiden ja käytäntöjen kokonaisuudeksi. Blažičin teettämän tutkimuksen mukaan vaadittavat tietoturvataidot eivät ole yhtenäisiä, sillä eri aloilla vaaditaan eri erikoisosaamisalueiden hallitsemista. Yhteinen piirre tutkimuksessa oli, että pätevyyttä ja taitoja voidaan saavuttaa menestyksekkäämmin harjoitusympäristöjen avulla. (Blažič 2019, 4-8.)

2.2 Kybertappoketju

Amerikkalaisen Lockheed Martin yrityksen vuonna 2011 tekemästä sotilaallisesta mallista peräisin oleva kybertappoketju on vaiheittainen lähestymistapa kyberhyökkäyksien ymmärtämiseen. Sen tavoitteena on tunnistaa pysäyttää haitallisten toimijoiden toimet. Kybertappoketjua voidaan myös kutsua kyberhyökkäyksen elinkaareksi. Se koostuu seitsemän vaiheen ketjusta ja se on integroitu alusta loppuun kulkeva prosessi. Sitä kutsutaan ketjuksi, sillä mikä tahansa kohta, mikä ei tapahtu keskeyttää koko prosessin. (Hutchins, Cloppert & Amin s.a., 4-5.)



Kuva 1. Kybertappoketju (mukaillen Hutchins ym. s.a., 4-5)

Kuten kuvasta 1 havaitaan, kybertappoketjun ensimmäinen vaihe on tiedusteluvaihe. Siinä identifioidaan, valitaan ja profiloidaan kohde. Tiedustelulla kerätään informaatioita potentiaalisesta kohteesta. Kohde voi olla organisaatio tai yksittäinen henkilö. Tiedustelun voi jakaa kahteen eri osaan, jotka ovat passiivinen ja aktiivinen tiedustelu. Passiivisessa tiedustelussa kerätään tietoa ilman, että kohde tietää sitä. Tällainen toiminta voi olla vaikkapa sosiaalisen median käyttöä tiedonkeruussa. Aktiivinen tiedustelu on yleensä syvempää profilointia. Kybertilassa se voisi tarkoittaa esimerkiksi organisaation palvelimien porttiskannausta, minkä avulla selvitetään palvelimen avonaisia portteja. (Yadav & Mallari 2015, 2.)

Aktiivista tiedustelua voidaan toteuttaa verkko- ja haavoittuvuusskannereilla. Postonin mukaan Nmap-ohjelmistolla pystytään skannaamaan kohdeverkkoja selvittäen niiden käyttöjärjestelmät, sekä mitä palveluita siellä on käytössä. Haavoittuvuusskannerilla kuten Nikto:lla voidaan skannata web-palvelin yleisten haavoittuvuuksien varalta. Monet aktiivisen tiedustelun työkalut ovat äänekkäitä ja jättävät jäljen siitä, että niitä on käytetty. (Poston, H. 2019.)

Aseistamisvaiheessa hyödynnetään tiedustelun pohjalta saatuja tietoja ja luodaan hyötykuorma, jota käytetään kyberaseena. Teknisesti ottaen aseistaminen voi olla tiedustelussa löydetyn

haavoittuvuuden käyttöä varten valmistautumista, eli haittaohjelman luontia. Haittaohjelma voidaan luoda yhdistämällä etäkäyttöä mahdollistava haitallinen koodi olemassa olevaan haavoittuvuuteen. Esimerkkinä aseistamisesta olisi tämän haitallisen koodin syöttäminen Microsoft Office – dokumenttiin, mikä suorittaa koodin, kun käyttäjä avaa sen. (Hutchins, Cloppert & Amin s.a., 4; Yadav & Mallari 2015, 2-3.)

Kuljetusvaiheessa siirretään haittaohjelma kohteen ympäristöön. Tämä vaihe on todella kriittinen osa kybertappoketjua ja on suuressa vastuussa kyberhyökkäyksen onnistumisesta. Kuljetusmetodeja on monia ja jos haittaohjelma tai haavoittuvuus vaatii käyttäjän toimenpiteitä, pitää myös haittaohjelman tai kuljetustavan olla tarpeeksi kiinnostava kohde käyttäjälle laukaistakseen haitallisen ohjelman. Tiedusteluvaiheessa kerätty tieto tulee olemaan tärkeässä roolissa kuljetusvaiheessa, sillä se edesauttaa haittaohjelman onnistunutta kuljetusta. Mikään yksittäinen kuljetustapa ei ole täysin varma toimiakseen aina. Yleensä kyberhyökkäyksissä käytetään monia eri kuljetustapoja varmistamaan onnistunut kuljetus. Yksi tunnetuimmista kuljetusmetodeista kalasteluhyökkäys. Kalasteluhyökkäyksessä käytetään sosiaalista manipulointia, missä käyttäjälle voidaan lähettää sähköpostiviesti, jonka avulla yritetään saada käyttäjältä tietoja. Hyökkääjä voi maskeerata itsensä näyttämään luotettavalta entiteetiltä saadakseen kohdekäyttäjän tunnukset haltuunsa. (Pawankumar, Dash & Ansari 2022, 3; Yadav & Mallari 2015, 3-4.)

Kybertappoketjun neljännessä vaiheessa hyväksikäytetään haavoittuvuutta. Tämän voi laukaista esimerkiksi hyvätahtoinen käyttäjä, joka on manipuloitu avaamaan kiinnostavan näköinen tiedosto, minkä hän sai sähköpostiinsa hyökkääjältä. Tässä vaiheessa on yleensä tärkeää, että haittaohjelma suorittaa näkymättömästi haitalliset toimensa. (Yadav & Mallari 2015, 3-4.)

Latausvaiheessa haittaohjelma pyrkii saastuttamaan kohdejärjestelmän. Nykypäiväiset haittaohjelmat ovat tarpeeksi edistyneitä huomaamaan automaattisia suojauksia ja kykenevät jopa piiloutumaan niiltä. Muutamia näistä tekniikoista ovat esimerkiksi suojaustuotteiden sammuttaminen ja emulaatioympäristöjen tunnistaminen. Emulaatioympäristöllä tarkoitetaan ympäristöä, joka on tarkoitettu esittämään oikeaa ympäristöä, missä tietoturva-asiantuntija voisi tarkemmin tutkia haittaohjelmaa. Jos haittaohjelma tunnistaa emulaatioympäristön, se voi esimerkiksi maskeerata itsensä tavalliseksi ohjelmaksi. (Yadav & Mallari 2015, 4-5.)

Kuudennetta vaihdetta kutsutaan nimellä komenna ja kontrolloi. Tässä vaiheessa hyökkääjä on pystyttänyt hallinta- ja kommunikaatiokanavat kohteen verkkoon. Hyökkääjä voi seuraavaksi liikkua verkossa laitteiden välillä tai suorittaa tietojen varastamista. Laitteista toiseen liikkumista verkon sisällä kutsutaan lateraaliksi liikkumiseksi tai sivuttaissiirtymäksi. Sen tavoitteena on levittäytyä

koko kohde verkkoon tai päästä ennalta määrätyn kohteen luokse. (SentinelOne 2023b; Zeng & Germanos 2019, 5.)

Viimeisessä vaiheessa tehdään ennalta määritellyt toimenpiteet kohteessa. Tämä riippuu tavoitteesta, mihin hyökkääjät alun perin pyrkivät, kun hyökkäystä suunniteltiin. Tyypillisesti tämä tavoite on arkaluontoisen tiedon kerääminen, salaaminen ja kuljettaminen ulos järjestelmästä. Vaihtoehtoisesti tunkeutujat voivat käyttää hakkeroitua konetta hyppypisteenä muihin verkossa oleviin koneisiin. (Hutchins, Cloppert & Amin s.a, 5 ; Zeng & Germanos 2019, 5.)

3 Harjoituksen ympäristö

Tässä luvussa käydään läpi harjoitusympäristön sijainti, eli minne harjoitusympäristö rakennetaan. Myös käyttäjätukipolitiikkaa tarkastellaan tässä luvussa koska sillä on suuri merkitys varsinaisen harjoitusympäristön toiminnallisuuteen.

3.1 Amazon Web Services

Amazon Web Services -pilvipalvelu on yksi maailman kattavimmista ja tunnetuimmista pilvipalveluista. Palvelu koostuu sekoituksesta IaaS, PaaS ja SaaS-palveluista. AWS-palvelut voivat tarjota organisatorisia työkaluja kuten laskentatehoa, tietokantatallennusta ja sisällön toimituspalveluita. (Barney 2022.) IaaS, PaaS ja SaaS tarkoittavat eri palvelumalleja. Palvelumallit vaikuttavat myös eri tavoin palvelunostajan vastuisiin. IaaS tarkoittaa infrastruktuuria palveluna, SaaS tarkoittaa ohjelmistoa palveluna ja PaaS tarkoittaa alustaa palveluna.

SaaS voidaan määritellä palveluntoimitusmalliksi, jossa annettaan käyttäjälle applikaatio käyttöönsä. Yleensä SaaS-palveluita käytetään selaimen kautta. Käyttäjällä ei ole pääsyä applikaation arkkitehtuuriin kuten tietoverkkoihin, palvelimiin ja varastoihin. Palveluntarjoaja vastaa kokonaisvaltaisesti applikaatiosta. IaaS-palvelumalli auttaa palvelunostajaa modernisoimaan ja laajentamaan infrastruktuuriaan. IaaS-mallia voidaan pitää pilvilaskentapalvelumallien perustana, sillä se tarjoaa muun muassa virtuaalisen palvelintilan, verkkoyhteydet, IP-osoitteet ja kuormantasauksen. Palvelunostajat voivat rakentaa oman ympäristönsä ja varsinaisen alla olevan infrastruktuuripalvelun ylläpidon hoitaa pilvipalveluntarjoaja. Pilvipalveluntarjoajan vastuu ulottuu siis alustoihin, joita käytetään infrastruktuuripalvelun tuottamiseen. PaaS-mallissa yleisimmät käyttäjät ovat ohjelmistokehitysyritykset. Ohjelmistojen tuottajat tarvitsevat alustan, jossa ohjelmistoa voidaan ajaa. PaaS-malli mahdollistaa palvelunostajan ikään kuin vuokraamaan muun muassa käyttöjärjestelmät, laitteistot, tallennustilan ja verkkokapasiteetin. Palvelunkäyttäjän omalle vastuulle jää vain oman sovelluksensa tietoturva. (Freet, Agrawal, John & Walker 2015, 150-153.)

Narulan, Jainin ja Prachi:n (2015, 503) mukaan AWS-pilvipalvelu on täydellinen esimerkki pilvipalvelusta, joka ei ainoastaan tarjoa erinomaisia pilvipalveluita, vaan tarjoaa myös luottamuksellisuuden, eheyden ja asiakkaan datan saatavuuden. IT-resurssit ovat saatavilla halvalla hinnalla ja etukäteen ei tarvitse maksaa mistään. Käyttäjä maksaa pelkästään resursseista mitä hän tarvitsee. Jos resurssien määrä on arvioitu väärin, skaalautuvuus kumpaankin suuntaan onnistuu helposti.

Tässä käyttötapauksessa tullaan käyttämään seuraavia AWS-pilviympäristön palveluita:

- EC2-instanssit

- EC2-instanssien käyttäjädata
- Virtuaalinen yksityinen pilvi
- Turvaryhmät
- Internet-yhdyskäytävä
- Aliverkko
- Kustomoitu reititystaulu

EC2-instanssit ovat Amazon-pilvipalvelun virtuaalisia pilvilaskentaympäristöjä. Tuttavallisemmin niitä kutsutaan virtuaalikoneiksi. Pilvilaskenta on IT-resurssien toimitusta tarpeen vaatiessa internetin kautta. (AWS s.a.c.) Tämä eroaa perinteisestä yrityksen On Premise – ratkaisusta huomattavasti, sillä yrityksen ei tarvitse haaskata aikaa ja rahaa ylläpitääkseen fyysistä datakeskusta.

EC2-instanssien käyttäjädatalla tarkoitetaan dataa, mitä suoritetaan, kun instanssi käynnistyy. Tämän ominaisuuden avulla pystytään suorittamaan esimerkiksi komentoja tai skriptejä, joiden avulla voidaan konfiguroida instanssin sisällä olevia palveluita automaattisesti. (AWS s.a.e.) Tämä ominaisuus on todella kätevä automatisoinnin kannalta ja sitä voidaan hyödyntää monien eri käyttöjärjestelmien kanssa.

Virtuaalista yksityistä pilveä (eng. Virtual Private Cloud) kutsutaan tuttavallisemmin lyhenteellä VPC. Se mahdollistaa AWS-resurssien käytön määrittämääsi virtuaaliverkkoon. Virtuaaliverkko muistuttaa perinteistä tietoverkkoa, jota käyttäisit yrityksen omassa datakeskuksessa mutta siinä on AWS:n skaalautuvan infrastruktuurin käyttöetuja. Sen ominaisuuksina on mm. turvaryhmät, internet-yhdyskäytävät, aliverkot ja reititystaulut. (AWS s.a.f.)

Turvaryhmä (eng. security group) toimii virtuaalisena palomuurina EC2-instansseille, joilla voidaan kontrolloida sisään tulevaa ja ulos menevää liikennettä. Niitä voi spesifioida useampia kuin yhden. Lisäksi voidaan asettaa erinäisiä sääntöjä turvaryhmään. Turvaryhmät mahdollistavat liikenteen kulun juuri niistä porteista tietyillä protokollilla, mitä sille asetetaan. Turvaryhmissä on tärkeää huomioida niiden tilallisuus. Jos sisään tuleva pyyntö päästetään läpi määrityksissä, myös ulospäin menevä vastaus pääsee läpi. Näitä turvaryhmiä voidaan määritellä useampia kuin yksi EC2-instanssia kohden. (AWS s.a.g.; Checkpoint, s.a.)

AWS:n internet yhdyskäytävä on VPC:n komponentti, mikä mahdollistaa kommunikaation instanssien välillä virtuaalisessa yksityisessä pilvessä sekä internetin välillä. Kun instanssi lähettää liikennettä internetin suuntaan, liikenne ohjautuu internet yhdyskäytävään. Samoin käy myös, kun

internetistä tulee liikennettä kohti instanssia. Turvaominaisuuksia, kuten turvaryhmiä, voidaan käyttää internet-yhdyskäytävän turvaamiseen. (AWS s.a.i.)

Aliverkot (eng. subnets) ovat IP-osoitteiden alue virtuaalisessa yksityisessä pilvessä. Niiden avulla voidaan osioida VPC:n IP-avaruus pienempiin, helpommin hallittaviin osiin. Aliverkkoja voi olla julkisia tai yksityisiä. Julkisella aliverkolla on reitti internettiin, esimerkiksi juuri internet-yhdyskäytävän läpi. (AWS s.a.j.)

Yksityisillä aliverkoilla ei ole reittiä internettiin, mutta ne voivat kommunikoida internetin kanssa NAT-yhdyskäytävän läpi. NAT-yhdyskäytävä mahdollistaa yksityisessä aliverkossa olevien instanssien kommunikoinnin internettiin, mutta internetissä olevat ulkoiset palvelut eivät voi aloittaa kommunikointia instanssien kanssa. (AWS s.a.k.)

Virtuaalisen, yksityisen pilven reititystauluilla voidaan konfiguroida, kuinka liikenne reitittyy VPC:n sisällä. Jokaisessa VPC:ssä on oletusreititystaulu, jota sen kaikki aliverkot käyttävät, jos muuten ei ole määritelty. Reititystaulu sisältää listan sääntöjä, mitä kutsutaan reiteiksi. Reitit määrittelevät, miten liikenne reitittyy kohteensa mukaisesti. (AWS s.a.l.)

3.2 AWS-pilviympäristön käyttäjätukipolitiikka

AWS-pilviympäristössä tiettyjen tietoturvatestauksien tekeminen on sallittua. Omille AWS-infrastruktuureille voidaan suorittaa testauksia ilman erillistä lupaa. Testauksen ei tule vahingoittaa AWS:än omaa infrastruktuuria tai muiden asiakkaiden käyttämiä resursseja. Tästä syystä testaukseen kiellettyjä aktiviteettejä ovat muun muassa DNS-palveluihin kohdistuvat hyökkäykset, palvelunestohyökkäykset ja tulvahyökkäykset, jotka kohdistuvat portteihin, protokolliin tai kyselyihin. Palvelunestohyökkäyksiä saa simuloida, mutta ne ovat rajoitettuja ja vaativat toisen AWS:än politiikan noudattamista, sekä se täytyy hyväksyttää AWS:llä etukäteen. Hyväksyttyjä palveluita ja aktiviteetteja ovat muun muassa EC2-instansseihin kohdistuvat testaukset, NAT-yhdyskäytävät ja web-applikaatiopalomuurit. (AWS s.a.a.) Käyttäjätukipolitiikka mahdollistaa testausympäristön rakentamisen AWS-pilvipalveluympäristöön.

4 Työkalut

Tässä luvussa esitellään työkalut mitä ympäristön rakentamisessa ja konfiguroinnissa käytettiin. Eri työkalut soveltuvat parhaiten eri työn vaiheisiin.

4.1 Ansible

Ansible on avoimen lähdekoodin automaatiotyökalu. Sillä voidaan konfiguroida järjestelmiä ja orkestroida edistyneitä työnkulkuja, jotka tukevat esimerkiksi applikaatioiden käyttöönottoa. Ansible-automatisointityökalulla on arkkitehtuurillisesti kontrolli-noodi ja hallittavat noodit. Tämä konsepti tarkoittaa, että kontrolli-noodilla voidaan komentaa etänä muita hallittavia noodeja. Työkalu on suuressa käytössä yritysmaailmassa muun muassa palvelinten automatisoinnissa ja se käyttää OpenSSH-kirjastoa pakettien kuljetukseen. Jotta Ansible toimisi, se tarvitsee inventaarion ja tunnukset. Inventaario on konfiguraatiotiedosto, jossa on lista laitteista. Laitteet voivat olla palvelimia, verkkolaitteita tai mitä vain, mitä Ansiblella halutaan hallita. Ilman tätä listaa Ansiblella ei voida komentaa toisia laitteita. (Ansible s.a.a; EDUCBA s.a.a.) Ansible on imperatiivinen työkalu, jonka tarvitsee tietää kaikki tarvittavat työvaiheet päästääkseen maaliin. Ansiblea on järkevintä käyttää palvelinten konfiguraation hallinnassa.

SSH on kryptograafinen verkkoprotokolla, jota käytetään turvalliseen etäkirjautumiseen laitteille. Tyypillisiä käyttötarkoituksia SSH-protokollalle yritysmaailmassa on taata turvallinen pääsy etänä hallittaviin laitteisiin ja mahdollistaa automatisointiprosessien toimivuus. Sillä pystyy myös turvallisesti siirtämään tiedostoja laitteiden välillä. Tämä protokolla toimii käyttäjä-palvelin-mallissa mikä tarkoittaa, että yhteys muodostetaan käyttäjältä palvelimelle. SSH käyttäjä aloittaa yhteyden luomisprosessin ja käyttää julkiavain-kryptografiaa varmistaakseen kohdepalvelimen identiteetin. Tämän prosessin jälkeen SSH-protokolla käyttää vahvaa symmetristä salausta ja tiivistealgoritmejä varmistaakseen datan eheyden mitä käyttäjän ja palvelimen välillä vaihdetaan. (SSH 2023a.)

SSH-yhteydessä käytetään avainpohjaista autentikaatiomekanismia nimeltään julkiavainautentikaatio. Pohjimmiltaan joitain istuntokohtaisia tietoja allekirjoitetaan käyttämällä yksityistä tunnusavainta. Allekirjoitus lähetetään palvelimelle, joka tarkistaa onko allekirjoitukseen käytetty avain määritetty valtuutetuksi avaimeksi. Tämän jälkeen palvelin tarkistaa digitaalisen allekirjoituksen käyttämällä valtuutetun avaimen julkista avainta. Yksityistä avainta ei tulisi koskaan lähettää palvelimelle (SSH 2023b.)

4.2 Terraform

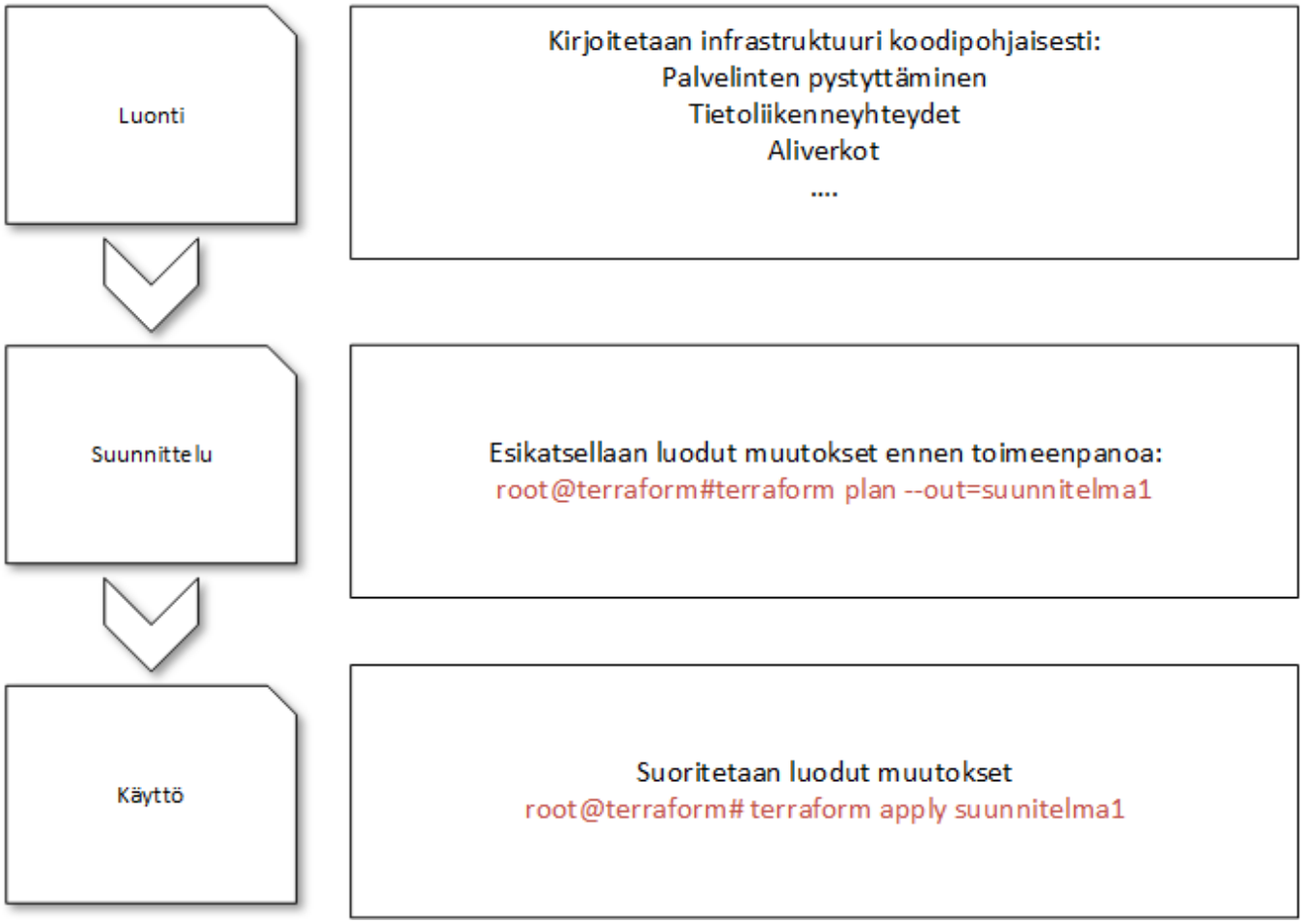
Terraform on avoimen lähdekoodin työkalu, jonka on luonut yritys nimeltään HashiCorp. Terraform mahdollistaa infrastruktuurin määrittelyn koodina käyttäen yksinkertaista ohjelmointikieltä. Sillä onnistuu myös infrastruktuurin pystytys ja hallinta monissa julkisissa pilvipalveluissa, kuten AWS:ässä. Infrastruktuuri koodina –käsite pitää sisällään koodin kirjoittamisen ja ajamisen infrastruktuurin määrittelemiseksi, käyttöönottamiseksi ja päivittämiseksi. DevOps-mallissa keskeinen näkemys on, että melkein kaikkea voi hallita koodipohjaisesti. Brikmanin mukaan on olemassa neljä ylätasolla kuvattua kategoriaa infrastruktuuri koodina –määritelmälle. Ne ovat ad-hoc-skriptit, konfiguraatiohallintatyökalut, palvelinten mallinnetyökalut ja palvelinten hallintatyökalut (Brikman 2017, johdanto; Brikman 2023, luku 1.) Terraform kuuluu näistä palvelintenhallintatyökaluihin.

Terraformin päätarkoitus on määritellä resursseja mitkä kuvaavat infrastruktuurin kohteita ja sen konfiguraatio on täydellinen kuvaus sen omalla kielellään, miten se hallinnoi annettua infrastruktuurikokoelmaa. Infrastruktuurin hallinta koodipohjaisesti korvaa IT-resurssien hallinnassa tarvittavat tavalliset toimintatavat ja manuaalisen työn koodilla. Se automatisoi infrastruktuurin hallinnan lähdekoodin kautta. Käyttäjän tulee määritellä, mitä halutaan luoda ja missä. Terraform hoitaa itse loput ja provisioi määritellyn infrastruktuurin. Tämä toimintatapa tekee Terraformista deklaratiivisen työkalun. Imperatiivisissa työkaluissa koko reitti valmiiseen tuotokseen pitää määritellä tietyssä järjestyksessä. (Bichard 2019; Terraform s.a.a.) Imperatiiviset työkalut voidaan mieltää myös proseduurisiksi niiden käyttäytymisen mukaan. Esimerkki proseduurisesta työkalusta mitä tässä projektissa käytettiin on Ansible, mistä kerrottiin kappaleessa 4.1. Niin kuin monessa muussakin asiassa, infrastruktuurin kirjoittaminen koodina helpottuu huomattavasti, jos käyttäjällä on oikeat työkalut oikeisiin työvaiheisiin tai käyttötapauksiin. Kumpikaan tapa ei ole toista huonompi, vaan kaikki liittyy siihen, mitä halutaan saavuttaa.

Tärkeä osa Terraformia ovat tarjoajat. Tarjoajat antavat pääsyn käyttää esimerkiksi pilvitarjoajia ja muita API-ratkaisuja. Jotkut tarjoajat vaativat käyttäjää konfiguroimaan asetuksia ennen kuin Terraform voi käyttää heidän resurssejaan. Terraformin konfiguraatio tarvitsee myös tiedon, mitä tarjoajia vaaditaan operaation suorittamiseen. (Terraform s.a.e.) Tarjoajalla tarkoitetaan esimerkiksi AWS-pilvipalvelua.

Työnkulku Terraformissa kiteytyy kolmeen kohtaan, jotka voidaan havaita kuvasta 2. Ne ovat koodin kirjoittaminen, suunnittelu ja käyttäminen. Kirjoittamisvaiheessa luodaan Terraform-konfiguraatiotiedostot käyttämällä Terraform:in konfiguraatiokieltä. Suunnitteluvaiheessa esikatsellaan

muutokset ennen niiden toimeenpanoa. Käyttämisvaiheessa kirjoitettu koodi, joka on esikatseltu ja todettu hyväksi, suoritetaan. (Terraform s.a.c.)



Kuva 2. Terraform:in työnkulku (Mukaillen Terraform 2023)

Terraform-kielen syntaksi, eli lauserakenne on suhteellisen helppoa ihmisille lukea ja kirjoittaa. Se koostuu kahden keskeisen syntaksirakenteen ympärille, joita ovat argumentit ja lohkot. Argumenteilla tarkoitetaan, kun jollekin nimelle annetaan arvo. Lohkolla tarkoitetaan konttia tai säilöä muulle. (Terraform s.a.b.)

```
resource "aws_instance" "my_disposable_linux_vm" {

  ami             = "<ami-id-tähän>"
  instance_type   = "t3.micro"


}
```

Kuva 3. Terraform – konfiguraation lohko

Kuvassa 3 näkyy Terraform – kielellä oleva lohko, jonka sisällä on kaksi argumenttia. Ennen yhtäsuuruusmerkkiä tulee argumentin nimi ja sen jälkeen arvo. Tämä lohko on tyypiltään resurssi lohko, millä voidaan määritellä infrastruktuuriresurssi. Jokainen lohkotyyppi määrittelee, kuinka monta etikettiä siinä pitää olla. Resurssilohkotyyppi vaatii kaksi etikettiä, jotka ovat ”aws_instance” sekä ”my_disposable_linux_vm” tässä esimerkissä. Niillä kerrotaan Terraformille, että kyseessä on AWS-pilvipalvelun resurssi mikä halutaan luoda nimellä ”my_disposable_linux_vm”. (Terraform s.a.b)

Terraformin moduulit ovat kontteja eri resursseille, joita käytetään yhdessä. Terraformin konfiguraatiotiedostot päättyvät .tf-tiedostopäätteeseen. Moduuli koostuu kokoelmasta konfiguraatiotiedostoja, joita pidetään yhdessä hakemistossa. Moduulit ovat tärkein tapa pakata resurssikokoonpanoja Terraformin kanssa. (Terraform s.a.d.)

4.3 Skriptit

Kuori (eng. shell) on ohjelma, joka tarjoaa rajapinnan käyttäjälle käyttää käyttöjärjestelmän palveluita. Kuori hyväksyy komentoja käyttäjiltä ja kääntää ne sopivaksi mitä käyttöjärjestelmän ydin ymmärtää. (GeeksforGeeks, s.a.)

Powershell on monialustainen tehtävien automatisointiin perustuva ratkaisu, joka koostuu komentorivikuoresta, skriptauskielestä ja konfiguraationhallintakehyksestä. Powershelliä voidaan käyttää Windows-, Linux- ja macOS-käyttöjärjestelmillä. Skriptauskielenä Powershell on yleisimmin käytetty automatisoimaan järjestelmien hallintaa. Sitä voidaan myös käyttää rakentamiseen, testaamiseen ja ratkaisujen pystyttämiseen. (Microsoft 2023.) Vaikka Powershell-ratkaisua voidaan ajaa Linux:illa se vaatii erillisen asennuksen eikä ole valmiiksi asennettuna. Tästä syystä useat henkilöt käyttävät käyttöjärjestelmien omia komentokieliä automatisoidakseen tehtäviä. Kuvassa neljä havainnollistetaan powershell-skriptin toimintaa.

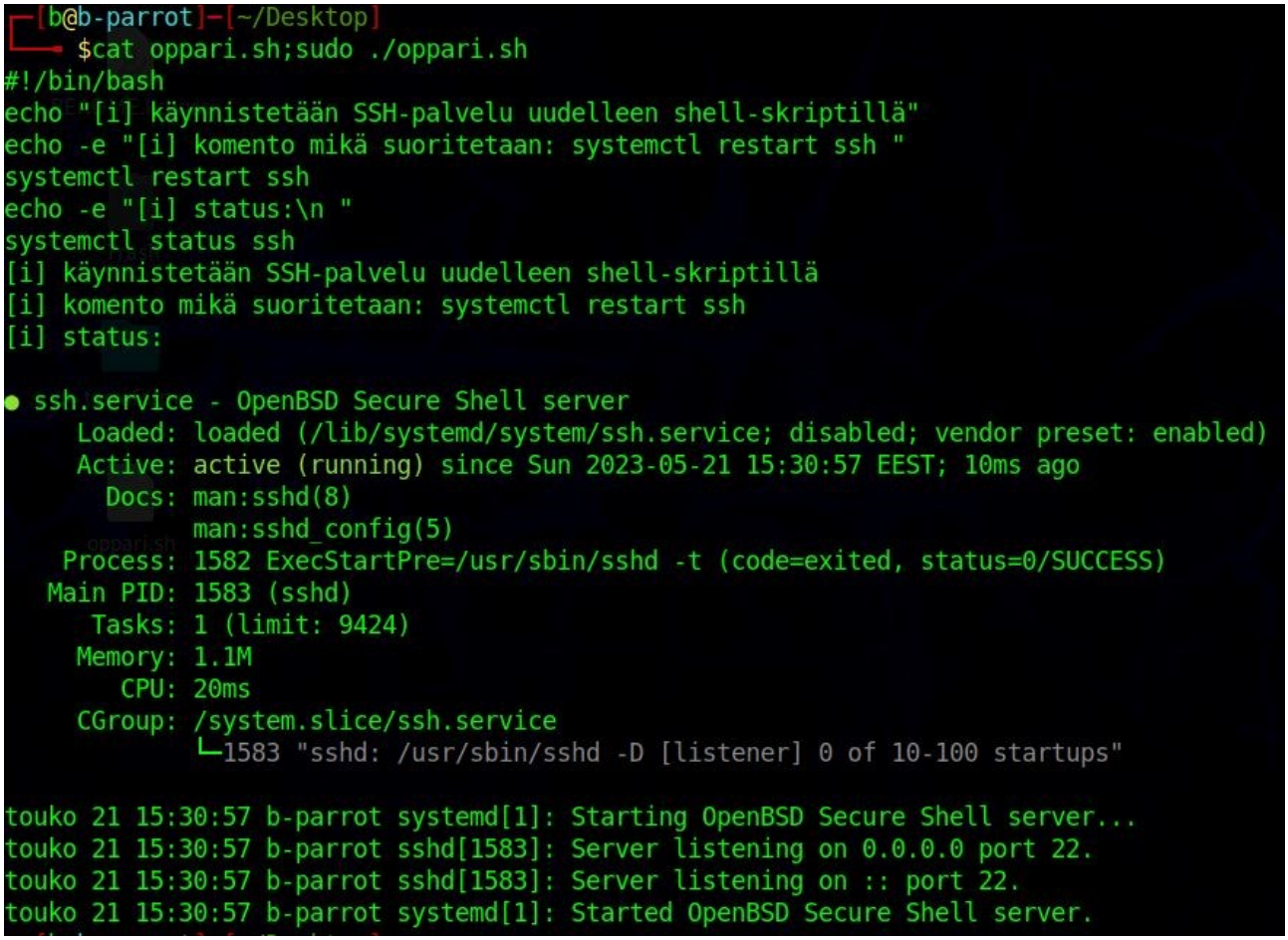
```
PS C:\Users\mikko\OneDrive\Työpöytä> type oppari.ps1
Write-Host "Powershell-skripti joka vaihtaa tiedostosta koira sanan kissaksi : "


(Get-Content -path animals.txt -Raw) -replace 'koira','kissa' | Set-Content animals.txt
PS C:\Users\mikko\OneDrive\Työpöytä> type .\animals.txt
valas
muurahainen
koira

PS C:\Users\mikko\OneDrive\Työpöytä> .\oppari.ps1
Powershell-skripti joka vaihtaa tiedostosta koira sanan kissaksi :
PS C:\Users\mikko\OneDrive\Työpöytä> type .\animals.txt
valas
muurahainen
kissa


PS C:\Users\mikko\OneDrive\Työpöytä>
```

Kuva 4. Powershell-script

Bash on natiivi kuori Linux-käyttöjärjestelmille, kun taas Powershell on Windows-käyttöjärjestelmille aina vuodesta 2007 asti. Molempia käytetään tehtävien automatisointiin. (Asghar, A 2022.)

Ryderin (2018, johdanto) mukaan bash on myös skriptauskieli, jonka on kehittänyt Brian Fox vuonna 1989. Bashissa on monia yleisiä ominaisuuksia ohjelmointikieliin, jotka tekevät siitä käyttökelpoisen yleisiin ohjelmointitehtäviin.

Kuvassa viisi havainnollistetaan bash-skriptin käyttöä palvelun uudelleenkäynnistymiseen. Käyttöjärjestelmien komentojen ketjuttaminen skripteillä on hyvä taito osata ja sitä voidaan hyödyntää kappaleessa 3.1 mainitussa käyttäjädatan käytössä AWS:än virtuaalikoneiden kanssa.



Kuva 5. Bash-skriptin sisältö ja suorittaminen

5 Testausympäristön luonnin automatisointi AWS-pilvipalveluun

5.1 Lähtötilanteen kuvaus ja suunnittelun esittely

Tämän opinnäytetyön toimeksiantaja on CGI Suomi Oy:n kyberturvallisuuskeskus. CGI on vuonna 1976 perustettu IT- ja liiketoimintakonsultoinnin palveluyhtiö, joka tuottaa myös tietoturvapalveluita. CGI Suomen kyberturvallisuuskeskus valvoo yhtiön asiakkaiden ympäristöjä sekä tuottaa kyberturvan monitorointi-, vaste-, tutkinta- ja hyökkäyssimulaatiopalveluita. (CGI 2023b; CGI 2023c.)

Alati kehittyvä uhkamaisema ja uudet tietoturvatrendit pakottavat alalla olevat henkilöt omaksumaan ja oppimaan uusia asioita jatkuvasti. Tätä varten helposti pystytettävä ja tuhottava harjoitusympäristö on todella tärkeässä roolissa niin uusien kuin vanhojenkin työntekijöiden osaamisen kannalta. Eritoten uusille työntekijöille tämän kaltainen testausympäristö toimii ponnahduslautana oppia uusia tekniikoita ja tapoja, miten kyberhyökkäyksiä toteutetaan ja miten ne huomataan. Tuotoksen kohderyhmänä toimii CGI Suomen kyberturvallisuuskeskuksen työntekijät.

Toimeksianto koskee testausympäristön automatisointia AWS-pilvipalveluun, missä lähtökohtaisesti eriytetyssä verkossa sijaitsee hyökkäyskone ja maalikone. Hyökkäyskoneella on tarkoitus laukaista kyberhyökkäyksiä kohti maalikonetta. Testausympäristön automatisointiin käytetään useita eri automatisointitekniikoita ja työkaluja, joita ovat:

- Terraform – Automatisointityökalu, jolla automatisoidaan AWS-pilvipalvelun resurssien luonti
- Ansible – Automatisointityökalu, jolla automatisoidaan tiettyjä toimenpiteitä ennen Terraformin käyttöä.
- Bash – skriptit, jolla mahdollisestaan AWS-instansseille hyökkäystyökalujen lataaminen käynnistysvaiheessa sekä yhteyttä vaativien konfiguraatioiden automatisointi.
- Powershell – skriptit, jolla mahdollistetaan kohdekoneena toimivalle AWS-instanssille tarvittavat muokkaukset, kuten tietoturvatason heikentäminen.

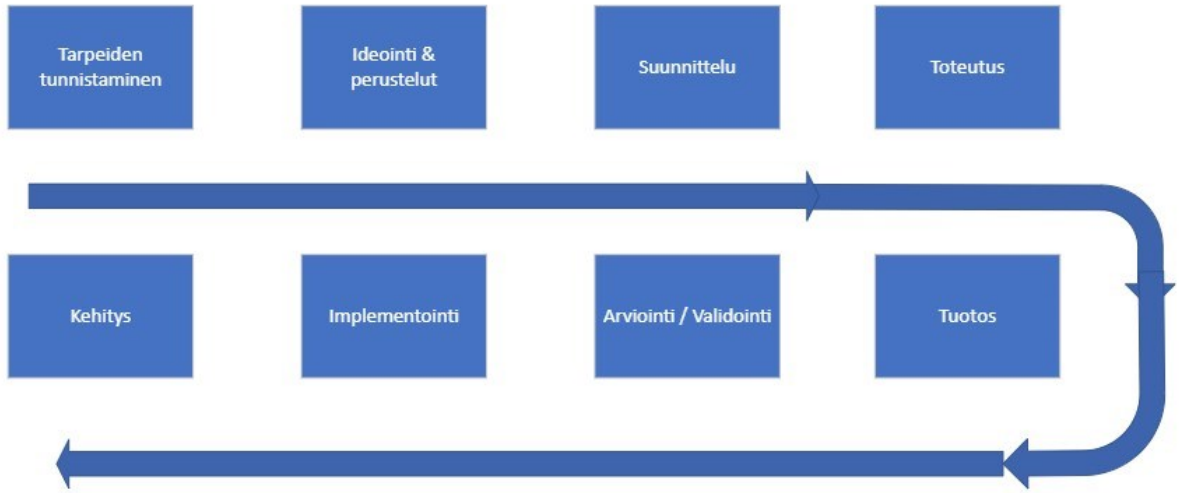
Terraform mahdollistaa infrastruktuurin pystyttämisen koodimaisesti. Sitä voi muuttaa moneen muuhunkin eri käyttötarkoitukseen kuin edellä mainitun testausympäristön automatisointiin. Sillä voitaisiin muun muassa replikoida tuotantoympäristö testidatalla. Kokonaisen infrastruktuurin pystyttäminen ja tuhoaminen Terraformilla on vaivatonta ja kustannustehokasta.

Lopputuotos on hyväksyttävissä, kun testausympäristö sisältää yhteydet yksityiseen aliverkkoon, missä sijaitsee hyökkäyskone ja maalikone. Tämän lisäksi hyökkäyskoneen ja maalikoneen välillä

tulee olla verkkoyhteys. Opinnäytetyön aikaraamien puitteissa toimivan lokitietojen keruun rakentaminen ympäristöön ei ole mahdollista, vaan se siirtyy jatkokehityksen tehtäviin.

5.2 Tuotoksen tuottamisen kuvaus

Tämän toimeksiannon kehittämistyön menetelmänä toimi lineaarinen malli, joka soveltuu hyvin koodipohjaiseen infrastruktuurin luontiin. Brikmanin (2022) mukaan Terraform rohkaisee deklaratiiviseen tyyliin, missä määritellään tuotoksen lopullinen tila ja itse työkalu on vastuussa siitä, miten tila saavutetaan. Salosen, Elorannan, Hautalan ja Kinoksen (2017, 51-52) mukaan lineaarinen ajattelu kehittämistoimintana helpottaa tekijää tai tekijöitä nähdä kokonaisuus eheänä, sillä tehtävät suoritetaan loogisessa järjestyksessä. Kuvassa 4 havainnoidaan kehittämistoiminnan eri vaiheita.



Kuva 6 Kehittämistoiminnan vaiheet (mukaillen Salonen ym. 2017, 52)

Toimeksianto aloitettiin elokuussa 2022 kuvan 6 mukaisilla vaiheilla. Tarve oli selkeästi tunnistettu, sillä kustomoitavaa harjoitusympäristöä ei ole ollut. Tästä syystä perusteluita ei juurikaan tarvittu. Tietysti varsinaiselle toteutuksen sijainnille tarvittiin perusteet. Sijainniksi valittiin AWS-pilvipalvelu, sillä CGI on AWS:n strateginen kumppani. Myös AWS:n tarjoaman pilviresurssien hinta on maailman kärkiluokkaa edullisuudessaan. Työkaluvalintoja mietittiin tarkkaan, mutta ne päätyivät yksiselitteisesti Terraformiin infrastruktuurin rakentamisen kohdalla. Myös Pulumi-nimistä IAC-työkalua harkittiin työn tekemiseen. Terraform oli yksinkertaisesti parempi tähän käyttötarkoitukseen, sillä

Pulumin kanssa tulee haasteita skaalautuvuudessa sekä tilojen hallinta oletuksena tapahtuu Pulumissa SaaS-ratkaisuna eikä paikallisesti. Muu automatisointi on paras tehdä skriptauskielillä, kuten bash ja powershell riippuen käyttöjärjestelmästä. Myös konfiguraatiohallintatyökalujen kuten Ansible, Saltstack ja Puppet:in käyttöä harkittiin skriptien sijaan. Lopulta komentojen osuus oli niin pieni, että se oli helpointa toteuttaa skripteillä. Jatkoa ajatellen Ansible on varsin varteenotettava työkalu hoitamaan palvelinten konfiguraatiot skriptien sijaan.

Suunnitteluvaiheessa tunnistettiin eri komponentit, mitkä oli määrä luoda. Komponenttien erottelu omiin konfiguraatiotiedostoihinsa oli järkevintä hallinnan kannalta. Terraform-työkalun oma työnkulku tuki erittäin hyvin toteutuksen testaamista sillä ympäristön tuhoaminen ja uudelleenpystyttäminen oli vaivatonta. Terraform toimii myös tilallisesti, eli se muistaa missä tilassa infrastruktuuri on ollut, jos jotain muutoksia oli tarve tehdä. Koko ympäristöä ei aina tarvinnut rakentaa uusiksi testauksien aikana.

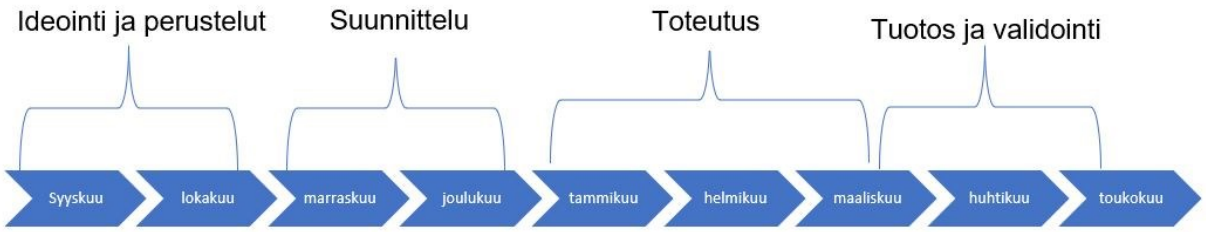
Toteutusvaiheessa ilmeni jatkuvasti lisäkonfiguraatioiden tarpeita liittyen hyökkäyskoneen toimintaan. Hyökkäyskoneena toimivan palvelimen konfiguraatioita kehitettiin iteratiivisesti, ja kaikki tarvittava funktionaalisuus saatiin ladattua edellä mainitulle palvelimelle käyttäjädatan avulla.

Tuotos määriteltiin valmiiksi, kun hyökkäyskoneelta sai yhteyden kohdekoneena toimivaan Windows-palvelimeen. Empirian lopullisen tuotoksen kuvauksen loppupuolella esitellään kybertappoketjun kaltaista lähestymistapaa Windows-palvelimen tunkeutumistestaamiseen.

Validointi toteutettiin muutamalla kohderyhmään kuuluvalla henkilöllä, jotka toimivat tietoturva-asiantuntijoina CGI:n kyberturvakeskuksessa. Ensimmäinen validointi tapahtui omatoimisesti, jotta voidaan varmistua ympäristön toimivuudesta.

Validoinnista saatu suullinen palaute hyödynnettiin ja korjaavia toimenpiteitä tehtiin. Palautetta tuli ohjeistuksesta, että miten ympäristöön pääsee käsiksi, työkalujen puutoksesta hyökkäyskoneella sekä että ympäristön pystyttämistä ei itse voi toteuttaa. Kun validoijilta kysyttiin, että mikä toimi erityisen hyvin, vastaus oli yksiselitteinen, että koko ympäristö toimi niin kuin pitikin. Siihen tässä toteutuksessa pyrittiinkin. Tietenkin tämä työ osoittautuu käyttäjälle varsin yksinkertaisessa muodossa, eli yhteyden luomisella palvelimelle ja toisen palvelimen havaitseminen sieltä. Korjaustoimenpiteitä mitä toteutettiin validoinnin kautta, oli ohjeiden parantaminen. Opinnäytetyön aikarajan puitteissa automatisointi missä ympäristö pystytettäisiin, vaikka nappia painamalla verkkosivulla tai muussa UI-ratkaisussa ei ollut mahdollista. Myöskään hyökkäystyökalujen määrä ei ole ratkaisevaa itse opinnäytetyön toiminnallisuuden kannalta, joten nekin siirtyvät jatkokehityksen puolelle. Hyökkäystyökalujen lisääminen palvelimelle olisi ollut triviaalia mutta kuten jo mainittu, ei

ratkaisevaa tai laatua parantavaa työtä. Jatkokehityksestä kerrotaan opinnäytetyön pohdintaa käsittelevässä kappaleessa. Kuvassa 7 havainnollistetaan opinnäytetyön aikajanaa.



Kuva 7. Opinnäytetyön aikajana karkeasti kuvattuna

5.3 Lopullinen tuotos

Ympäristö rakennetaan AWS-pilvipalveluun, sillä se on kustannustehokasta ja helposti ylläpidettävissä. Tuotosta toteutettiin palvelimella, jonka ainoa tarkoitus on suorittaa Terraform-työkalun ajoja. Esivaatimuksia tuotoksen toteuttamiselle oli Terraform-työkalu sekä pääsy AWS-pilviympäristöön, missä luodaan pääsyavain, jolla ympäristöä voidaan muokata ohjelmallisesti.

Terraform tarvitsee edellä mainitun AWS-avaimen voidakseen toteuttaa tarvittavia muutoksia. Ohjelmallisen avaimen luonti tapahtuu AWS-pilvipalvelun portaalista menemällä IAM-portaaliin. IAM-portaali on identiteetin ja pääsyhallinnan muokkaukseen tarkoitettu palvelu, jossa täytyy luoda käyttäjä, jolle annetaan tarvittavat roolit. Rooleilla määritellään, mitä käyttäjä voi tehdä pilviympäristössä. Nämä samat roolit periytyvät avaimeen, joten on tärkeää luoda mahdollisimman pienillä oikeuksilla oleva avain tuotoksen tekemiseen. Seuraavaksi palvelulle määritellään, että käytetään ulkoista applikaatiota, jonka jälkeen avain on luotu. Avain koostuu tunnisteesta sekä salaisuudesta. Salaisuutta ei tulisi ikinä julkistaa. Pilvipalveluiden avainten vuotaminen julkisiksi aiheuttaa monia tietoturvapoikkeamia joka vuosi.

Itse Terraform-konfiguraatiot kehitetään sille määritellyllä palvelimella, joten koko ympäristön rakennus tapahtuu siellä. Luodaan juuri luoduille AWS-avaimille tiedosto, mistä Terraform osaa käyttää niitä. Kuten kuvasta 8 voidaan havaita, bash-skriptillä luodaan käyttäjän kotihakemistoon piilotettu kansio ja sen sisään tiedosto, jossa on AWS-avaimien käyttöä tarvittavat tiedot.

```
#!/bin/bash
mkdir -p ~/.aws/
cat <<EOF > ~/.aws/credentials
[default]
aws_access_key_id = <avain_tulee_tähän>
aws_secret_access_key = <avaimen_salaisuus_tulee_tähän>

EOF
```

Kuva 8. Bash-skripti

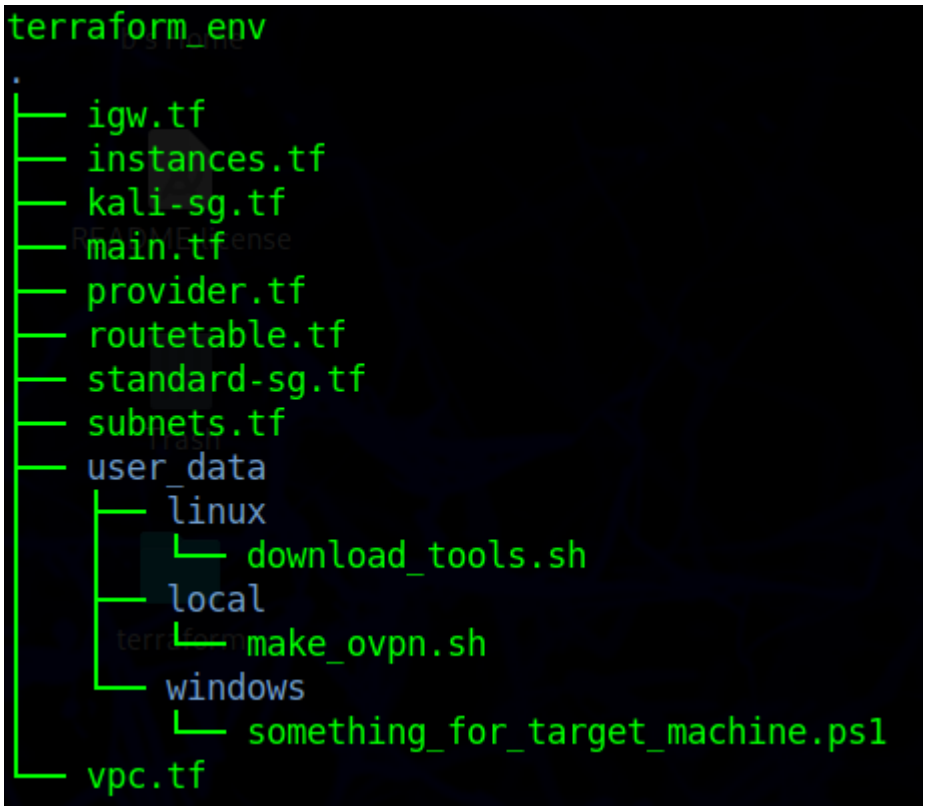
Seuraavaksi luodaan hakemisto Terraformia varten. Terraform kokoaa kaikki .tf-päätteiset tiedostot samasta hakemistosta ja tekee muutokset niiden sisällön perusteella. Hyvien käytäntöjen mukaan erityyppisten resurssien konfiguraatiot tulisi sijoittaa omaan konfiguraatiotiedostoonsa, jotta ongelmanselvittely ja itse rakentaminen ei hankaloituisi. Toteutusta suunnitellessa konfiguraatiot voidaan erotella seuraavasti:

- Internet yhdyskäytävä
- Internet rajapinta
- julkinen IP-osoite
- SSH-avaimet
- AWS-Instanssit
- Turvaryhmät
- Reititystaulu
- Aliverkot
- Virtuaalinen yksityinen pilvi
- Tarjoaja (AWS)

Konfiguraatiotiedostojen määrää voisi kaventaa vielä yhdistämällä tietynlaiset resurssit omiin konfiguraatiotiedostoihinsa. Hyvänä esimerkkinä toimisi internet yhdyskäytävän ja internet rajapinnan rakentamista varten toimivat konfiguraatiot yhteen konfiguraatioon.

Näiden lisäksi on hyvä luoda erillinen hakemisto automatisoinneille, joita halutaan tapahtuvan AWS-instanssin käynnistysvaiheessa. AWS:ssä on mahdollista suorittaa komentoja instanssin käynnistysvaiheessa. Tätä kutsutaan käyttäjädatan syöttämiseksi instanssiin. Myös Terraform pystyy käyttämään tätä tärkeää ominaisuutta. Hyviä käytäntöjä seuraten toteuttaja voi luoda hakemiston, jonka sisään luodaan kuvaavat hakemistot eri komennoille tai skripteille, mitä halutaan kohdeinstansseilla suorittaa. Tämä selkeyttää luomisvaiheessa toteuttajaa erottelemaan, mitä on määrä suorittaa milläkin instanssilla. Kuvassa 9 näkyy selkeästi toteutettu Terraform-työtila missä

konfiguraatiotiedostot on eroteltu käyttötarkoituksiensa perusteella ja käyttäjädataa varten olevat skriptit on siirretty alihakemistoihin.



Kuva 9. Puunäkymä Terraform-työtilasta

Ympäristö sisältää neljä palvelinta. Kaksi näistä palvelimista on tietoliikenneyhteyksiä varten olevia instansseja ja kaksi muuta toimivat hyökkäys- ja maalikoneina. Jotta yksityiseen aliverkkoon olisi mahdollista päästä julkisesta verkosta, siihen voidaan käyttää VPN-palvelinta, joka reitittää liikennettä yksityisessä aliverkossa olevalle koneelle. Turvaryhmämääritykset ja liikenteen reitittäminen mahdollistavat myös yhteyden ottamisen, vaikka pelkästään käyttäjän koneen IP-osoitteesta. Tällä toimenpiteellä saadaan pienennettyä pinta-alaa, mistä yhteyksiä voidaan ottaa palvelimille.

Toinen tietoliikenneyhteyksiä varten toimiva palvelin toimii välityspalvelimena yksityisissä aliverkoissa toimiville instansseille. Tämän välityspalvelimen kautta ne saavat yhteyden internettiin koneiden luomisvaiheessa, milloin edellä mainittu käyttäjädata mahdollistaa erinäisten ohjelmien ja kirjastojen asennuksen instansseille. Liikenteen kontrollointi resursseille tapahtuu turvaryhmämäärityksissä, jotka konfiguroidaan tarpeiden mukaan.

Kokonaan eriytetyssä ympäristössä edellä mainittua välityspalvelinta ei olisi. Eriytetyssä ympäristössä tarvittavien kirjastojen ja binäärien lataaminen tapahtuisi niiden vientinä suoraan koneille käyttäjädataa hyödyntäen. Käyttötapauksen mukaan tämä verkkoyhteys hyökkäyskoneelta internettiin on kuitenkin hyvä olla olemassa. Testausympäristöä käyttävän henkilöstön työkalupreferenssit vaihtelevat ja tämä yhteys mahdollistaa eri työkalujen lataamisen yksityisessä aliverkossa olevalle hyökkäyskoneelle. Tässä piilee myös riskinsä. Hyökkäyskoneelta voidaan vahingossa suorittaa kyberhyökkäyksiä myös julkisessa internetissä oleville kohteille. Linux-käyttöjärjestelmien välityspalvelinasetukset voidaan määrittää ympäristömuuttujilla.

```
export http_proxy="http://<proxy_palvelimen_ip_tähän:portti>"
export https_proxy="http://<proxy_palvelimen_ip_tähän:portti>"
```

Kuva 10. http-välityspalvelimen asettaminen Linux-käyttöjärjestelmällä

Tästä syystä tietoturvataidot ja ymmärrys siitä, mitä tekee, ovat tärkeässä roolissa tämänkaltaisen testausympäristön käytössä. Tietomurron yritykseen syyllistyminen ei vaadi, kuin epähuomiossa väärän IP-osoitteen tai nimen syöttämisen hyökkäystyökalulle, jos koneelta on yhteys internettiin. Erityisesti tunkeutumistestauksessa tärkeä nyrkkisääntö on olla käyttämättä työkaluja joidenka toimintaa ei ymmärrä.

Kun Terraform aloittaa ympäristön rakentamisen ja sille on määritelty käyttäjädatan suorittaminen palvelimilla, automaatioon kuten bash-skriptiin voidaan määritellä välityspalvelimen käyttö. Täten voidaan mahdollistaa internetistä pakettien lataaminen juuri luodulle instanssille. Kun palvelin on käynnistynyt, yllä olevan kuvan 10 mukainen välityspalvelin-konfiguraatio ei ole enää voimassa. Käyttäjä pystyy jälkikäteen asettamaan sen käyttöön, mutta se ei välttämättä ole tarpeellista. Käyttäjädataa on käytetty tämän ympäristön osin kyberhyökkäystyökalujen lataamiseen sille tarkoitetulle instanssille sekä Windows-palvelimen tietoturvatason heikentämiseen tarkoituksellisesti. Käyttäjädatan syöttämisellä voitaisiin myös viedä SSH-avaimia instansseille, joihin yhteys halutaan muodostaa. Koska Terraform tukee natiivisti SSH-avaimien luontia, tässä projektissa avainten vientiä ei ole toteutettu käyttäjädatan kautta.

VPN-palvelinta tarvitaan yhteyden muodostamiseen käyttäjän omalta koneeltaan yksityisessä aliverkossa olevalle koneelle. Tässä tapauksessa käytetään jo valmiiksi konfiguroitua Linux-palvelinta, missä pyörii Openvpn-ohjelmisto. Valmista, konfiguroitua VPN-palvelinta ei tarvitsisi, sillä käyttäjädatalla on mahdollista asentaa ja konfiguroida VPN-palvelin kokonaan. Käyttäjädata tukee myös tiedostojen siirtämistä palvelimelta, jolla Terraformia komennetaan kohdekoneille, jotka

tullaan luomaan. Turvaryhmillä voidaan granulaarisesti määritellä, mistä hallintayhteyden saa ottaa kohdekoneelle. Näin saadaan pienennettyä vektoria, mistä yhteys voidaan aloittaa ja mihin palveluun se liittyy. Ansible-automatisointeja ohjaavalla ja suorittavalla palvelimella ajetaan tarvittavat VPN-sertifikaatit Terraform-palvelimelle. Terraform-palvelimella käyttäjädataa hyödyntäen viedään tarvittavat sertifikaatit ja avaimet juuri pystytetylle VPN-palvelimelle, jotta käyttäjänpuoleinen konfiguraatio voidaan luoda. Tätä konfiguraatiota käytetään ottamaan yhteys yksityisessä aliverkossa olevalle instanssille. Tämän käyttäjänpuoleisen konfiguraation rakentamisen voi automatisoida mm. bash-skriptillä. Kuvassa 11 näkyy lokaalisti ajettava bash-skripti millä rakennettaan käyttäjälle openvpn-konfiguraatiotiedosto jonka avulla yhteys voidaan muodostaa harjoitusympäristöön.

```
#!/usr/bin/bash

userparam="$1"

#assigning variables to fetch data to populate user.ovpn
eip=$(cd /opt/path/to/terraform/env/;terraform output -raw vpn_public_eip)
ca=$(cat /opt/openvpn/easyrsa3/pki/this_is_ca.crt)
cert=$(cat /opt/openvpn/easyrsa3/pki/issued/this_is_client.crt)
key=$(cat /opt/openvpn/easyrsa3/pki/private/this_is_client.key)

#try catch when EIP is not created

if [[ $eip == *"Warning"*  ]]; then
  echo -e "[!] Error occured in Elastic IP fetching with terraform command.You might have not created elastic IP.\n[i] exiting gracefully."
  exit 1
fi
mkdir -p /path/to/vpn/users/$userparam/
cat <<EOF > /path/to/vpn/users/$userparam/$userparam.ovpn
client
dev tun
proto tcp
remote $eip 443
resolv-retry infinite
nobind
remote-cert-tls server
cipher AES-256-GCM
verb 3
<ca>
$ca
</ca>
<cert>
$cert
</cert>
<key>
$key
</key>
EOF
```

Kuva 11. Bash-skripti, jolla voidaan muokata openvpn-käyttäjäkonfiguraatio toimivaksi

Rivillä kuusi määritellään muuttujalle ”eip” arvo, joka luodaan Terraformilla. Terraformissa voidaan printata tuloksia ulos terminaaliin näkyviin käyttämällä output-konfiguraatioblokkia. Luotu ”eip” arvo tulee olemaan VPN-palvelimen julkinen IP-osoite, joka on AWS kontekstissa elastinen julkinen IP. Tämä lyhyt skripti tarkistaa myös, onko Terraform ajossa tapahtunut virhe, jonka takia tuota julkista IP-osoitetta ei olisikaan tullut. Jos näin on käynyt, skripti lopettaa ajamisensa, eikä jatka konfiguraation luomista.

Terraformiin on sisäänrakennettu konfiguraatiomuotoilutyökalu. Terraform-konfiguraation syntaksi vaatii tiettyjä sisennyksiä, jotta konfiguraatio olisi validi. Käyttämällä komentoa ”terraform fmt”

Terraform muotoilee syntaksisi oikeanlaiseksi automaattisesti. Muotoilukomennon käyttäminen ei ole välttämätöntä, mutta siitä on suuri apu syntaksin tarkastuksen kanssa. Kuitenkin jos konfiguraatio sisältää syntaksivirheitä, kuten puuttuvia aaltosulkeita, niin niitä se ei luo.

Terraformin alustamiskomennolla alustetaan hakemisto, missä konfiguraatiotiedostot sijaitsevat. Alustamisessa tarkoitetaan tässä kontekstissa sitä, että kerrotaan Terraform:ille, mitä konfiguraatioita on ja mitä ne tekevät. Joka kerta kun konfiguraatiotiedostot muuttuvat, tulee tämä alustus tehdä. Kuvassa 12 näkyy onnistunut alustus sekä alustetut toimijat. Jos konfiguraatiotiedosto olisi sisältänyt virheitä, alustaminen olisi epäonnistunut. Jos jostain syystä alustaminen olisikin mennyt läpi, Terraformin suunnitteluvaihe olisi huomannut sen ja ilmoittanut virheviestillä puuttuvan merkin tai konfiguraation.

```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/template from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/tls v4.0.4
- Using previously-installed hashicorp/template v2.2.0
- Using previously-installed hashicorp/aws v4.27.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Kuva 12. Terraformin onnistunut alustus

Terraform:in työnkulun mukaan seuraavaksi tulee suunnitella tuleva muutos käyttämällä ”terraform plan”-komentoa. Tämän komennon tulostus näyttää kaikki tulevat muutokset, jotka konfiguraatiot tekevät. Kuva 13 näyttää, että suunnitelman mukaan 19 uutta resurssia luodaan.

```
Plan: 19 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + advmachine_private_ip = "10.0.2.15"
  + private_key           = (sensitive value)
  + proxy_public_ip       = (known after apply)
  + target_win_private_ip = "win priv ip : 10.0.2.11"
  + vpn_public_eip        = (known after apply)

Saved the plan to: oppari

To perform exactly these actions, run the following command to apply:
    terraform apply "oppari"
```

Kuva 13. Terraform plan –komennon tulostuksen loppuosa

Terraformin työnkulun pakollinen osa, eli suunnitteluvaihe on toteutettu. Jos konfiguraatiotiedostot olisivat sisältäneet virheitä, tämän komennon ajaminen ei olisi mennyt onnistuneesti maaliin.

```
Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:

advmachine_private_ip = "10.0.2.15"
private_key = <sensitive>
proxy_public_ip = <sensitive>
target_win_private_ip = "win priv ip : 10.0.2.11"
vpn_public_eip = <sensitive>
```

Kuva 14. Onnistuneen terraform apply-komennon tuloste

Jos konfiguraatiotiedostot ovat tehty oikein, koko ympäristön pitäisi nyt olla toiminnassa. Koska ympäristö mihin kaikki luotiin sijaitsee AWS-pilvipalvelussa, myös pilvipalvelun portaalista näkee juuri luodut instanssit kuten kuvassa 15 näkyy. AWS-portaalista näkee muutkin juuri luodut resurssit.

Name	Instance ID	Instance state	Instance type	Status check
testenv-tf-proxy	i-012e2c86eaa754fad	Running	t3.micro	2/2 checks passed
testenv-tf-vpn	i-031dbba8c7c20aff0	Running	t3.micro	2/2 checks passed
testenv-tf-target-machine-win	i-08fdd4783b2e563fc	Running	t3.micro	2/2 checks passed
testenv-tf-ec2-kali	i-0f959ecf378f5b0d7	Running	t3.medium	2/2 checks passed

Kuva 15. AWS-portaalin instanssit-välilehti

Seuraavaksi siirrytään koneelle, mistä yhteys voidaan muodostaa VPN-palvelimelle. Koneella olisi suotavaa olla OpenVPN-käyttäjäohjelmisto, millä yhteyksiä voidaan helposti muodostaa. Kuvan 11 skriptillä luodaan ovpn-tiedostopäätteinen konfiguraatiotiedosto, millä luodaan turvallinen VPN-tunneli hyökkäyskoneelle käyttäen edellä mainittua ohjelmaa. VPN-tunneli mahdollistaa salatun

yhteyden käyttäjän ja VPN-palvelimen välillä ja VPN-palvelin itsessään on konfiguroitu reitittämään liikenne aliverkkoon missä harjoituskoneet sijaitsevat.

Kuvassa 16 näkyy porttiskannaus nmap-työkalulla. Porttiskannaus toteutetaan hyökkäyskoneelta käsin, jonka käyttöjärjestelmä on yleinen tunkeutumistestaamiseen tarkoitettu Kali Linux. Nmap on yksi maailman tunnetuimmista verkkoskannereista. Nmap – ohjelmaa ajettiin kahdella eri parametrillä, jotka näkyvät kuvassa väliviivamerkkien jälkeen. Nämä parametrit mahdollistavat nmap-ohjelman käyttämään oletusskriptejä ja tunnistamaan versiot. Työkalulla skannattiin kohdekoneen avoimet portit, jotka paljastaa avonaisen Windows etätyöpöytä –sovelluksen. Tämä tarkoittaa, että etätyöpöytään voi yrittää ottaa yhteyksiä, mutta autentikoitumiseen vaadittavia tunnuksia ei ole.

```
┌──(root㉿kali)-[~]
└─# nmap -Pn -sC -sV -T5 10.0.2.11
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-12 18:14 UTC
Nmap scan report for 10.0.2.11
Host is up (0.00013s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: SUPERHARDENEDWI
|   NetBIOS_Domain_Name: SUPERHARDENEDWI
|   NetBIOS_Computer_Name: SUPERHARDENEDWI
|   DNS_Domain_Name: SuperHardenedWindowsMachine
|   DNS_Computer_Name: SuperHardenedWindowsMachine
|   Product_Version: 10.0.20348
|_  System_Time: 2023-05-12T18:15:02+00:00
| ssl-cert: Subject: commonName=SuperHardenedWindowsMachine
| Not valid before: 2023-05-11T09:24:31
|_Not valid after:  2023-11-10T09:24:31
|_ssl-date: 2023-05-12T18:15:07+00:00; +1s from scanner time.
MAC Address: 06:E9:FD:9E:12:30 (Unknown)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.83 seconds
```

Kuva 16. Porttiskannauksen tulokset

Tulokset paljastavat myös liudan muita tietoja. Oletus-skriptien käyttö lähetti keskeneräisen autentikaatiopyynnön tyhjällä tunnuksella kohteeseen, joka palauttaa NTLMSSP-viestin, joka taas paljastaa kohteen nimen ja käyttöjärjestelmän. Tämä kokonaistieto, joka skannauksesta on saatu, kuuluu kybertappoketjun ensimmäiseen vaiheeseen, aktiiviseen tiedusteluun. Hyökkääjällä on siis tiedossa kohdekoneen nimi, käyttöjärjestelmä ja avonainen portti palveluun, jolla etäyhteyden voisi muodostaa. Edistynyt hyökkääjä voisi näillä tiedoilla jo yrittää sosiaalista manipuloinnin keinoin saada tietoonsa käyttäjätunnukset, joita voisi käyttää etäyhteyden muodostamiseen. Koska tämä on testausympäristö, missä ei ole oikeita käyttäjiä, niin sosiaalisen manipuloinnin tekniikat eivät toimi.

Kohdekoneen etätyöpöytä-sovelluksen autentikaatiota voi koittaa murtaa väsytyshyökkäyksellä. Väsytyshyökkäyksiä on monenlaisia ja niissä pyritään järjestelmällisesti yrityksen ja erehdyksen kautta arvaamaan salasana oikein. Tunnettuja väsytyshyökkäystekniikoita ovat muun muassa sanakirja- ja raakahyökkäys. Sanakirjahyökkäyksessä hyökkääjällä on kerätty lista potentiaalista salasanoista ja käyttäjätunnuksista, joita käytetään automatisoidusti kokeillen kaikkien näiden sanojen yhdistelmiä. Raakahyökkäystä voidaan kutsua myös nimellä kryptoanalyysihyökkäys. Tiedusteluvaiheessa kerätyt tiedot, kuten yrityksen työntekijöiden nimet tai käyttäjätunnus-nimeämispolitiikat, ovat hyödyllisiä tietoja. Jo näiden avulla voidaan muodostaa sanakirja käyttäjistä. Salasanakirjan muodostamiseen voi käyttää apunaan tietomurtojen kohteina olleita palveluita, joita ihmiset saattaisivat käyttää. Joidenkin tietomurtojen käyttäjätunnus- tai salasanatietokannat ovat vuotaneet hyökkäyksen takia Internettiin, mistä ne ovat kaikkien saatavilla.

Molemmat taktiikat voivat olla todella aikaa vieviä ja helposti estettävissä eri teknologioiden avulla, kuten kaksivaiheisen tunnistautumisen. Kaksivaiheisessa tunnistautumisessa onnistuneen käyttäjätunnus ja salasana –kombinaation jälkeen tulee vielä syöttää turvakoodi jostain ulkoisesta laitteesta tai applikaatiosta. Väsytyshyökkäyksiä ei tulisi käyttää, jollei hyökkääjä ole aivan varma, että sitä estäviä kontrolleja ei ole. Vaarana on myös tunnusten lukkiutuminen jatkuvien epäonnistuneiden kirjautumisien jäljiltä. Seuraavassa testausskenaariossa oletetaan, että turvakontrolleja väsytyshyökkäyksiin ei ole. Kuvassa 17 suoritetaan sanakirjahyökkäys kohdekoneelle käyttämällä Windows-käyttöjärjestelmän oletus-pääkäyttäjää ”Administrator”:ia. Vastaava käyttötunnus oikeuksiltaan Linux-käyttöjärjestelmissä olisi ”root”.

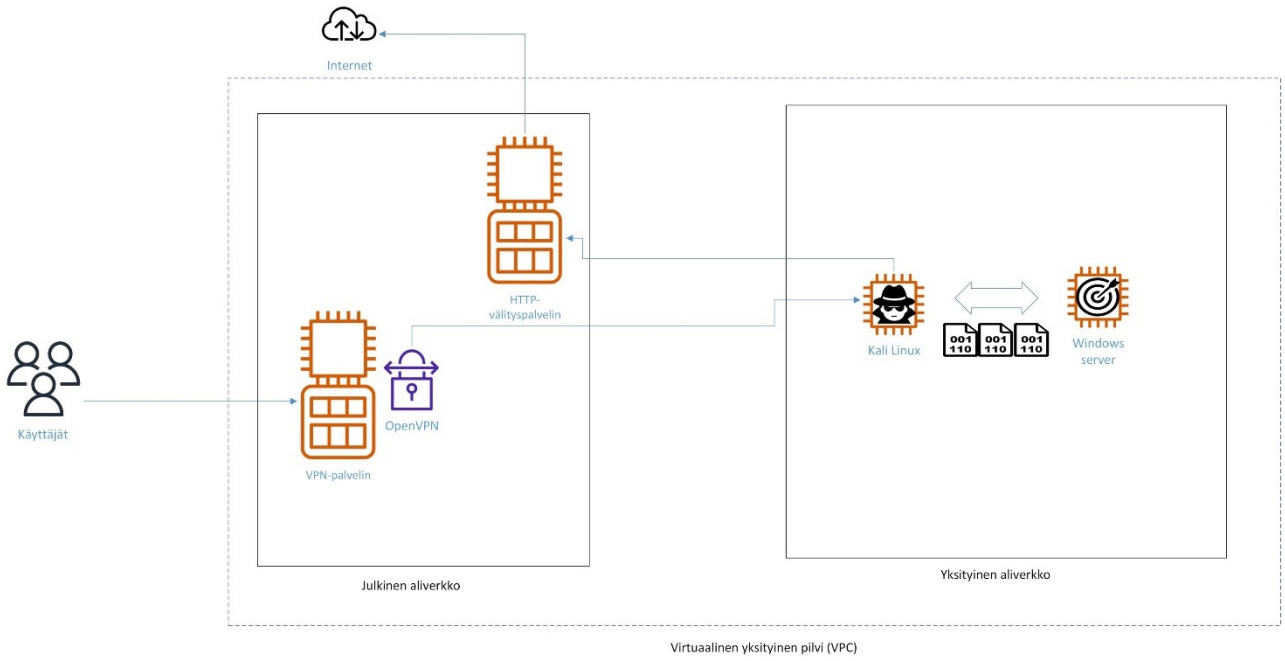
```
┌──(root㉿kali)-[~]
└─# hydra -l Administrator -P pass rdp://10.0.2.11
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or f
or illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-12 18:15:24
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connections and -
W 1 or -W 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 10 login tries (l:1/p:10), ~3 tries per task
[DATA] attacking rdp://10.0.2.11:3389/
[3389][rdp] host: 10.0.2.11   login: Administrator   password: supersecret
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-12 18:15:25
```

Kuva 17. Onnistunut väsytyshyökkäys

Yllä olevassa kuvassa 17 näkyy vihreällä tekstillä toimivat tunnukset, jotka saatiin arvattua oikein hydra-työkalulla. Hydra on tunnettu sisäänkirjautumismurtotyökalu, joka tukee useita eri protokollia. Näistä yksi protokolla on kohteena toimiva Windows etätyöpöytä-sovelluksen protokolla nimeltään Windows etätyöpöytäprotokolla. Hydralla ajettiin sanakirjahyökkäys Windows-palvelinta vasten ja

oikea salasana löytyi. Oikean maailman skenaarioissa salasanat eivät olisi näin helppoja ja väsytyshyökkäyksen mahdollisuutta etätyöpöydälle tuskin olisi.

Kuvassa 18 näkyy yksinkertaistettu arkkitehtuurikuva testausympäristöstä. Sen tarkoitus on havainnollistaa käyttäjän reitti ympäristöön ja tietoliikenneavaus internettiin yksityisestä, muutoin eristetystä aliverkosta. Yksityisessä aliverkossa tietoliikenne kulkee vapaasti koko verkon alueella.



Kuva 18. Pelkistetty arkkitehtuurikuva ympäristöstä

5.3.1 Validointi

Tässä luvussa kerrotaan ympäristön validoinnista. Validoinnin suoritti kaksi CGI kyberturvakeskuksen asiantuntijaa. Luvussa 5.2 kerrottiin validoinnin tuomat korjausehdotukset.

Validointi todettiin onnistuneeksi, kun validoijat pääsivät testausympäristöön sisään ja pystyivät näkemään aliverkossa olevan kohdekoneen. Testausympäristön oikea käyttötarkoitus on harjoitella kyberhyökkäystekniikoita sekä niiden havaitsemista. Tästä syystä halusin validoijien myös kokeilla murtautua kohdekoneelle. Kohdekone oli siis tavallinen Windows-palvelin, jonka

järjestelmänvalvoja-tunnuksen salasana oli muutettu helposti murrettavaksi. Helpottaakseni lisävalidointiosuutta loi hyökkäyskoneelle käyttäjätunnus- ja salasanalistat millä voidaan murtautua kohdekoneelle.

Molemmat validoijat pääsivät ympäristöön vaivattomasti ja löysivät aliverkossa sijaitsevan kohdekoneen skannaamalla verkkoavaruutta. Alla olevassa kuvassa 19 näkyy verkon skannaustulokset. Tuloksissa näkyy kahden eri koneen tulokset mistä alemmat tulokset ovat itse hyökkäyskoneen avoimet portit.

```
┌──(root㉿kali)-[~]
└─# nmap -sV 10.0.2.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-15 06:21 UTC
Nmap scan report for 10.0.2.1
Host is up (0.00025s latency).
All 1000 scanned ports on 10.0.2.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 06:5E:D4:42:DE:38 (Unknown)

Nmap scan report for 10.0.2.11
Host is up (0.00017s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: 06:BD:46:B1:D7:5C (Unknown)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 10.0.2.15
Host is up (0.0000040s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 9.2p1 Debian 2 (protocol 2.0)
3389/tcp open  ms-wbt-server xrdp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (3 hosts up) scanned in 29.23 seconds

┌──(root㉿kali)-[~]
└─# ls
passwordlist.lst  userlist.lst
```

Kuva 19. Validointia tekevän henkilön porttiskannaus

Molemmat validoijat myös lähtivät samalla tekniikalla ja työkalulla murtautumaan Windows-palvelimelle. Tekniikkana toimi sanakirjahyökkäys avonaisena olevaan Windows-käyttöjärjestelmän etätyöpöytäohjelman porttiin kuten kuvassa 20 näkyy.

```
┌──(root㉿kali)-[~]
└─# hydra -L userlist.lst -P passwordlist.lst rdp://10.0.2.11
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-15 06:24:15
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connections a
nd -W 1 or -W 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 66 login tries (l:6/p:11), ~17 tries per task
[DATA] attacking rdp://10.0.2.11:3389/
[3389][rdp] host: 10.0.2.11   login: Administrator   password: supersecret
[ERROR] freerdp: The connection failed to establish.
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-15 06:24:22

┌──(root㉿kali)-[~]
└─#
```

Kuva 20. Väsytyshyökkäys ja kohdekoneen tunnukset

Itse murtautumishaastetta ei ollut järkevää tehdä hankalaksi koska oikea validoinnin kohde oli ympäristön toimivuus. Näiden validointien perusteella voidaan todentaa, että ympäristö on täysin toimiva.

6 Pohdinta

Jatkokehitystarpeita ja suuntia on monia. Eräs isoimmista ja kiinnostavista jatkokehityskohteista olisi haavoittuvien palvelimien lisääminen ympäristöön maalikoneiksi. Näitä voisi mallintaa eri vaikeustasoisiksi sekä eri kokoisiksi. Kokoerolla tarkoitetaan useamman maalikoneen ketjua missä harjoiteltaisiin sivuttaissiirtymien tekoa ja tunnistamista. Myös kokonaisvaltaisen aktiivihakemisto-ympäristön rakentaminen ja testaaminen olisi hyvä lisä.

Harjoitusympäristö soveltuu myös eri antivirus- ja päätelaitesuojausohjelmistojen testaamiseen. Yleisesti antivirus- ja päätelaitesuojausohjelmistoja testataan kokeilemalla eri hyökkäystekniikoita ja kirjaamalla ylös mitä se pystyy havaitsemaan ja mitä ei. Tämänkaltainen harjoittelu voidaan myös hyödyntää tunkeutumistestaus-toimeksiannoissa, jos saadaan selville asiakkaan tietoturva-ohjelmistot.

Hallinnollisesta ja operationaalisesta näkökulmasta kaikenkattava lokienkeruu ja erinäisten lokienkäsittelyjärjestelmien käyttö on pakko lisätä kehitystehtävien kärkeen. Näin päästään valvomaan palvelimien ja verkkojen toimintaa. Myös puolustamis- ja detektiokyvyn parantamiseen liittyvä harjoittelu vaatii lokienkeruun ja palvelimen millä lokeja voidaan käsitellä keskitetysti. Lokienkeruun lisäksi eri päätelaitesuojaus- ja antivirusohjelmistojen käyttäminen ympäristössä eräänlaisina sensoreina tuovat lisää näkyvyyttä puolustajille ja hankaloittavat hyökkääjän testausta. Molemmat osapuolet hyötyvät suuresti tämänkaltaisesta harjoittelusta.

Ympäristöstä tuli juuri sellainen, kun oli suunniteltu. Terraform kielenä itsessään on helposti uusiokäytettävä ja ympäristö on rakennettu helposti skaalattavaksi. Uusiokäytettävyys on yksi suurimmista hyödyistä infrastruktuuri koodina-menetelmää käytettävässä infrastruktuurin rakentamisessa. Terraform:in avulla voidaan rakentaa eräänlaisia palasia kuten toimiva lokienkeräysjärjestelmä, jota voidaan helposti siirtää aina uuteen ympäristöön, jos sellainen luodaan.

Palvelinten konfigurointi käyttäjädatalla tulee varmasti jossakin vaiheessa siirtää paremmalle työkalulle kuin skripteille. Ansible-konfiguraatiohallintatyökalu tulee melko varmasti otettua käyttöön skriptien tilalle.

Mielestäni toimeksianto onnistui hyvin ja pysyi aikataulun puitteissa. Lineaarisen mallin tuoma hyöty rakentamiseen toimii hyvin ja tulevan, iteratiivisen validoinnin mitä työkollegoilta saa tuo varmasti mielenkiintoisia projekteja tämän asian suhteen.

7 Lähteet

Ansible s.a.a. How Ansible Works. Luettavissa: https://www.ansible.com/overview/how-ansible-works. Luettu: 1.5.2023.

Asghar, A. 2022. Powershell vs Bash/shell scripting. Luettavissa: https://linuxhint.com/powershell-bash-shell-scripting/. Luettu: 21.5.2023.

AWS s.a.a. Penetration Testing. Luettavissa: https://aws.amazon.com/security/penetration-testing/. Luettu: 12.3.2023.

AWS s.a.b. What is Amazon EC2? Luettavissa: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. Luettu: 6.4.2023.

AWS s.a.c. What is cloud computing? Luettavissa: https://aws.amazon.com/what-is-cloud-computing/. Luettu: 6.4.2023.

AWS s.a.d. What is aws? Luettavissa: https://aws.amazon.com/what-is-aws/. Luettu: 12.3.2023.

AWS s.a.e. Work with instance user data. Luettavissa: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-add-user-data.html. Luettu: 7.4.2023.

AWS s.a.f. Run commands on your Windows instance at launch. Luettavissa: https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ec2-windows-user-data.html. Luettu: 7.4.2023.

AWS s.a.g. Amazon EC2 security groups for Linux instances. Luettavissa: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html. Luettu: 7.4.2023.

AWS s.a.h. What is Amazon VPC? Luettavissa: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html. Luettu: 7.4.2023.

AWS s.a.i. VPC Internet Gateway. Luettavissa: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html. Luettu 9.4.2023.

AWS s.a.j. How Amazon VPC works. Luettavissa: https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html. Luettu: 9.4.2023.

AWS s.a.k. NAT gateways. Luettavissa: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html. Luettu: 9.4.2023.

AWS s.a.l. VPC Route Tables. Luettavissa: https://docs.aws.amazon.com/vpc/latest/user-guide/VPC_Route_Tables.html. Luettu: 9.4.2023.

Barney, N. & Gillis, A.S. 2022. Amazon Web Services (AWS). Luettavissa: https://www.tech-target.com/searchaws/definition/Amazon-Web-Services. Luettu: 12.3.2023.

Beuran, R., Pham, C., Tang, D., Chinen, K., Tan, Y & Shinoda, Y. 2017. CyTrONE: An Integrated Cybersecurity Training Framework. Luettavissa: DOI: 10.5220/0006206401570166. Luettu: 5.5.2023.

Bichard,L. 2019. Declarative vs. Imperative infrastructure as a code. Luettavissa: https://openupthecloud.com/declarative-vs-imperative-infra/. Luettu: 8.5.2023.

Blažič, B. 2019. The cybersecurity labour shortage in Europe: Moving to a new concept for education and training. Elektroninen tietoaineisto. Luettavissa: https://doi.org/10.1016/j.tech-soc.2021.101769. Luettu: 6.5.2023.

Brikman,Y 2017. Terraform: Up and running. O'Reilly Media, Inc. Delaware. E-kirja. Luettu: 25.4.2023.

Brikman,Y 2022. Terraform: Up and running. 3. uudistettu painos. O'Reilly Media, Inc. Delaware. E-kirja. Luettu: 7.5.2023.

Campbell,B. 2019. The Definitive Guide to AWS Infrastructure Automation : Craft Infrastructure-as-Code Solutions. Apress. Berliini. E-kirja. Luettu: 12.5.2023.

CGI 2023a. CGI yrityksenä. Luettavissa: https://www.cgi.com/fi/fi/cgi-yrityksena. Luettu: 27.5.2023.

CGI 2023b. Kyberturvallisuuskeskus SOC. Luettavissa: https://www.cgi.com/fi/fi/tietoturva/SOC Luettu: 3.5.2023.

CGI 2023c. Offensiiviset tietoturvapalvelut. Luettavissa: https://www.cgi.com/fi/fi/tietoturva/offensiivinen-tietoturvapalvelu. Luettu: 21.5.2023

Checkpoint, s.a. What are AWS Security Groups? Luettavissa: https://www.checkpoint.com/cyber-hub/cloud-security/what-is-aws-security-groups/. Luettu: 13.5.2023.

EDUCBA s.a.a. Ansible Inventory File. Luettavissa: https://www.educba.com/ansible-inventory-file/. Luettu: 2.5.2023.

Freet, D., Agrawal, R., John, S. & Walker, J. 2015. Cloud Forensics Challenges from a Service Model Standpoint: IaaS, PaaS and SaaS. Luettavissa: https://doi.org/10.1145/2857218.2857253. Luettu: 22.05.2023.

GeeksForGeeks, s.a. Introduction to Linux Shell and Shell Scripting. Luettavissa: https://www.geeksforgeeks.org/introduction-linux-shell-shell-scripting/. Luettu: 21.5.2023.

Gil 2022. Cloud Pricing Comparison: AWS vs. Azure vs. Google Cloud Platform in 2022. Luettavissa: https://cast.ai/blog/cloud-pricing-comparison-aws-vs-azure-vs-google-cloud-platform/. Luettu: 12.3.2023.

Hutchins, E., Cloppert, M. & Amin, R. s.a. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Lockheed Martin Corporation. North Bethesda. Luettavissa: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf. Luettu: 12.5.2023.

Microsoft 2023. Luettavissa: https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3#scripting-language. Luettu: 21.5.2023.

Narula, S., Jain, A. & Prachi. 2015. Cloud Computing Securtity: Amazon Web Service. Luettavissa: DOI 10.1109/ACCT.2015.20. Luettu: 21.5.2023.

Noël, R. 2003. Global Information Assurance Certification Paper. SANS Institute. Yhdysvallat. Luettavissa: https://www.giac.org/paper/gsec/2730/building-security-test-environment/104653. Luettu: 12.03.2023.

Pawankumar,S. , Dash, B & Ansari, M. 2022. Anti-Phishing Techniques – A review of Cyber Defense Mechanisms, s.153-155. Luettavissa: https://www.researchgate.net/profile/Bibhu-Dash-5/publication/362143730_Anti-Phishing_Techniques_-A_Review_of_Cyber_Defense_Mechanisms/links/62d8661525155478d53ffd6f/Anti-Phishing-Techniques-A-Review-of-Cyber-Defense-Mechanisms.pdf. (myös DOI löytyy linkistä) Luettu: 5.5.2023.

Poston, H. 2019. Top 10 network recon tools. Luettavissa: https://resources.infosecinstitute.com/topic/top-10-network-recon-tools/. Luettu: 13.5.2023.

Ryder, T. 2018. Bash Quick Start Guide. Packt Publishing Limited. Birmingham. E-kirja. Luettu: 21.5.2023.

Salonen, K., Eloranta, S., Hautala, T. & Kinos, S. 2017. Kehittämistoiminta ja kehittämisen menetelmiä ammatillisessa korkeakoulutuksessa. Turun ammattikorkeakoulun oppimateriaaleja 108. Turku. Luettavissa: https://julkaisut.turkuamk.fi/isbn9789522166494.pdf. Luettu: 8.5.2023.

SentinelOne 2023a. What is the Cyber Kill Chain? Steps, Examples, & How to Use It. Luettavissa: https://www.sentinelone.com/cybersecurity-101/cyber-kill-chain/. Luettu: 12.5.2023.

SentinelOne 2023b. What is Lateral Movement? Definition & Examples. Luettavissa: https://www.sentinelone.com/cybersecurity-101/lateral-movement/. Luettu: 5.5.2023.

SSH 2023a. SSH Protocol - Secure Remote Login and File Transfer. Luettavissa: https://www.ssh.com/academy/ssh/protocol. Luettu: 2.5.2023.

SSH 2023b. Basic overview of SSH Keys. Luettavissa: https://www.ssh.com/academy/ssh-keys. Luettu: 2.5.2023.

Suse, 2023. On-Premises. Luettavissa: https://www.suse.com/suse-defines/definition/on-premises/. Luettu: 17.5.2023

Terraform 2023. Luettavissa: https://developer.hashicorp.com/terraform/intro. Luettu: 13.3.2023.

Terraform s.a.a. Terraform Language Documentation. Luettavissa: https://developer.hashicorp.com/terraform/language. Luettu: 25.4.2023.

Terraform s.a.b. Terraform Configuration Syntax. Luettavissa: https://developer.hashicorp.com/terraform/language/v1.3.x/syntax/configuration. Luettu: 25.4.2023.

Terraform s.a.c. The Core Terraform Workflow. Luettavissa: https://developer.hashicorp.com/terraform/intro/core-workflow. Luettu: 1.5.2023.

Terraform s.a.d. Modules. Luettavissa: https://developer.hashicorp.com/terraform/language/modules. Luettu: 5.5.2023.

Terraform s.a.e. Provider configuration. Luettavissa: https://developer.hashicorp.com/terraform/language/providers/configuration. Luettu: 9.5.2023.

Valtionvarainministeriö. 2020. Tuottavuutta pilvipalveluilla – Ohje julkisen hallinnon pilvipalvelujen hyödyntämiseen. Valtiovarainministeriön julkaisuja 2020:66. Valtiovarainministeriö. Helsinki. Luettavissa:https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/162451/VM_2020_66.pdf?sequence=4. Luettu: 13.3.2023.

Yadav, T. & Mallari, R. 2015. Technical aspects of Cyber Kill Chain. Luettavissa: https://arxiv.org/pdf/1606.03184.pdf. Luettu: 6.5.2023.

Yamin, M., Katt, B. & Gkioulos, V. 2019. Cyber Ranges and Security Testbeds: Scenarios, Functions, Tools and Architecture, Computers & Security. Elektroninen tietoaineisto, Luettavissa: https://doi.org/10.1016/j.cose.2019.101636. Luettu: 5.5.2023.

Zeng, W & Germanos V. 2019. Modelling Hybrid Cyber Kill Chain. Luettavissa: https://ceur-ws.org/Vol-2424/paper10.pdf. Luettu: 5.5.2023.

Liitteet