Anttoni Tapio

AUTOMAATION HYÖDYNTÄMINEN JA KEHITYS MICROSOFT SENTINEL -PALVELUSSA

Opinnäytetyö

Tekniikan ammattikorkeakoulututkinto

Kyberturvallisuuden koulutus

2024

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä	Anttoni Tapio
Työn nimi	Automaation hyödyntäminen ja kehitys Microsoft Sentinel -palvelussa
Toimeksiantaja	CGI Suomi Oy
Vuosi	2024
Sivut	62 sivua, liitteitä 10 sivua
Työn ohjaajat	Kimmo Kääriäinen, Jarkko Hänninen

TIIVISTELMÄ

Kyberuhkien jatkuvasti kehittyessä kyberturvallisuusvalvomot (Security Operations Center, SOC) kohtaavat yhä suuremmissa määrin yhä haastavampia hyökkäyksiä, jotka vaativat nopeita, tarkkoja ja skaalautuvia vastatoimia. Uhkien manuaalinen käsittely lisää uhan mitigoimiseen kuluvaa aikaa, sekä kasvattaa inhimillisten virheiden tuomaa riskiä. Microsoft Sentinelin kaltaisilla Security Orchestration, Automation, and Response- eli SOAR-alustoilla hyödynnettävät automaattiset vastatoimenpiteet tarjoavat ratkaisuja näihin ongelmiin, ja ovat näin ollen kriittinen osa modernin SOCin toimintaa.

Tämän opinnäytetyön tavoitteena oli kehittää toimeksiantajan kypsyysastetta hyödyntää automaatiota hälytysten käsittelyssä Microsoft Sentinel -palvelussa. Tavoitteen saavuttamiseksi luotiin ohjekirja Sentinelin automaatioratkaisujen kehitykselle.

Työssä käytettiin interventionistista tutkimusotetta. Primääriaineisto kerättiin kenttäpäiväkirjaan kirjatuilla havainnoilla automaatioratkaisujen kehityksestä ja testauksesta, jonka pohjalta ohjekirja luotiin. Kenttäpäiväkirjan sisältö analysoitiin laadullisesti, ja analyysin pohjalta saatiin koostettua ohjekirjaan tutkimuksen tavoitteita vastaava sisältö.

Tutkimuksen tuloksena luotiin ohjekirja, jonka sisältö koostuu pääosin läpikäyntiohjeista sekä niitä tukevasta teoriasta. Ohjekirjan läpikäyntiohjeissa kuvatut ratkaisut testattiin käytännössä tutkimuksen aikana, ja siten saatiin taattua niiden luotettavuus. Toimeksiantajan henkilöstö voi jatkossa hyödyntää ohjekirjaa kehittäessään Sentinelin automaatioratkaisuja asiakasympäristöissä.

Asiasanat: automaatio, kyberturvallisuus, Logic Apps, ohjekirja, Sentinel

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Engineering
Author	Anttoni Tapio
Thesis title	Utilizing and developing automation in Microsoft Sentinel
Commissioned by	CGI Suomi Oy
Time	2024
Pages	62 pages, 10 pages of appendices
Supervisors	Kimmo Kääriäinen, Jarkko Hänninen

ABSTRACT

As cyber threats continue to evolve, Security Operations Centers (SOC) are increasingly faced with more complex attacks that require rapid, accurate, and scalable responses. Handling threats manually prolongs the time needed for mitigation and increases the risk of human error. Automated response measures enabled by Security Orchestration, Automation, and Response (SOAR) platforms, such as Microsoft Sentinel, provide solutions to these challenges and are, thus, a critical component of modern SOC operations.

The objective of this thesis was to enhance the commissioning organization's maturity in utilizing automation for handling alerts in Microsoft Sentinel. To achieve this objective, a guidebook was created for the development of automation solutions in Sentinel.

An interventionist research approach was used for the study. The primary data was collected through observations recorded in a field diary on the development and testing of automation solutions, which served as the basis for creating the guidebook. The content of the field diary was analyzed qualitatively, and the analysis formed the basis for compiling the guidebook's content to meet the study's objectives.

The outcome of the study was a guidebook which consists of step-by-step instructions supported by explanations of concepts. The solutions described in the guidebook's walkthrough instructions were practically tested during the study, ensuring their reliability. The commissioning organization's staff can use the guidebook in the future when developing Sentinel automation solutions for customer environments.

Keywords: automation, cyber security, guidebook, Logic Apps, Sentinel

SISÄLLYS

1 JOHDANTO

Kyberturvallisuusvalvomoiden (Security Operations Center, SOC) kohtaamiin yleisimpiin ja suurimpiin haasteisiin kuuluu muun muassa resurssipula, hälytysten ylikuormittavuus, sekä automaation puute (Kasa 2024). Hyökkäysten kehittyminen ja määrällinen kasvu vaatii edistyneempien automaatiokykyjen hyödyntämistä mahdollisimman nopean ja tehokkaan uhkien havaitsemisen ja torjunnan saavuttamiseksi, ja jotta hälytysten käsittelyn tuomaa taakkaa saataisiin kevennettyä SOC-analyytikoilta. Tämän myötä SOCien on syytä turvautua yhä enemmän Security Orchestration, Automation, and Response- eli SOAR-alustojen mahdollistamiin automaatioihin uhkien havaitsemisessa ja torjunnassa.

Tämän opinnäytetyön aiheena on kehittää toimeksiantajalle ohjekirja automaation hyödyntämiseen hälytysten käsittelyssä Microsoft Sentinel -palvelussa. Ohjekirjassa kuvaillaan, kuinka automaatioratkaisujen kehitys ja implementointi palvelussa voidaan toteuttaa, sekä demonstroidaan tämä läpikäyntiohjeiden avulla. Ohjeissa käytettyjä esimerkkiratkaisuja ei sellaisenaan tarkoiteta implementoitavaksi suoraan asiakasympäristöihin, vaan niiden tarkoituksena on ainoastaan havainnollistaa kehityksen prosessit sekä osoittaa mahdollisuuksia kehitettäviin ratkaisuihin. Ohjekirjaa toimeksiantajan työntekijät voivat hyödyntää erilaisten automaatioratkaisujen käytännön implementoinnissa todellisiin asiakasympäristöihin.

Tavoitteena koko työllä on, että ohjekirjan avulla Microsoft Sentinelin automaation hyödyntämisen kypsyysaste toimeksiantajalla kasvaa, jonka myötä voidaan saavuttaa nopeammat reagointiajat, minimoida inhimilliset virheet hälytysten käsittelyssä, sekä kohdata kyberuhkat ennakoivammin.

Työn toimeksiantaja on CGI Suomi Oy, joka on yksi suurimmista Suomessa toimivista IT- ja liiketoimintakonsultoinnin palveluyhtiöistä. Yhtiö kuuluu Kanadassa vuonna 1976 perustettuun CGI:n kansainväliseen yritykseen, joka tuottaa IT-palveluita maailmanlaajuisesti sekä julkisen että yksityisen sektorin toimijoille. (CGI 2024a.) Työn toimeksiannon ja toteutuksen tarkka sijainti yhtiössä on sen kyberturvallisuusvalvomo eli Security Operations Center

(SOC), joka on osa CGI Suomi Oy:n kyberturvallisuuden yksikköä tarjoten asiakasyrityksille IT ja OT-ympäristöjen valvontapalveluja (CGI 2024b).

2 TUTKIMUSASETELMA

Tutkimusasetelmassa tarkastellaan tutkittavan työn aiheellisuutta selvittämällä sekä toimeksiantajalta että toimialalta yleisesti, kuinka relevantti aiheenomainen ongelma on; onko sen ratkomista lähtökohtaisesti hyödyllistä ja mielekästä lähteä yrittämään. Tämä voidaan selvittää tarkastelemalla aiempia tutkimuksia alalta, ja löytää sieltä yhteiset ongelmakohdat toimeksiantajan kokemien haasteiden kanssa. Tällä lähestymisellä voidaan puolestaan muodostaa sopiva tutkimusongelma, sitä tukevat tutkimuskysymykset sekä menetelmä sen tutkimiselle.

2.1 Tutkimusongelma

Tämän opinnäytetyön tutkimusongelmana on automaation hyödyntämisen puute Microsoft Sentinelin hälytysten käsittelyssä SOCissa. Automaation hyödyntämisen puute johtaa moniin hankaluuksiin SOCin toiminnassa, niin toimeksiantajalla kuin yleisestikin toimialalla. Näistä hankaluuksista kenties yleisin on hälytysväsymys, joka tarkoittaa käytännössä SOC-analyytikoiden ylikuormittumista suuren hälytysmäärän johdosta, joista valtaosa on turhia.

SOCille kaikista tärkeintä on tunnistaa uhat ja reagoida niihin nopeasti. Automatisoinnin puute johtaa usein monella tavalla tässä tehtävässä epäonnistumiseen. Microsoft Sentinel SOAR-toiminnallisuuksillaan tarjoaa mahdollisuuksia näiden haasteiden ylittämiseen automaatioratkaisuilla. Sentinelissä voidaan automaatiolla esimerkiksi korreloida eri lähteistä tulevaa kyberuhkatietoa keskenään, suorittaa ensianalyysiä, tehdä vastatoimia ja rikastaa hälytysten tuomaa informaatiota. Sentinelin tarjoamien ratkaisujen optimoidulla hyödyntämisellä voidaan lähteä ratkomaan tutkimusongelmana olevaa automaation puutetta hälytysten käsittelyssä.

2.2 Tutkimuskysymykset

Tutkimuskysymysten avulla tutkimusongelmaa saadaan kohdistettua, ja sen ratkaisuun määrittyy konkreettisia tavoitteita, joita kohti lähteä työn

toteutuksessa. Tässä tapauksessa työn tuotoksena tehdään ohjekirja, joten tutkimuskysymyksiä voidaan asettaa sen valossa: mihin kysymyksiin ohjekirjan tulisi pystyä vastaamaan, jotta siitä saataisiin mahdollisimman suuri hyöty irti tutkimusongelman ratkaisemiseksi. Tutkimuskysymykset ovat:

- Miten Microsoft Sentinelin automaatioratkaisuja hälytysten käsittelyyn voidaan kehittää?
- Millaisia automaatioratkaisuja hälytysten käsittelyyn Microsoft Sentinelissä voidaan kehittää?
- Kuinka Microsoft Sentinelin automaatioratkaisujen kehittämisen tietotaitoa voidaan kasvattaa SOCissa?

2.3 Tutkimusote

Tämä opinnäytetyö käyttää interventionistista tutkimusotetta, jossa tavoitteena on saavuttaa positiivinen muutos toimeksiantajan organisaatiossa. Kyseessä on konstruktiivinen tutkimus, joka tähtää ratkaisun kehittämiseen toimeksiantajan kokemaan käytännön ongelmaan. Työn tuotoksena luodaan ohjekirja toimeksiantajan, eli CGI Suomi Oy:n SOC:in työntekijöille, jotka voivat ohjekirjaa hyödyntäen implementoida automaatioratkaisuja asiakasympäristöihin jatkossa parantaen automaation hyödyntämisen kypsyysastetta SOCissa. Ohjekirjaan tulee ohjeistus Sentinelin automaatiosääntöjen ja playbookien luomiselle läpikäyntiohjeita hyödyntäen.

Läpikäyntiohjeissa käytetään esimerkkiratkaisuja, joiden tavoitteena on olla yksinkertaisia, mutta riittävän kattavia demonstroimaan olennaiset toiminnallisuudet. Ratkaisut kehitetään ja testataan työn aikana, ja niiden toteutukseen järjestetään tarvittava testiympäristö testiresursseineen Sentinelissä ja mahdollisesti muissa tarvittavissa Azuren palveluissa. Työssä kehitettävien ratkaisujen tavoitteena ei ole olla sellaisenaan toteutettavia ratkaisuja asiakasympäristöihin, vaan niiden päätavoitteena on havainnollistaminen, jotta ohjekirjan käyttäjät oppisivat oikeita ratkaisuja kehittämään. Työn suunnittelun alkuvaiheessa harkittiin myös enemmän kehittämistutkimuksen kaltaista lähestymistä, jolloin työ olisi keskittynyt nimenomaan uusien tuotantoon pantavien automaatioratkaisujen kehitykseen yleistettävän ohjekirjan sijaan, mutta suunnittelun edetessä todettiin, että ohjekirjan kaltainen konstruktiivisempi lähestyminen soveltuu työhön paremmin. Ohjekirjan hyöty tulee sen skaalautuvuudesta, kun Sentinelin automaation kehittämisen tietotaitoa saadaan levitettyä laajemmalti

toimeksiantajan yksikössä. Toinen hyvä syy tähän valintaan on, että tuotantoon tarkoitettujen spesifien automaatioratkaisujen kehityksen dokumentointi voisi olla liian sensitiivistä tietoa julkaistavaksi opinnäytetyöksi.

2.4 Aineistonkeruu

Primääriaineisto tässä tutkimuksessa tulee koostumaan automaatioratkaisujen kehitystä havainnoivasta kenttäpäiväkirjasta, sekä sen pohjalta luodusta ohjekirjasta. Ratkaisujen kehitys ja siihen liittyvä testaaminen havainnoidaan kenttäpäiväkirjan avulla, johon merkitään kehitykseen liittyviä ideoita, niiden testaamisen tuloksia, testauksessa ilmeneviä ongelmakohtia, sekä muita havaintoja. Testaamisen myötä saadaan kerättyä ohjekirjaan toimivaksi todetut ohjeet.

Automaatioratkaisut kehitetään ja testataan toimeksiantajan omistamassa Azure-ympäristössä, joka sisältää Sentinelin ja on tarkoitettu toiminnallisuuksien testaamiseen. Lähtökohtaisena suunnitelmana on luoda ympäristöön ainakin seuraavat resurssit testaamista varten:

- Testikäyttäjä
- Hälytyssääntö
- Playbook
- Automaatiosääntö

Työn aikana voi ilmetä tarvetta muunkinlaisten resurssien luomiselle. Näille edellä mainituille resursseille on kuitenkin tarve nähtävissä jo ennalta. Läpikäyntiohje, joka ohjekirjan havainnollistamiseksi luodaan, tulee todennäköisesti sisältämään seuraavanlaisen käyttötapauksen:

1. Testikäyttäjä kirjautuu sisään
2. Kirjautuminen laukaisee hälytyksen
3. Hälytys laukaisee automaatiosäännön
4. Automaatiosääntö laukaisee playbookin
5. Playbook tekee toimenpiteitä kohdistuen testikäyttäjään ja mahdollisesti muihinkin resursseihin, kuten hälytyksen tilaan

Kuten luotavat resurssit, myöskin edellä mainittu käyttötapaus voi testaamisen aikana kokea täydennystä/muutoksia. Tavoitteena on saada aikaiseksi relevantteja toimintoja tarpeeksi demonstroiva, mutta myöskin sopivan yksinkertainen käyttötapaus pitäen mielessä havainnollistamisen tarkoituksen.

2.5 Aineiston analyysi

Työn tuotoksena syntyvä primääriaineisto tullaan analysoimaan laadullisesti. Kenttäpäiväkirjan sisältö analysoidaan sisällönanalyysin menetelmiä soveltaen. Tässä käytetään aineistolähtöistä lähestymistä, jossa luokitellaan analyysivaiheen aikana aineistosta tutkimusongelman ja -kysymysten kannalta sopivat kategoriat (Elo ym. 2022, 218). Kategorioista puolestaan pyritään tunnistamaan teemoittelemalla toistuvia piirteitä (Kallinen & Kinnunen 2021). Kun teemoittelu on tehty, luodaan tunnistettujen teemojen myötä ohjekirjan sisältö. Lopuksi arvioidaan, kuinka hyvin ohjekirja vastaa tutkimuskysymyksiin, ja onnistutaanko sen avulla ratkaisemaan tutkimusongelma.

2.6 Aiemmat tutkimukset ja kyselyt

Opinnäytetyön aiheen kuuluisi olla relevantti sekä työn toimeksiantajalle että kyberturvallisuuden alalle. Paneutumalla aiempiin aihetta koskeviin tutkimuksiin ja kyselyihin voidaan kartoittaa aiheen lisätutkimuksen tarvetta sekä löytää pinnalla olevat ongelmakohdat, jotka yhdistyvät sekä alalla yleisesti, että toimeksiantajalla.

Tietoturvatyökalujen toimittajilla on tapana optimoida työkalunsa ei väärien hälytysten, vaan huomaamatta jääneiden uhkien mukaan (Tilbury, Flowerday 2024a, 1). Tämä johtaa herkästi siihen, että käytettävät työkalut puskevat analyytikoiden käsiteltäväksi suunnattoman määrän hälytyksiä pienistäkin indikaattoreista, jottei vaan huomaamaton uhka jää ainakaan työkalun toimittajan vastuulle. Suuri turhien hälytysten määrä aiheuttaa hälytysväsymystä, joka on yksi tunnetuimmista SOCien kohtaamista haasteista. Tilburyn ja Flowerdayn mukaan (2024b, 3.1) SOCien keskeiset automaatiota edellyttävät haasteet ovat jaettavissa neljään kategoriaan: kyberuhkatiedon puute, uhkien tunnistamisen tehottomuus, hälytysväsymys, sekä hälytysten sisältämän informaation vaikeaselkoisuus. Tilburyn ja Flowerdayn analysoimista artikkeleista suurin osa, jopa 63 %, mainitsi hälytysväsymyksen (2024b, 3.1.3).

Hälytysväsymys on yksi iso manuaalisen käsittelyn haasteita, joihin toki sisältyy muitakin työntekijöitä kuormittavia tekijöitä. Vuonna 2023 Tinesin 900

SOC-ammattilaisen kyselyssä avainhavainnoiksi todettiin, että 63 % SOC-ammattilaisista kokee loppuunpalamisen tunnetta jollain tasolla. Yli puolet vastaajista vastasi, että tulevat todennäköisesti vaihtamaan työpaikkaansa seuraavan vuoden aikana. Manuaalinen työ todettiin kaikista turhauttavimmaksi osuudeksi työssä, ja jopa 93 % prosenttia vastaajista totesi, että automaatio voisi parantaa heidän työn ja vapaa-ajan välistä tasapainoa tekemällä työstään tehokkaampaa. (Hinchy 2023.)

SANSin vuoden 2024 SOC-kyselyssä kysyttiin 403:lta SOC-ammattilaiselta ympäri maailmaa suurinta estettä SOCin täyden hyödyntämisen saavuttamiseksi organisaatiossa. Suosituimmaksi yksittäiseksi vastaukseksi osoittautui automaation ja orkestroinnin puute, jonka vastasi 18,3 % vastaajista. (Crowley 2024.)

Automaatio ja sen kehitys siis selkeästi koetaan tarpeelliseksi, ja tämä on todettu myös opinnäytetyön toimeksiantajalla. Toisaalta myös liiallinen nojaaminen automaattisiin työkaluihin väärissä kohdissa ja väärin tavoin on riski, ja voi johtaa kriittisten hälytysten huomioimatta jättämiseen (Tilbury, Flowerday 2024a, 1). Tämä on erittäin tärkeää huomioida automaattisten ratkaisujen implementoinnissa myöskin.

3 KESKEISIÄ LYHENTEITÄ JA NIMIÄ

Tässä luvussa esitellään keskeisiä lyhenteitä ja nimiä liittyen työssä käytettävään toimintaympäristöön.

3.1 SOC

Security Operations Center (SOC) eli suomalaisittain kyberturvallisuusvalvomo on yksikkö, jonka tehtävänä on valvoa tietoturvapoikkeamia reaaliajassa sekä reagoida niihin vastatoimenpiteillä. Yrityksillä voi olla ympäristönsä tarkkailuun oma SOC, tai ulkoistettu palvelu siihen. Tämän työn toimeksiantajan CGI:n SOC tarjoaa ulkoistettua kybervalvontapalvelua asiakasorganisaatioille sekä yksityisellä että julkisella sektorilla (CGI 2024b).

SOCien henkilöstörakenne vaihtelee riippuen yksiköstä, mutta yleisesti ottaen SOCiin lukeutuvia rooleja ovat yksikön manageri, analyytikot, kehittäjät ja arkkitehdit sekä ulkoistettua palvelua tarjoavilla yksiköillä myöskin asiakkuuden rajapinnassa toimivat palvelupäälliköt.

Analyytikot usein jakautuvat eri tasoihin, ja tavallisen mallin mukaan näitä tasoja on kolme. Ensimmäisen tason analyytikot pääosin käsittelevät tulevia hälytyksiä, tekee niistä ensimmäiset analyysit sekä kevyet vastatoimet. Tarpeen vaatiessa ykköstason analyytikko voi eskaloida tapauksen kakkostason analyytikolle, jonka tehtävänä on hoitaa edistyksellisempää analyysia ja pikaista reagointia vaativia kriittisiä tapauksia. Kolmannella tasolla puolestaan tehdään sekä ennakoivaa uhkametsästystä tapauksista, jotka eivät vielä ole ehtineet hälytyksen aiheuttavan tietoturvapoikkeaman tasolle, kuin myös syvää forensiikkaa jo tapahtuneista ja käsitellyistä poikkeamista. Kaikki SOCit eivät kuitenkaan noudata tämän mallin mukaisia tasoja, eikä välttämättä tasoihin perustuvaa rakennetta ollenkaan, vaan roolit voivat jakautua myös muilla tavoin.

3.2 SIEM

Security Information and Event Management (SIEM) on järjestelmä, joka kerää ja analysoi tapahtumalokitietoja useista lähteistä ja havaitsee poikkeamat normaalista toiminnasta. SIEM-teknologian työkalut kokoavat tietoja sovelluksista, laitteista ja käyttäjistä, sekä luovat niistä hälytyksiä määriteltyjen sääntöjen avulla. SIEM tarjoaa keskitetyn paikan loki- ja hälytystietojen käsittelyyn ja analysointiin, ja on hyvin yleinen SOCien käyttämä työkalu. (Microsoft Security 2024a.)

3.3 SOAR

Security Orchestration, Automation, and Response (SOAR) on alusta, jonka työkaluilla voidaan automatisoida kyberuhkien havaitsemiseen, estämiseen ja käsittelyyn liittyviä toimia. SOAR mahdollistaa tietoturvatiimien tehokkaamman toiminnan vähentämällä manuaalista työtä ja nopeuttamalla uhkien käsittelyä. Teknologia koostuu orkestroinnista, automatisoinnista ja tapausten käsittelystä, jotka yhdessä parantavat prosessien tarkkuutta ja nopeutta. SOAR-teknologia eroaa SIEM-järjestelmistä, jotka keskittyvät tietojen

keräämiseen ja analysointiin, kun taas SOAR toimii näiden tietojen pohjalta automatisoiden uhkien käsittelyn. SOAR- ja SIEM-teknologiat toimivat parhaiten yhdessä, jolloin saavutetaan parempi näkyvyys uhkiin ja tehokkuus niiden torjunnassa. (Microsoft Security 2024b.)

3.4 Azure

Azure on Microsoftin tarjoama pilvipalvelualusta. Se tarjoaa laajan valikoiman työkaluja ja palveluita, joiden avulla yritykset ja kehittäjät voivat rakentaa ja hallita resursseja Microsoftin maailmanlaajuisen datakeskusverkoston kautta. Azure tukee monia ohjelmointikieliä, kehitysalustoja ja käyttöjärjestelmiä. Sen avulla käyttäjät voivat hyödyntää palveluita, kuten virtuaalikoneita, tallennustilaa, tietokantoja ja verkkoresursseja, kuin myös tekoälyä, koneoppimista ja analytiikkaa. Organisaatiot voivat skaalata resursseja tarpeidensa mukaan, mikä tekee Azuren käytöstä mukautuvaa eri työkuormiin. Azuren integraatio Microsoftin muiden tuotteiden, kuten Office 365:n ja Active Directoryn kanssa tekee siitä kattavan ratkaisun yrityksille, jotka haluavat hallita infrastruktuuriaan, kehittää sovelluksia tai käsitellä tietoja pilvessä. Azure mahdollistaa myös hybridipilviratkaisut, joissa yhdistetään paikallisia ja pilviresursseja. Azure on hyvin tietoturvaan sonnustautunut pilvipalvelualusta, ja siihen on integroitu monia tietoturvatyökaluja. (Microsoft Azure 2024.)

3.5 Entra ID

Entra ID, aiemmin tunnettu nimellä Azure Active Directory (Azure AD), on Microsoftin pilvipohjainen identiteetin ja pääsynhallinnan (IAM) ratkaisu. Sillä hallitaan käyttäjäidentiteettejä, laitteita ja sovelluksia pilvi- tai hybridiympäristöissä. Entra ID:ssä voidaan esimerkiksi luoda ja hallinnoida käyttäjätilejä, ja se tukee turvallisia tunnistautumismenetelmiä, kuten monivaiheista todennusta (MFA) ja salasanattomia metodeja, kuten FIDO2-avaimia. Se mahdollistaa pääsynhallintapolitiikkojen asettamisen, jotta käyttäjät voivat käyttää sovelluksia ja resursseja vain tietyissä olosuhteissa, kuten tietyiltä laitteilta tai sijainneista (Conditional Access). Entra ID integroituu moniin Azuren pilven SaaS-sovelluksiin, kuten Microsoft 365, ja se sisältää myös toimintoja käyttöoikeuksien hallintaan sekä roolipohjaiseen pääsynhallintaan (Role-Based Access Control, RBAC), jotta käyttäjillä on sopivat käyttöoikeudet tarvitsemiinsa tehtäviin. (Microsoft Entra ID 2024.)

Tietoturvallisuus on keskeinen osa Entra ID:tä, sillä usein monet tietoturvauhkat liittyvät jollain tapaa identiteetteihin ja pääsynhallintaan, ja tässä rajapinnassa murtotapaukset usein ensimmäisenä havaitaan. Entra ID:ssä on tietoturvapoikkeamia valvova kokonaisuus nimeltä Entra ID Protection, joka riskipohjaisilla algoritmeilla auttaa havaitsemaan ja reagoimaan identiteettiin liittyviin uhkiin. Sen saa integroitumaan Sentineliin, ja se on tärkeä osa tietoturvavalvonnan kokonaisuutta Microsoftin tuotteistossa.

3.6 Sentinel

Sentinel on pilvipohjainen SIEM- ja SOAR-alusta, joka integroi dataa useista lähteistä, kuten identiteeteistä, sovelluksista, laitteista ja verkoista, tarjoten keskitetyn paikan tietoturvapoikkeamien hallinnalle ja käsittelylle. Sentinel kerää tietoja niin pilvi-, kuin paikallisista järjestelmistä, sekä kolmansien osapuolten palveluista, ja korreloi tapahtumia eri lähteistä tunnistaakseen mahdolliset uhat. Siihen on suoraan integroitu muita Azuren palveluita, kuten Log Analytics ja Logic Apps, joita voidaan hyödyntää tietoturvatapahtumien tutkimisessa ja niihin reagoimisessa. Alusta luo automaattisesti hälytyksiä ja muodostaa niistä analytiikkaa käyttäen niin sanottuja tapahtumia (incident), joihin kerätään tiedot samaan tapahtumaan liittyvistä hälytyksistä. Sentinel tarjoaa myös mahdollisuuden luoda itse omia niin kutsuttuja analytiikkasääntöjä KQL-hakukielen avulla (Kusto Query Language), joiden pohjalta myös voidaan luoda tapahtumia. SOAR-ominaisuuksilla varustettuna Sentinelissä voidaan automatisoida monia tietoturvatapahtumien hallitsemiseen ja käsittelyyn liittyviä prosesseja, kuten vastatoimenpiteitä ensireaktion nopeuttamiseksi. (Microsoft Sentinel 2024d.)

3.7 Logic Apps ja Sentinelin playbookit

Logic Apps on Azuren tarjoama PaaS (Platform as a Service) eli pilvipalvelualusta, joka mahdollistaa erilaisten prosessien automaation ilman, että tarvitaan laajaa koodausosaamista. Siinä käytetään graafista käyttöliittymää, jonka avulla voidaan kehittää ja automatisoida erilaisia prosesseja sekä integraatioita pilvessä ja paikallisessa ympäristössä. Logic Apps pohjautuu JSON- eli JavaScript Object Notation -formaatin koodiin, jota

voi myös käyttää näkymänä kehityksessä graafisen käyttöliittymän sijaan. (Azure Logic Apps 2024d.)

Logic Appin prosessit seuraavat JSON-pohjaista kieltä nimeltä Workflow Definition Language (WDL), joka määrittää sen sisältämien toimintojen järjestyksen suorituksen aikana (Azure Logic Apps 2024c). Logic App koostuu eri vaiheista: alussa on laukaisin (trigger), eli toiminto, joka käynnistää Logic Appin suorituksen. Esimerkiksi uusi hälytys Sentinelissä voi toimia laukaisimena. Tätä seuraa muut toiminnot, jotka suoritetaan Logic Appin aikana, kuten vaikkapa jonkin vastatoimen suorittaminen ja/tai sähköpostin lähettäminen. Toimintojen kanssa voidaan hyödyntää erilaisia ohjelmoinnista tuttuja toimia, kuten ehtoja ja silmukoita. (Azure Logic Apps 2024d.)

Logic Apps on keskeinen komponentti Sentinelissä automaattisten toimien osalta. Sentinelissä voidaan luoda niin sanottuja playbookeja, jotka käytännössä ovat Logic Appeja, jotka käyttävät Sentinelille luotua yhteystyyppiä (Microsoft Sentinel 2024b). Tällä yhdistelmällä saadaan tehtyä automaatioketjuja, jotka käynnistyvät tiettyjen ehtojen täyttyessä, kuten tietynlaisen hälytyksen tapahtuessa. Käyttötapauksia tälle ovat esimerkiksi hälytysten käsittely ja eskalointi, uhkien vastatoimet, ilmoitukset sähköpostiin tai muihin integroituihin kanaviin, sekä hälytystietojen rikastaminen.

3.8 Managed identity

Managed identity eli hallinnoitu identiteetti on Azuren automaattisesti luoma identiteetti, jota voidaan käyttää autentikoinnissa Azuren resurssien kuten Logic Appien kanssa. Hallinnoitua identiteettiä käyttämällä ei tarvitse erikseen pitää tallessa tunnuksia missään koodissa tai avainten hallintapalvelussa, vaan tunnusten hallinnointi ja niillä autentikointi hoidetaan automaattisesti taustalla Azuren toimesta. (Microsoft Entra 2024b.) Hallinnoidun identiteetin käyttäminen on kätevä ja turvallinen tapa hoitaa etenkin Logic Appin autentikointi, ja sitä tulisi suosia käyttäjätunnusten tai manuaalisesti hallinnoitavien sovellustunnusten (Service Principal) sijaan. Hallinnoituja identiteettejä on kahdenlaisia: järjestelmän luoma (system-assigned) ja käyttäjän luoma (user-assigned). On tapauskohtaista, kumman käyttö on kannattavampaa. Järjestelmän luoma hallinnoitu identiteetti on aina

resurssikohtainen; se luodaan resurssin luomisen yhteydessä ja se myös poistuu kun kyseinen resurssi poistetaan. Esimerkiksi yksittäinen Logic App voi olla tällainen resurssi, jolle luodaan Logic Appin luomisen myötä järjestelmän luoma hallinnoitu identiteetti. Puolestaan käyttäjän luoma on erikseen luotava Azure-resurssi, jota voidaan käyttää autentikoinnissa muiden resurssien kanssa. Käyttötapauksena voisi olla vaikkapa yksi käyttäjän luoma hallinnoitu identiteetti, jota joukko Logic Appeja käyttää. (Microsoft Entra 2024b.)

Hallinnoitu identiteetti luo sovellustunnuksen Logic Appille, ja kyseiselle sovellustunnukselle voidaan myöntää oikeuksia tarvituille toiminnoille. Sovellustunnuksia voi toki käyttää ilman hallinnoitua identiteettiäkin, mutta silloin sen Client ID ja Client Secret täytyy säilyttää jossakin tallessa. Hallinnoidun identiteetin kanssa tätä ei tarvitse tehdä.

4 AUTOMAATION KEHITYS JA TESTAUS

4.1 Alkuvalmistelut

Aluksi täytyi suunnitella, että millaista käyttötapausta työssä lähdettäisiin kehittämään ja testaamaan. Tavoitteena oli keksiä yksinkertainen, mutta riittävän demonstroiva käyttötapaus, joka esittelisi keskeisiä automaatiossa käytettäviä toimia. Alustavan suunnitelman mukaan käyttötapaus sisältäisi ainakin seuraavat resurssit:

- Testikäyttäjä
- Hälytyssääntö
- Playbook (toisin sanoen Logic App)
- Automaatiosääntö

Käyttötapauksen rakenne puolestaan tulisi olemaan:

1. Testikäyttäjä kirjautuu sisään
2. Kirjautuminen laukaisee hälytyksen
3. Hälytys laukaisee automaatiosäännön
4. Automaatiosääntö laukaisee playbookin
5. Playbook tekee toimenpiteitä kohdistuen testikäyttäjään ja mahdollisesti muihinkin resursseihin, kuten hälytyksen statukseen

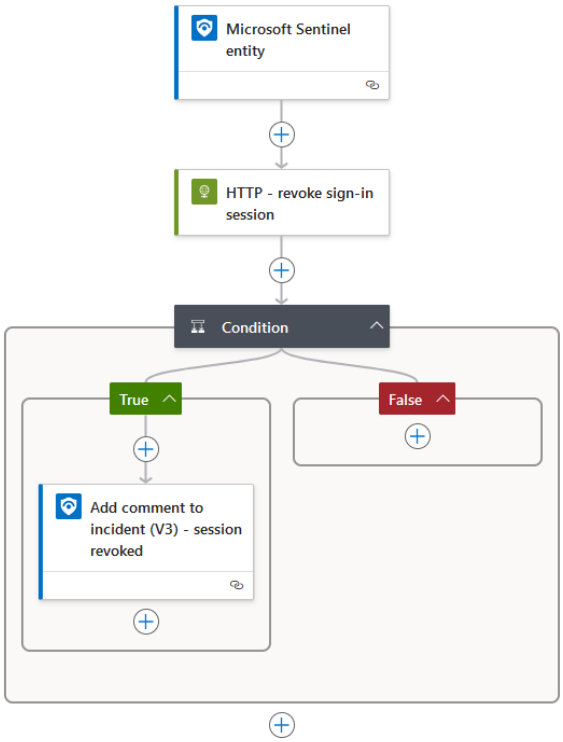
Näiden alustavien suunnitelmien pohjalta pystyi lähteä suunnittelemaan, että mitä alkuvalmisteluja tarvittaisiin. Ensimmäiseksi tarvittiin pääsy Sentineliin,

jota kehitystyössä voisi käyttää sekä riittävät oikeudet sinne. Toimeksiantajalla oli jo valmiiksi ennen työn alkamista olemassa oma Azure-ympäristönsä testaustarkoitukseen, joka on eristettynä tuotantoympäristöistä, ja jossa oli myös Sentinel pystytettynä. Koska kyseessä oli eristetty testiympäristö, niin sinne sai ylläpitäjiltä korkeimmat oikeudet kehitys- ja testaustyötä varten, sekä Azuren että Entran built-in roolien osalta. Sen myötä pääsi luomaan myöskin testikäyttäjän. Näin ollen ensimmäinen askel oli kunnossa.

Todettiin, että muut alustavassa suunnitelmassa määritellyt resurssit voidaan luoda myöhemmin itse kehitystyön aikana, joten voitiin siirtyä käyttötapauksen jatkosuunnitteluun. Täytyi päättää, että mitä toimenpiteitä playbook tekisi kohdistuen testikäyttäjään. Kaksi SOCissa keskeistä loppukäyttäjään kohdistuvaa vastatoimenpidettä ovat istuntojen katkaisu sekä salasanan vaihdon pakotus, joten nämä valittiin kehitettäviksi toimiksi.

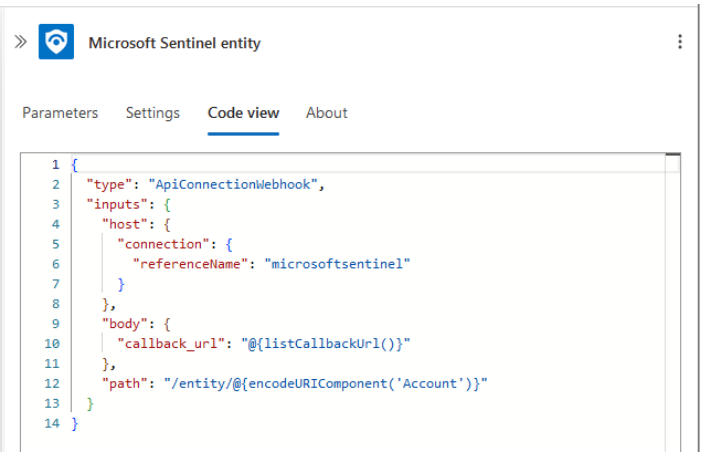
4.2 Istuntojen katkaisu

Sentinelin playbookeja voi luoda ylätasolla parilla eri tavalla: joko valmiista mallista, tai niin sanotusti tyhjästä. Tämä työ lähti liikkeelle Sentinelin tarjoamiin valmiisiin malleihin perehtymällä. Ensin luotiin istuntojen katkaisuun tarkoitettu playbook nimeltä ”Revoke Entra ID Sign-in session using entity trigger” (kuva 1) ja perehdyttiin sen sisältöön Microsoftin dokumentaatiota apuna käyttäen.



Kuva 1. Playbookin "Revoke Entra ID Sign-in session using entity trigger" sisältö

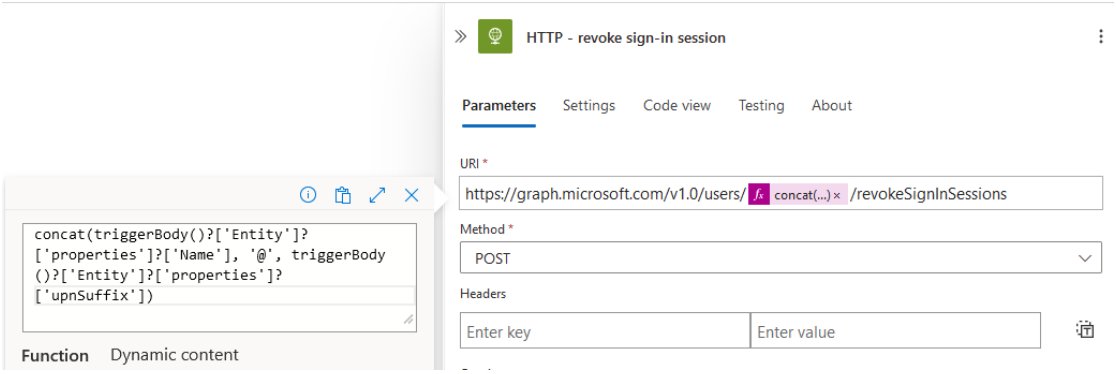
Playbookin ensimmäisenä vaiheena oli laukaisin, joka käynnistää playbookin suorituksen saadessaan datan Sentinelin entiteetistä. Kuvassa 2 on näytetty kyseinen toiminto JSON-koodina, josta näkee, että se on käytännössä webhook-tyypin rajapinta, jonka uniikkia niin sanottua ”callback URL” -osoitetta kutsumalla voidaan laukaista tämän playbookin suoritus. Siihen on määritelty, että data vastaanotetaan nimenomaan Account- eli käyttäjätyypin entiteeteistä.



Kuva 2. Entiteettityypin laukaisin JSON-koodina

Seuraavana vaiheena playbookissa oli istuntojen katkaisu Entra ID:ssä. Tämä tehdään API-kutsulla Microsoft Graphiin, joka tarjoaa pääsyn M365-palveluiden dataan (Microsoft Graph 2024a).

Kyseinen API-kutsu tehdään POST-metodilla päätepisteeseen ”https://graph.microsoft.com/v1.0/users/<concat(...)>/revokeSignInSessions”, jossa kohdassa ”concat(...)” on suorituksenaikainen lauseke, jolla kootaan käyttäjän User Principal Name (UPN) keräämällä tieto playbookin laukaisimeen tulevasta datasta (kuva 3).



Kuva 3. Istunnot katkaisevan API-kutsun URI sekä UPN:n kokoava lauseke

Käytetty formaatti tässä lausekkeessa on Workflow Definition Language (WDL), jota voidaan Logic Appeilla käyttää suorituksenaikaisten arvojen hakemisessa ja käsittelyssä. Lausekkeen alussa määritetään käytetty funktio, joka on tässä ”concat”, ja sen jälkeisten sulkujen sisään määritetään funktion käyttämät parametrit ja operaattorit. (Azure Logic Apps 2024c.)

WDL:ssä ”?”-operaattori arvon jälkeen tarkoittaa, että kyseinen arvo on valinnainen. Suoritus siis ei epäonnistu, jos kyseistä arvoa ei täsmälleen sellaisenaan löydy, vaan se palauttaa silloin vastaukseksi ”null”, ja suoritus jatkuu. Hakasulkeiden sisään määritellään JSON-rakenteen sisäkkäiset arvot. (Azure Logic Apps 2024c.)

Concat-funktiolla voidaan yhdistää merkkijonoja yhdeksi merkkijonoksi. Suomalaisittain tätä voidaan kutsua konkatenoinniksi. Tässä kyseisessä concat-funktiossa näkyi, että siinä konkatenoidaan kolme merkkijonoa:

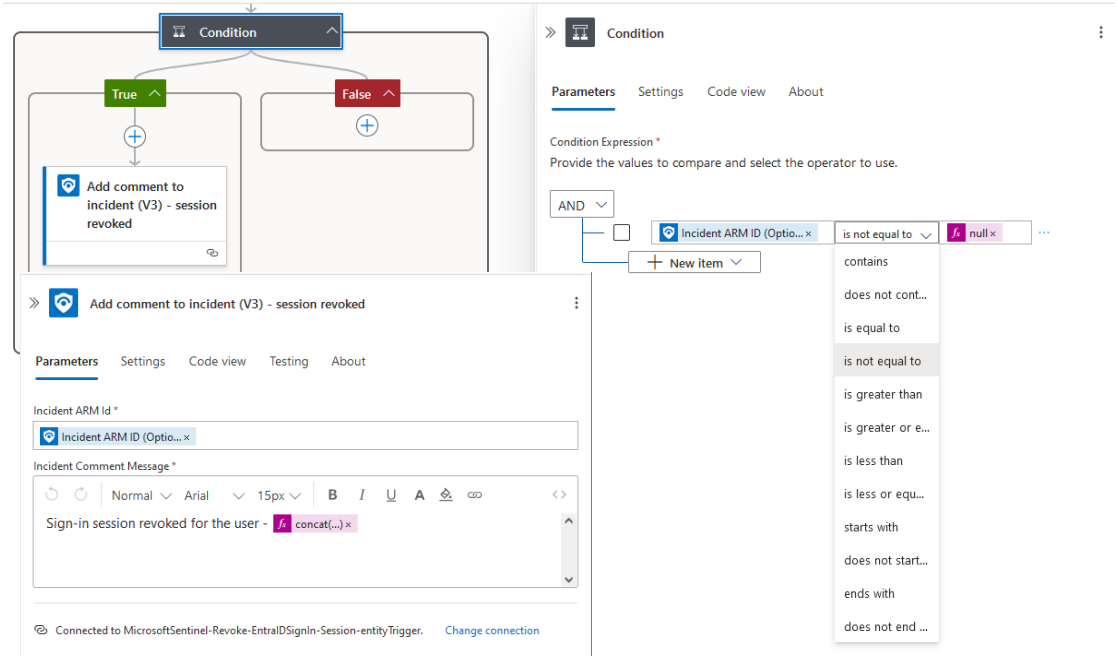
1) käyttäjän nimi (muodossa ”etunimi.sukunimi”)
2) merkki ”@”
3) UPN-pääte eli User Principal Namen loppuosa (muodossa ”domain.com”)

Nämä kolme merkkijonoa yhdistämällä saadaan muodostettua käyttäjän User Principal Name, eli ”etunimi.sukunimi@domain.com”.

API-päätepiste ”/revokeSignInSessions” resetoi käyttäjän tiedoissa olevan arvon ”signInSessionsValidFromDateTime” nykyhetkeen, joka käytännössä kirjaa käyttäjän ulos kaikista Azuren pilvisovellusten sekä selainten istunnoista (Microsoft Graph 2024c).

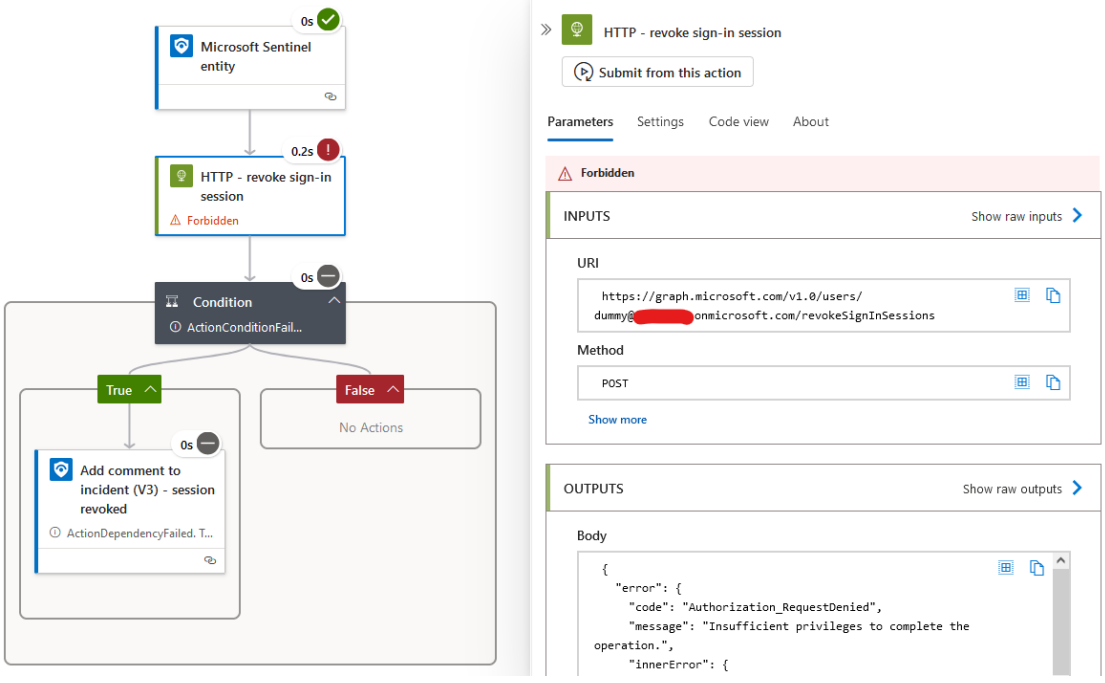
Autentikointityyppinä tässä API-yhteydessä käytetään Microsoftin tarjoamaa palvelua nimeltä ”system-assigned managed identity” eli suomalaisittain järjestelmän luoma hallinnoitu identiteetti. Sen myötä playbookille luodaan automaattisesti identiteetti, jonka kredentiaaleja ei käyttäjän tarvitse lainkaan säilyttää tai käsitellä, vaan tunnuksen hallinnointi tapahtuu Azuren toimesta automaattisesti taustalla. (Microsoft Entra 2024b.)

Lopuksi playbookissa oli Condition-toiminto, johon oli käytännössä määritelty, että jos playbookin laukaiseva käyttäjäentiteetti on valittu jonkin Sentinelin ”incidentin” eli tapahtuman alta, niin kyseisen tapahtuman tietoihin lisätään kommentti, joka kertoo, että käyttäjän istunnot ovat katkaistu (kuva 4).



Kuva 4. Condition-toiminto, sen ehtolauseke sekä True-polulle määritelty kommentin lisäys

Kun playbookin jokainen vaihe oli käyty läpi ja opeteltu ymmärtämään, mitä kukin toiminto playbookissa tekee, niin siirryttiin testaamaan kyseistä playbookia testikäyttäjällä. Ensimmäinen testaus johti playbookin epäonnistuneeseen suoritukseen. Syyksi selvisi playbookin riittämättömät oikeudet (kuva 5). Täytyi selvittää, mitä oikeuksia tarvitaan ja miten tarvittavat oikeudet lisätään. Tämän selvittely oli odottamattoman hankalaa, ja se vaati sekä Microsoftin dokumentaation että muiden verkkolähteiden tutkimista.



Kuva 5. Playbookin epäonnistuneen suorituksen näkymä virheviesteineen

Microsoftin dokumentaatio osoitti (Microsoft Graph 2024c), että playbookin hallinnoidulle identiteetille tarvittiin vähintään ”User.RevokeSessions.All”-oikeus kyseenomaiseen toimintoon, eli ”/revokeSignInSessions”-päätepisteen kutsumiseen. Kyseessä on applikaatio-oikeus, jonka lisääminen playbookin hallinnoidulle identiteetille ei onnistu Azuren portaalista, eikä myöskään Microsoftin dokumentaatiosta löytynyt yksiselitteistä tapaa tämän tekemiseen. Muita verkkolähteitä tutkimalla selvisi, että applikaatio-oikeuden voi lisätä PowerShell-skriptillä, mutta senkään osalta ei löytynyt täysin ajankohtaista tietoa, sillä useimmista lähteistä löytyneet skriptit käyttivät deprekoitunutta Azure AD PowerShell -moduulia. Päätettiin käyttää Päivisen blogissa (2024) käytettyä skriptiä mallina ja muuttaa se Microsoftin dokumentaatiota (Microsoft Graph 2024d) apuna käyttäen muotoon, joka käyttää Azure AD PowerShellin sijaan Microsoft Graph PowerShell -moduulia. Saatiin koottua skripti, joka onnistui lisäämään oikeuden playbookin hallinnoidulle identiteetille (kuva 6). Myös oikeuksien tarkistamiselle ja poistamiselle tehtiin vastaavanlaiset skriptit.

```
PS H:\> # Enter the name of the Logic App
$LogicAppName = "Revoke-EntraIDSignIn-Session-entityTrigger"

# Enter the name of the permission
$PermissionName = "User.RevokeSessions.All"

# Connect to Microsoft Graph
Connect-MgGraph -Scopes "AppRoleAssignment.ReadWrite.All", "Application.Read.All"

# Get Managed Identity Service Principal
$MI = Get-MgServicePrincipal -Filter "displayName eq '$LogicAppName'"

# Get Microsoft Graph Service Principal
$GraphServicePrincipal = Get-MgServicePrincipal -Filter "appId eq '00000003-0000-0000-c000-000000000000'"

# Get the app role
$AppRole = $GraphServicePrincipal.AppRoles |
    Where-Object {$_.Value -eq $PermissionName -and $_.AllowedMemberTypes -contains "Application"}

# Create new app role assignment
$params = @{
    PrincipalId = $MI.Id
    ResourceId = $GraphServicePrincipal.Id
    AppRoleId = $AppRole.Id
}

New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $MI.Id -BodyParameter $params
Welcome to Microsoft Graph!

Connected via delegated access using 14d82eec-204b-4c2f-b7e8-296a70dab67e
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs

NOTE: You can use the -NoWelcome parameter to suppress this message.


DeletedDateTime Id                                             AppRoleId
--------------- --                                             ---------
                fg3NY61SlOumYR7GGY-PsY6j__C23ppMuFdBpsWce5k 77f3a031-c388-4f99-b373-dc68676a...
```

Kuva 6. PowerShell-skripti, joka lisäsi oikeuden playbookin hallinnoidulle identiteetille

Skriptin suorittaminen vaati Azure CLI:n sekä Microsoft Graph PowerShell -moduulin asentamista. Asentamisten jälkeen tuli kirjautua PowerShellillä Azuren tenantiin, jossa playbook ja sen hallinnoitu identiteetti sijaitsivat. Tämä

tapahtuu komennolla ”az login”. Tarvittavien oikeuksien myöntäminen vaatii kirjautuvalta tunnukselta Privileged Role Administrator tai Global Administrator -oikeuksia (Microsoft Graph 2024b). Kirjautumisen jälkeen suoritettiin skripti. Oikeuden lisäämisen jälkeen playbook suoriutui onnistuneesti.

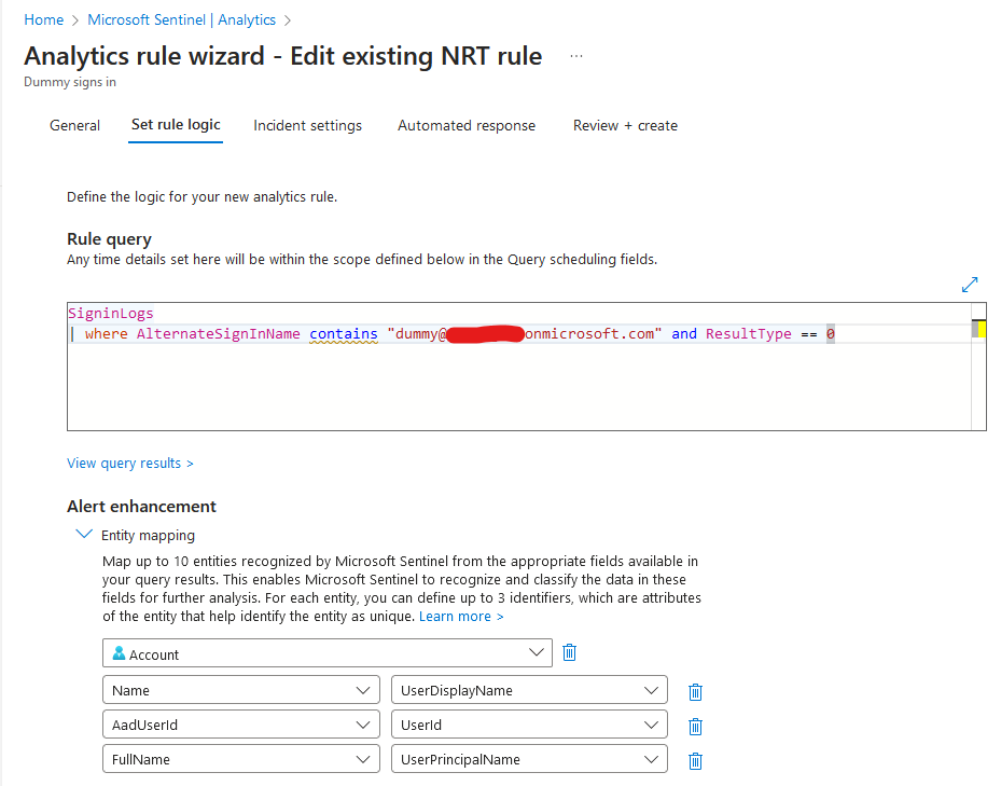
4.3 Kommentin lisääminen ja testitapahtuman luonti

Playbookissa ”Revoke Entra ID Sign-in session using entity trigger” oli toimintona myöskin kommentin lisääminen Sentinel-tapahtumaan. Tämä vaatii, että käyttäjä on jossakin Sentinel-tapahtumassa entiteettinä, jonka alta playbook voitaisi ajaa ja siten playbookin tulisi lisätä kommentti kyseiseen tapahtumaan. Täytyi siis luoda tässä kohtaa testitapahtuma käyttäjälle. Kuten alustavasti oli suunniteltu, tapahtuma voisi syntyä testikäyttäjän kirjautumisesta. Tämä olisi riittävän helppo ja hallittavissa oleva tapahtuma testailua varten.

Sentinelissä on mahdollista luoda tapahtuma analytiikkasäännön avulla. Vaihtoehtoja säännönluomisessa on käytännössä kolme: ”Scheduled query rule”, ”NRT (near-real-time) query rule” sekä ”Microsoft Incident creation rule”. Näistä jälkimmäisin ei sovellu tähän tarkoitukseen, sillä se luo Sentinel-tapahtumia vain automaattisesti Microsoftin eri tietoturvatuotteiden tuottamista hälytyksistä, eikä siihen ole konfiguroitavissa spesifejä tapauksia.

Asiaan perehtymisen jälkeen päädyttiin NRT-tyypin sääntöön. Ratkaisevana tekijänä oli matalampi viive; Scheduled eli ajastetuissa säännöissä on rajoitteena, että niissä on sisäänrakennettuna viiden minuutin viive, kun taas NRT-säännöissä viive on vain kaksi minuuttia (Microsoft Sentinel 2024c).

Analytiikkasäännön logiikkaan määriteltiin yksinkertaisesti vain testikäyttäjän onnistunut kirjautuminen, sekä Sentinel-tapahtuman tietoja rikastettiin käyttäjän näyttönimellä, ID:llä ja UPN:llä (kuva 7).



Kuva 7. Analytiikkasääntöön määritetty logiikka sekä tietojen rikastus

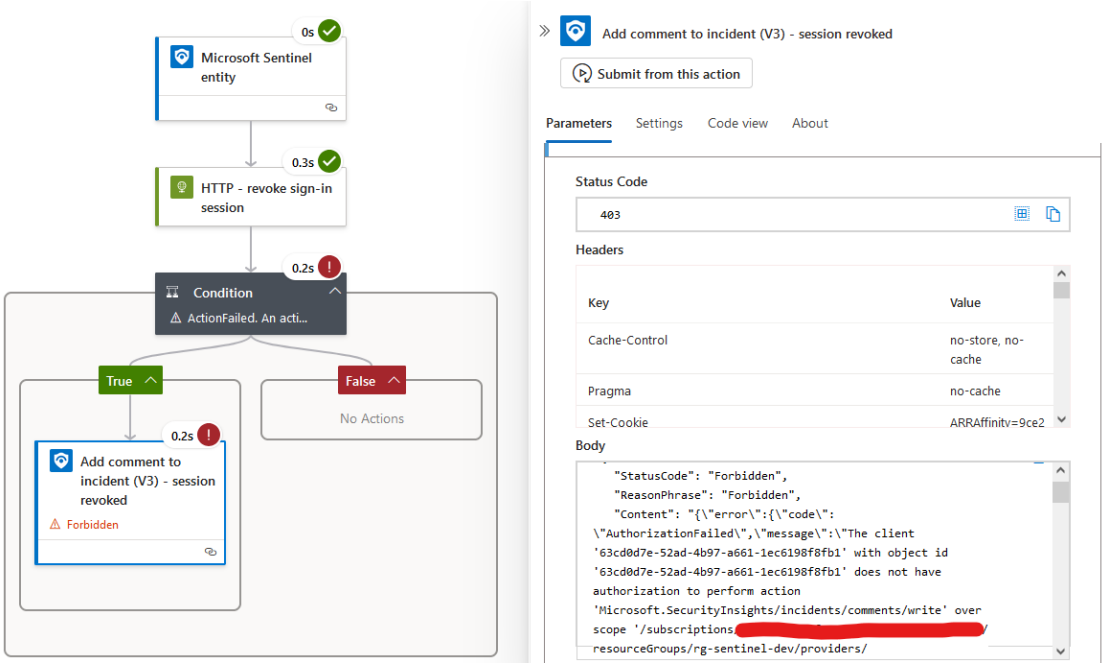
Säännön luomisen jälkeen kirjauduttiin sisään testikäyttäjällä, jonka jälkeen muutaman minuutin kuluttua syntyi Sentinel-tapahtuma, jonka entiteeteissä näkyi testikäyttäjä. Playbook ajettiin testikäyttäjälle, mutta tällä kertaa suoritus epäonnistui. Suoritushistorian tarkastelussa näkyi, että suoritus kaatui riittämättömiin oikeuksiin ”/revokeSignInSessions”-API-kutsun kohdalla. Samoilla oikeuksilla oli hetki sitten sama toiminto onnistunut, joten tämä herätti ihmetystä. PowerShellillä myös tarkastettiin, että sama ”User.RevokeSessions.All”-oikeus oli edelleen voimassa playbookin hallinnoidulla identiteetillä, kuten näkyy kuvassa 8.

```
API          : Microsoft Graph
Logic App    : Revoke-EntraIDSignIn-Session-entityTrigger
Permission   : User.RevokeSessions.All
Description  : Allow the app to revoke all sign in sessions for a user, without a signed-in user.
Resource ID  : 6f7e8b15-8cf8-4476-a013-b51e423092bf
App Role ID  : 77f3a031-c388-4f99-b373-dc68676a979e
```

Kuva 8. Ajetun PowerShell-skriptin tulos, joka tarkistaa playbookin hallinnoidulle identiteetille myönnetyt oikeudet

Epäilys oli, että ehkä kyseessä oli vain jokin hetkittäinen virhe, joten playbook ajettiin uudestaan. Uusi suoritus epäonnistui myös, mutta suoritushistorian mukaan nyt itse asiassa suoritus katkesi myöhemmässä vaiheessa (kuva 9). API-kutsu ”/revokeSignInSessions” meni läpi onnistuneesti, joten epäilys näytti

osuneen oikeaan. Tällä kerralla näytti puuttuvan oikeudet lisätä Sentinel-tapahtumalle kommentti.



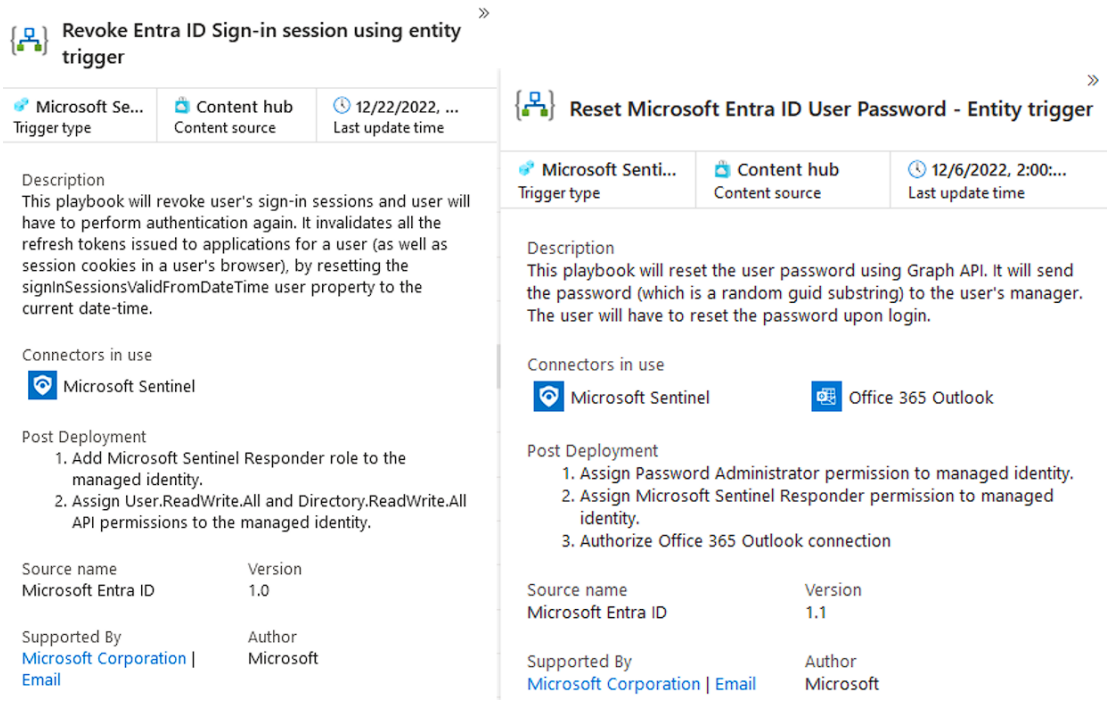
Kuva 9. Suoritushistoriassa näkyy, että puuttui oikeus lisätä kommentti Sentinel-tapahtumaan

Microsoftin dokumentaatiosta löytyi (Microsoft Sentinel 2024a), että Sentinel-tapahtuman päivittämiseen sekä kommenttien lisäämiseen tarvitaan Microsoft Sentinel Responder -rooli. Roolijako lisättiin Azuren IAM-portaalista playbookin hallinnoidulle identiteetille. Tämän jälkeen playbook ajettiin uudestaan, jolloin se suoriutui kokonaisuudessaan onnistuneesti katkaisten testikäyttäjän istunnot sekä lisäten kommentin Sentinel-tapahtumalle.

4.4 Salasanan resetointi

Seuraavaksi oli vuorossa perehtyä salasanan resetointiin playbookilla. Valmiista Sentinelin malleista löytyi playbook nimeltä ”Reset Microsoft Entra ID User Password - Entity trigger”. Tässä kohtaa huomattiin, että nämä valmiit mallit sisältävät myös neuvoja tarvittavista oikeuksista. Tämä oli jäänyt huomaamatta aiempaa playbookia luodessa. Mutta samalla huomattiin myös, että kyseisen playbookin ohjeessa neuvottiin antamaan korkeamman tason oikeudet, kuin mikä testatessa oli toiminut; ohjeen mukaan tarvittiin ”User.ReadWrite.All”- ja ”Directory.ReadWrite.All” -oikeudet, vaikka sekä Microsoftin dokumentaatiosta (Microsoft Graph 2024c) että käytännön testaamisen kautta selvisi, että riittävä oikeus todellisuudessa on

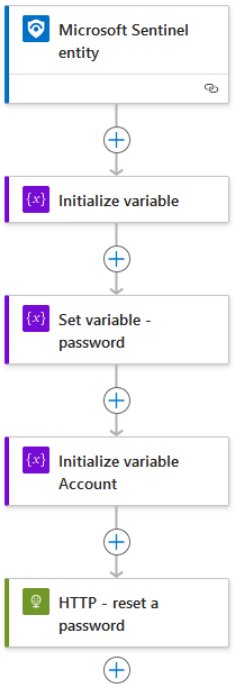
”User.RevokeSessions.All”. Kuvassa 10 näkyy molempien mainittujen playbookien kuvaukset ja ohjeet tarvittavista oikeuksista.



Kuva 10. Sentinelin valmiiden playbook-mallien kuvaukset ohjeineen

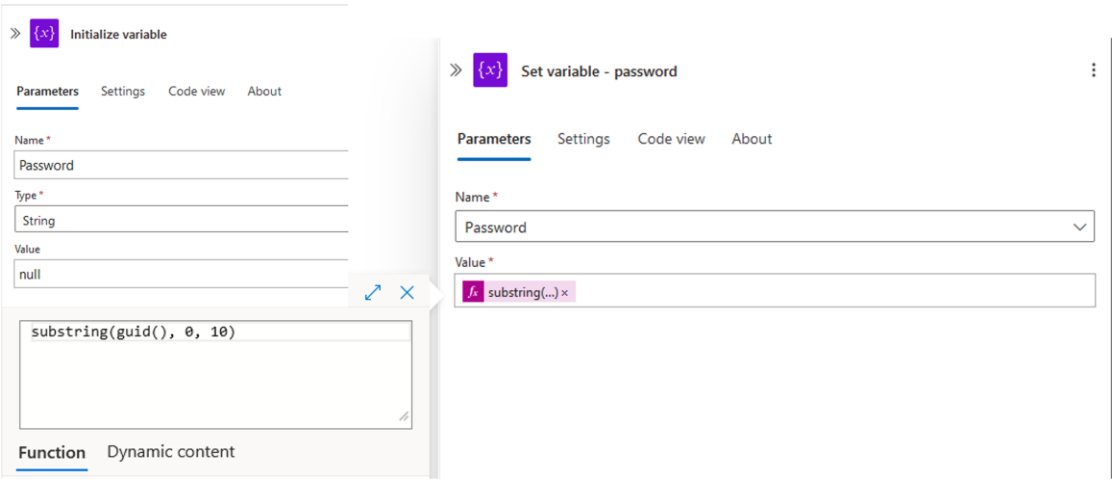
Todettiin, että todennäköisesti kyseisen playbook-mallin ohje on vanhentunut, ja ”User.RevokeSessions.All”-oikeutta ei mahdollisesti edes ollut vielä olemassa tuolloin, kun kyseinen ohje kirjoitettiin, sillä viimeisimpänä mallin päivitysajankohtana näkyi 22/12/2022. Tämä mielessä pitäen täytyi varmistaa seuraavankin mallin ohjeen paikkansapitävyys vielä erikseen sekä Microsoftin dokumentaatiosta, että testaamalla käytännössä.

Salasanaresetoinnin playbook luotiin ja sisältöön lähdettiin perehtymään vaihe vaiheelta, kuten aiemmankin playbookin kohdalla. Tähän playbookiin sisältyi myös toimintoja, jotka liittyivät sähköpostin lähettämiseen kohdekäyttäjän esihenkilölle. Nämä toiminnot päätettiin ottaa pois playbookista, jotta voitiin keskittyä yksinomaan salasanan resetoinnin testaamiseen. Kuvassa 11 on playbookin sisältö ylimääräisten toimintojen poistamisen jälkeen.



Kuva 11. Entiteetistä laukaistava salasanaresetoinnin playbook ylimääräisten toimintojen poistamisen jälkeen

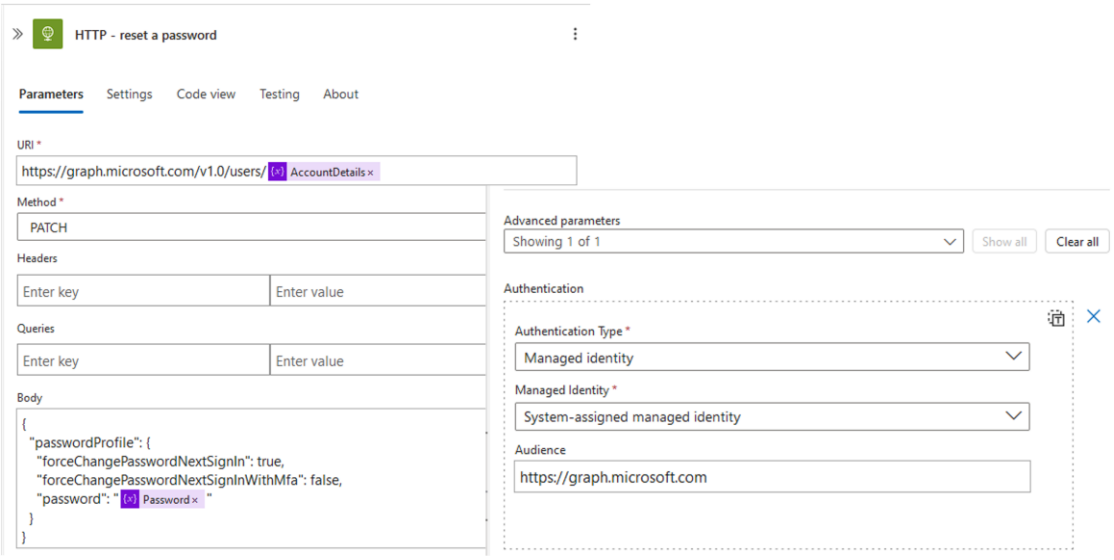
Playbookissa luodaan ensin tyhjä muuttuja salasanalle kohdassa ”Initialize variable”, ja sitten kohdassa ”Set variable - password” generoidaan guid- ja substring -funktioita käyttäen 10:n satunnaisen merkin salasana (kuva 12). Tämä tapahtuu siten, että guid-funktio luo uniikin tunnisteen, ja substring-funktiolla luodusta tunnisteesta otetaan vain 10 ensimmäistä merkkiä.



Kuva 12. Playbookin "Initialize variable" ja "Set variable - password" -vaiheet

Näiden jälkeen kohdassa ”Initialize variable Account” konkatenoidaan käyttäjän UPN ja asetetaan se muuttujan ”AccountDetails” arvoksi. Tätä muuttujaa käytetään playbookin viimeisessä vaiheessa ”HTTP – reset a

password”, jossa tehdään API-kutsu käyttäjän profiiliin Graphin kautta. Kutsun metodi on PATCH, ja sillä muokataan käyttäjän ”passwordProfile”-osion tietoja. Käyttäjälle asetetaan uusi väliaikainen salasana, joka playbookin aiemmassa vaiheessa luotiin, sekä pakotetaan käyttäjä vaihtamaan salasana seuraavan kirjautumisen yhteydessä. Myös tässä käytetään autentikointityyppinä järjestelmän luomaa hallinnoitua identiteettiä. Nämä asetukset on kuvattuna kuvassa 13.



Kuva 13. Playbookin "HTTP - reset a password" -vaihe asetuksineen

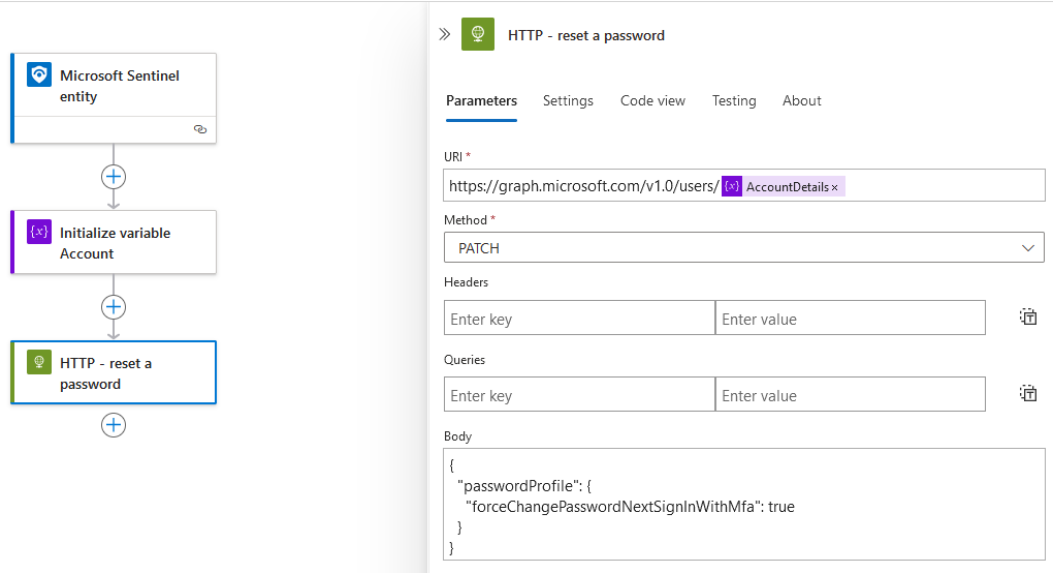
Playbookin sisällön tarkastelun jälkeen oli aika lisätä tarvittavat oikeudet. Playbook-mallin ohjeen mukaan tarvittiin Password Administrator -roolin jako Entrassa playbookin hallinnoidulle identiteetille. Mutta Microsoftin dokumentaation mukaan (Microsoft Graph 2024e) ”passwordProfile”-kohdan muokkaaminen vaatisi applikaation tapauksessa ”User.ReadWrite.All”- tai ”Directory.ReadWrite.All”-oikeuden, sekä User Administrator -roolijaon Entrassa. Koska User Administratorilla on selkeästi korkeammat oikeudet, kuin Password Administratorilla, niin päätettiin kokeilla ensin pelkkää Password Administratoria. Rooli asetettiin Azuren portaalista Entra ID:n sivulta ”Roles and assignments”.

Roolin asettamisen jälkeen testattiin playbookin suoritusta. Suoritus oli onnistunut, eikä testikäyttäjällä päässyt enää kirjautumaan sisään vanhalla salasanalla. Playbookilla asetettua salasanaa ei lähetetty mihinkään, mutta sen sai kaivettua playbookin suoritushistoriasta. Salasanan syöttämisen

jälkeen pyydettiin luomaan uusi salasana. Password Administrator oli siis oikeuksien puolesta riittävä rooli näihin toimenpiteisiin.

4.5 Salasanan vaihdon pakotus MFA:n kanssa

Pohdinnan aiheeksi tuli, että millä keinolla voitaisiin pakottaa käyttäjä vaihtamaan salasana ilman, että tarvitsee lähettää käyttäjälle väliaikaista salasanaa. Jos organisaatiolla on käytössä MFA:n vaativa linjaus, niin pelkkä salasanan vaihdon pakotus seuraavan kirjautumisen yhteydessä MFA:n kanssa voisi olla toimiva toimenpide. Päätettiin testata "forceChangePasswordNextSignInWithMfa"-arvon asettamista arvoon ”true”, ja ottaa väliaikaisen salasanan asetus pois käytöstä. Kuvassa 14 näkyy vasemmalla playbookin vaiheet ja oikealla playbookin viimeisen vaiheen asetukset tehtyjen muutosten jälkeen.



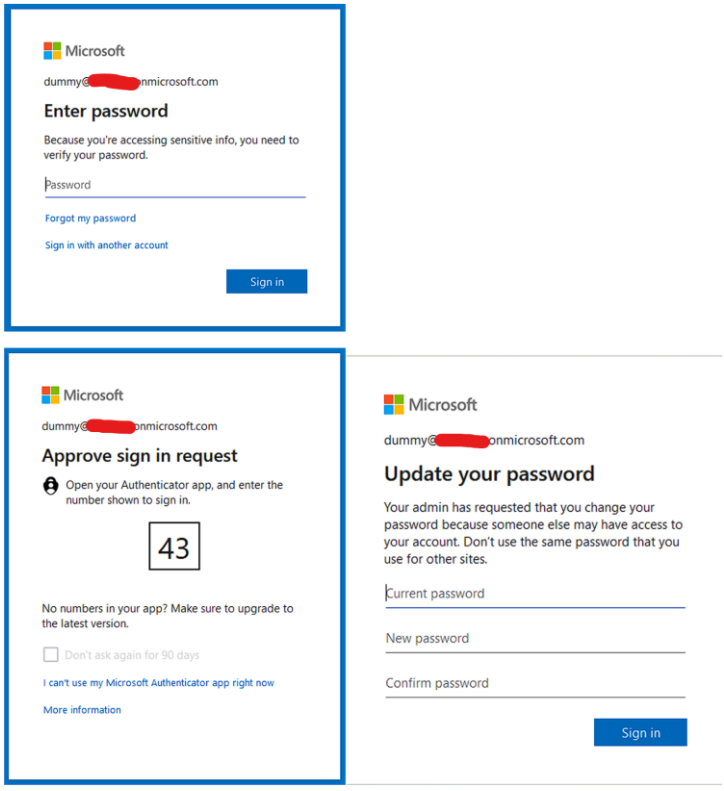
Kuva 14. Playbookin sisältö pelkällä salasanan vaihdon pakotuksella

Playbookia testattiin, ja sen kaikki vaiheet suoriutuivat onnistuneesti. Testikäyttäjää pyydettiin Azuren portaalissa kirjautumaan uudelleen, vaikkei istuntojen katkaisua erikseen tehtykään playbookissa API-kutsulla. Mutta kun käyttäjällä kirjautui uudestaan sisään ja pyydettiin luomaan uusi salasana, niin MFA:ta ei kysytty. Tämän epäiltiin johtuneen siitä, että aiemmin testikäyttäjällä kirjautuessa oli valittu optio, ettei MFA:ta tarvitse kysyä seuraavaan 90 päivään. On siis tärkeää huomioida, että potentiaalisesti hyökkääjä voisi päästä itse vaihtamaan salasanan ja kirjautumaan ilman MFA:ta, jos hän olisi

murtautuessaan päässyt valitsemaan samaisen ”Don’t ask again for x days” -option, ja saavuttanut siten pysyvän selainistunnon.

Testikäyttäjällä kirjauduttiin ulos ja kokeiltiin toisella selaimella kirjautumista. MFA:ta kysyttiin. Tällä kertaa jätettiin ”Don’t ask again for 90 days” -optio valitsematta, ja kokeiltiin playbookin ajoa uudestaan. Testikäyttäjältä pyydettiin jälleen uudelleenkirjautumista, mutta taaskaan ei MFA-kyselyä tullut. Vaikutti siis siltä, että aiempi hypoteesi ei pitänyt paikkansa siitä, että MFA-kyselyn puute olisi johtunut ”Don’t ask again for x days” -optiosta.

Sisäänkirjautumisen jälkeen playbookia yritettiin testata uudelleen, mutta tällä kertaa mitään ei näyttänyt tapahtuvan; testikäyttäjällä sai liikkua vapaasti portaalista ja päivitellä sivua, eikä uudelleenkirjautumisen pyyntöä tullut. Päätettiin ajaa istuntojen katkaisu testikäyttäjälle aiemmin testatulla playbookilla, jonka myötä testikäyttäjä kirjautui ulos. Kun testikäyttäjällä yritti kirjautua uudestaan sisään, pyydettin ensin salasana, jonka jälkeen MFA, ja lopuksi salasanan vaihto (kuva 15).



Kuva 15. Kirjautumisen yhteydessä pyydetyt asiat järjestyksessä

Huomioitavaa tässä oli, että nyt ”Don’t ask again for 90 days” -optio ei ollut saatavilla. Epäiltiin, että tämä on luultavasti ominaisuus, jonka tarkoitus on ehkäistä juuri aiemmin pohdittu skenaario, jossa hyökkääjä voisi murtautumisen jälkeen saavuttaa pysyvän selainistunnon. Tähän viittaisi myös salasanan päivityksen ikkunassa oleva teksti, joka viestii käyttäjälle mahdollisesta murtautumisesta.

Playbookia testattiin vielä uudestaan, eikä taaskaan käyttäjää uloskirjattu muuta kuin vasta istuntojen katkaisevan playbookin ajamisen jälkeen. Sen myötä testikäyttäjälle tapahtui jälleen samat asiat, kuin edellisellä kierroksella: kirjautuminen vanhalla salasanalla, MFA-kysely, ja salasanan vaihto.

Päätettiin vielä testata mitä käy, jos playbookissa asetetaan "forceChangePasswordNextSignIn" arvoon ”true”, ja "forceChangePasswordNextSignInWithMfa" arvoon ”false”. Samat asiat tapahtui kuin edellisellä kierroksella, eli vasta istuntojen katkaisun playbookin ajamisen jälkeen testikäyttäjä kirjautui ulos, ja sisäänkirjautuessa pyydettiin vanha salasana, MFA sekä salasanan vaihto. Eli myös tällä kertaa MFA:ta kysyttiin siitä huolimatta, että "forceChangePasswordNextSignInWithMfa" asetettiin arvoon ”false”.

Vaikuttaisi siis siltä, että MFA:n kysely tulee joka tapauksessa, mutta tämä luultavasti johtuukin siitä, että Microsoft vaatii nykyään Azuren portaaliin kirjautumisilta aina MFA:n, riippumatta Conditional Access -politiikoista tai muista asetuksista (Microsoft Entra 2024a).

Testailun perusteella vaikutti siltä, että vaikka toisinaan pelkkä salasanan vaihdon pakotuksen asettaminen pakottaa käyttäjän myös kirjautumaan uudelleen, niin se ei tapahdu joka kerta. Istuntojen katkaisu API-kutsulla puolestaan näytti toimivan luotettavasti, ja se yhdistettynä salasanan vaihdon pakottavaan API-kutsuun näytti toteuttavan käyttötapauksen, jossa käyttäjälle pakotetaan uloskirjautuminen, sekä salasanan vaihtaminen MFA-suorituksen kanssa. Kaksi edellämainittua API-kutsua voitaisiin yhdistää saman playbookin alle.

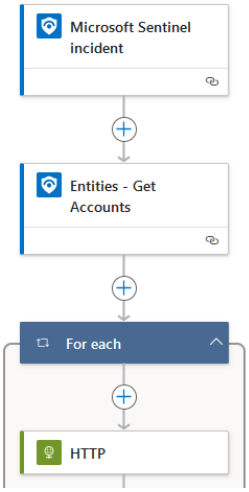
Salasanan resetointiin voi olla muitakin vaihtoehtoja, kuin mitä tähän mennessä on työssä tutkittu. Esimerkiksi jos käyttäjätunnus on Entra ID:n lisäksi myös on-prem AD:ssa, jolloin kyseessä olisi niin sanottu hybridi AD, niin voitaisi mahdollisesti vaatia toisenlaista lähestymistä. Salasanan resetoinnin mahdollisuuksia playbookilla oli kuitenkin demonstroitu tässä riittävästi tämän työn puitteissa, ja päätettiin siirtyä seuraaviin toimintoihin testailussa.

4.6 Tapahtumalaukaisimen playbook

Tähän mennessä oltiin testattu vain manuaalisesti ajettavia, eli Sentinelin entiteettiä laukaisimena käyttäviä playbookeja. Seuraavaksi siirryttiin suunnittelemaan playbook, joka voisi suoriutua automaattisesti Sentinel-tapahtuman syntyessä. Kyseiseen playbookiin voitaisiin lisätä sekä istuntojen katkaisu, että salasanan vaihdon pakotus aiempien testausten perusteella. Lisäksi Sentinel-tapahtumaan voitaisiin esimerkiksi lisätä kommentti, ja mahdollisesti muita tietoja.

Suunnittelu aloitettiin jälleen katsomalla, mitä Sentinelin valmiit mallit tarjoaa. Sekä istuntojen katkaisulle, että salasanan resetoinnille löytyi myös Sentinel-tapahtumasta laukaistavat mallit. Istunnot katkaisevan playbook-mallin (”Revoke Entra ID SignIn Sessions – incident trigger”) ohjeessa neuvottiin tällä kertaa, että tarvittaisiin ”User.ReadWrite.All”-oikeus playbookin hallinnoidulle identiteetille. Tämä jälleen poikkesi sekä Microsoftin dokumentaatiosta (Microsoft Graph 2024c), että käytännön testaamisen tuloksesta, kuin myös vastaavan entiteetistä laukeavan playbook-mallin ohjeesta. Päätettiin jatkaa toimivaksi todetun ”User.RevokeSessions.All”-oikeuden käyttämistä.

Playbook luotiin ja sen sisältöä alettiin tarkastelemaan. Entiteetistä laukaistavaan playbookin verrattuna olennaiseksi eroavaisuudeksi havaittiin, että playbookin laukaisun jälkeen on toiminto ”Entities – Get Accounts”, jossa kerätään kaikki kyseisen Sentinel-tapahtuman entiteetit, sekä toiminto ”For each”, joka selaa entiteetit läpi ja tekee API-kutsun vain entiteeteille, joiden tyyppi on ”Account” eli käyttäjätunnus. Kuvassa 16 näkyy mainitut playbookin alkuvaiheet.



Kuva 16. ”Revoke Entra ID SignIn Sessions – incident trigger” -playbookin alkuvaiheet

Tällä kertaa päätettiin olla testaamatta valmiista mallista tehtyä playbookia, sillä seuraavaksi tarkoituksena oli luoda uusi tapahtumalaukaisimen playbook tyhjästä, jonka kokoamisessa käytettäisiin vain referenssinä valmiin mallin playbookia. Uusi playbook luotiin Sentinelin Automation-osion yläriviltä kohdasta ”Create”, josta valittiin ”Playbook with incident trigger”. Playbookille annettiin nimi, ja sen yhteydessä keksittiin myös nimeämiskäytäntö, jota tulevatkin itseluodut playbookit voisivat käyttää (suluissa olevat valinnaisia):

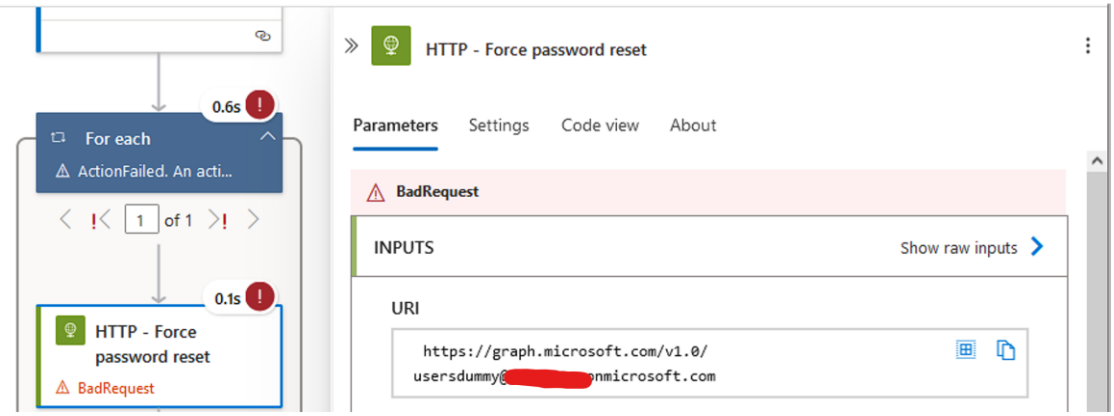
- (Asiakastunniste)-Laukaisintyyppi-Päätoiminto-(Kohdealusta/laajuus)
- Esimerkki: ASI-Incident-RevokeSessions_ResetPassword-EntraID

Perusteluna nimeämiskäytännölle on, että playbookien järjestely olisi selkeää, kun playbookit järjestyisivät listalla aakkosjärjestyksen mukaisesti ensin asiakaskohtaisesti, sitten laukaisintyypin mukaan, ja lopuksi playbookin käyttötarkoituksen ja -kohteen perusteella.

Tämän myötä uuden playbookin nimeksi muotoutui ”Incident-RevokeSessions_ResetPassword-EntraID”. Playbookille lisättiin mallin mukaisesti toiminnot entiteettien keräämiselle ja käyttäjätunnusten selaamiselle ”For each” -toiminnolla. Toiminnon alle lisättiin istuntojen katkaiseva API-kutsu, salasanan vaihdon pakottava API-kutsu, sekä kommentin lisäys Sentinel-tapahtumaan. Kun playbookin sisältö oli valmis, täytyi vielä lisätä tarvittavat oikeudet playbookin hallinnoidulle identiteetille, jotka olivat:

- User.RevokeSessions.All (application permission)
  - Tarvitaan istuntojen katkaisuun API-kutsulla
  - Lisätään PowerShell-skriptillä
- Password Administrator (Entra built-in role)
  - Tarvitaan salasanan vaihdon pakotukseen API-kutsulla
  - Lisätään Entra ID -portaalista
- Microsoft Sentinel Responder (Azure built-in role)
  - Tarvitaan kommentin lisäämiseen Sentinel-tapahtumaan
  - Lisätään Sentinelin resurssiryhmän IAM-portaalista

Kun oikeudet oli lisätty, niin testattiin playbookin ajoa. Sen pystyi ajamaan samalle testitapahtumalle, joka aiemmin luotiin. Ensimmäinen suoritus epäonnistui, mutta sen syyksi selvisi vain kirjoitusvirhe salasanan vaihdon pakottavassa API-kutsussa, josta oli jäänyt kauttaviiva puuttumaan ”users”-kansion jälkeen (kuva 17).



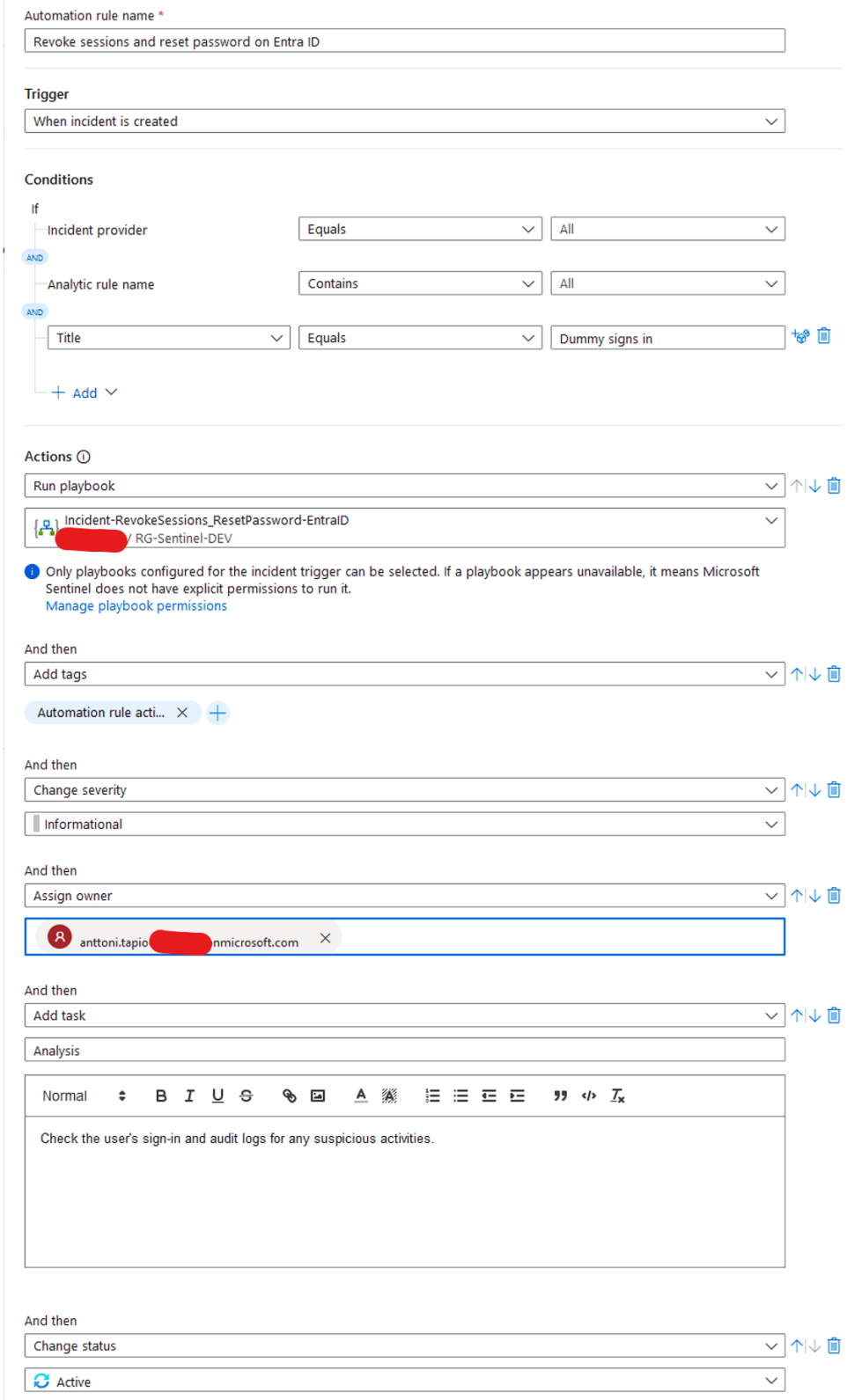
Kuva 17. Suoritushistoriassa näkyy, että URI:sta puuttuu kauttaviiva users-kansion jälkeen

Kirjoitusvirheen korjaamisen jälkeen playbookin suoritus onnistui. Testikäyttäjälle tapahtui odotetut asiat (uudelleenkirjautuminen, salasana, MFA, salasanan vaihto) ja Sentinel-tapahtumalle ilmestyi playbookin lisäämä kommentti.

4.7 Automaatiosäännön luonti

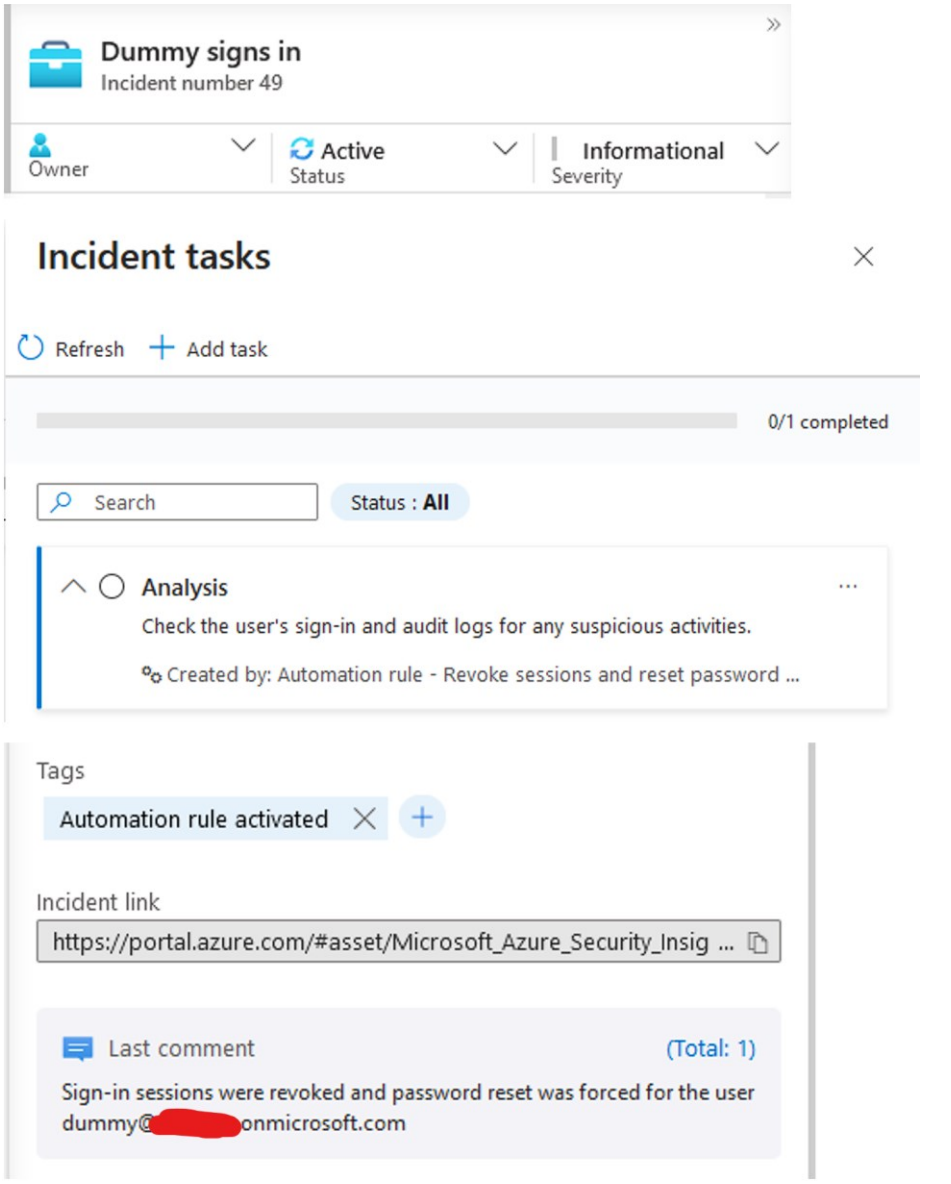
Seuraavana oli vuorossa automaatiosäännön luominen, joka kytkettäisiin laukaisemaan luotu playbook tiettyjen ehtojen täyttyessä. Uuden säännön pääsee luomaan samasta paikasta, kuin playbookit, eli Automation-osion yläriviltä kohdasta ”Create”, josta valitaan ”Automation rule”. Automaatiosäännön tekeminen Sentinelissä on hyvin suoraviivaista ja käyttäjäystävällistä, eikä paljoa konfiguroitavia osia ole. Tässä tapauksessa

käytännössä riitti, että annettiin kuvaava nimi, laukaisimeksi ”When incident is created”, ehdoksi ”If Title Equals ’Dummy signs in’” ja sen jälkeen Sentinel-tapahtumaan kohdistuvat toiminnot kohdassa ”Actions”. Demonstroinnin nimissä päätettiin lisätä yksi jokaisen tyyppistä toimintoa Actions-kohdassa, joita on yhteensä kuusi: playbookin käynnistys, tägin lisäys, severiteetin muutos, haltijan lisäys, tehtävän lisäys ja tilapäivitys. Kuvassa 18 näkyy automaatiosääntöön lisätyt asetukset kokonaisuudessaan.



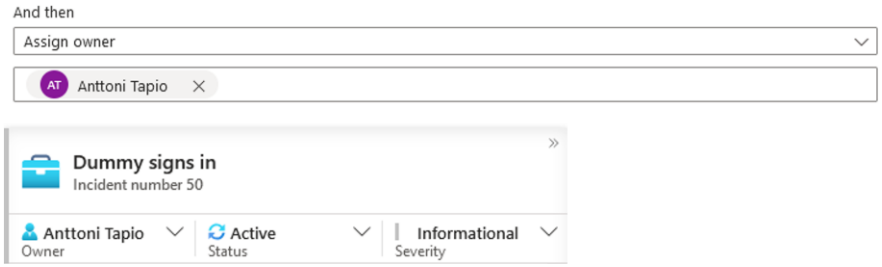
Kuva 18. Automaatiosäännön asetukset

Automaatiosääntö luotiin ja siirryttiin testaamaan sen toimintaa yhdessä playbookin kanssa. Testikäyttäjällä kirjauduttiin Azuren portaaliin. Muutaman minuutin kuluttua Sentineliin syntyi testitapahtuma. Kun sen tietoja mentiin tarkastelemaan, niin näkyi useita indikaattoreita automaatiosäännön toiminnasta (kuva 19).



Kuva 19. Automaatiosäännön tekemät toiminnot Sentinel-tapahtumaan

Sentinel-tapahtuman tila ja severiteetti muuttuivat, ja sen tietoihin lisättiin tehtävä, tägi sekä playbookin lisäämä kommentti. Kuudesta automaatiosääntöön määritetystä tehtävästä vain haltijan lisäys oli jäänyt puuttumaan. Syyksi sen epäonnistumiseen epäiltiin annetun tunnuksen väärää muotoa. Se korjattiin, jonka jälkeen automaatiosääntöä kokeiltiin uudestaan uudella testikäyttäjän kirjautumisella. Korjauksen myötä myös Sentinel-tapahtuman haltija päivittyi oikein (kuva 20). Myös testikäyttäjälle tuli pakotettu uloskirjautuminen ja salasanan vaihto pian Sentinel-tapahtuman syntymisen jälkeen.



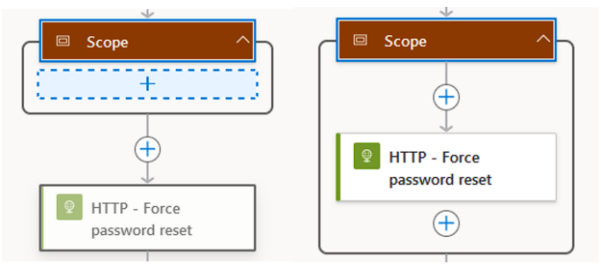
Kuva 20. Ylhäällä automaatiosääntöön tehty korjaus ja alhaalla sen tulos

4.8 Virheiden ja poikkeuksien käsittely

Toistaiseksi työssä luotuihin playbookeihin ei ollut sisällytetty minkäänlaista virheiden ja poikkeuksien käsittelyä. Jos jokin osio playbookista epäonnistui, niin se tavallisesti kaatoi koko playbookin suorituksen, eikä epäonnistuneesta suorituksesta tullut mitään ilmoitusta minnekään näkyvään paikkaan, muuta kuin playbookin suoritushistoriaan.

Tätä asiaa tutkimalla päädyttiin perehtymään toimintoon nimeltä ”Scope”. Scope-toiminnon sisälle voidaan laittaa yksi tai useampi toiminto, ja Scope tuottaa tuloksen sisältämiensä toimintojen suorituksista. Tuloksena voi olla ”Succeeded”, ”Failed”, ”Cancelled” tai ”Skipped”. Scopen jälkeen voi laittaa Condition-toiminnon, johon voidaan määrittää esimerkiksi, että jos edeltävän Scopen tulos on ”Succeeded”, niin tulos on ”True”, ja muutoin ”False”. Tätä metodia voidaan hyödyntää muun muassa virheiden ja poikkeamien käsittelyssä. (Azure Logic Apps 2024b.)

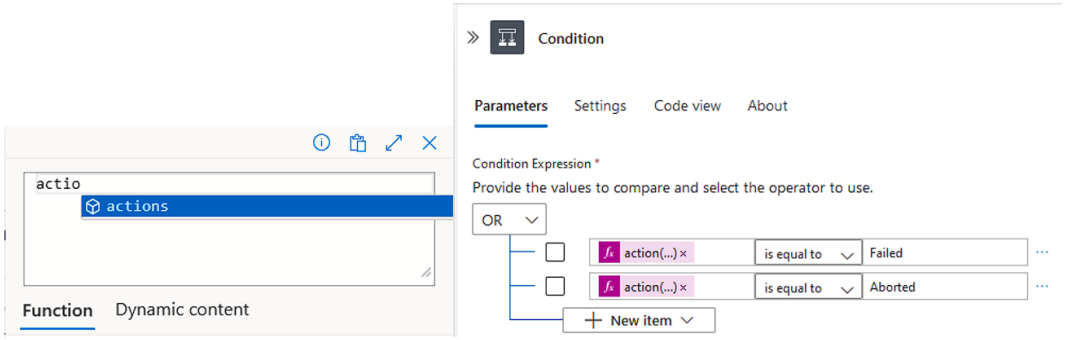
Aluksi Scope-toiminnon toteutusta lähdettiin kokeilemaan Microsoftin dokumentaatiosta löytyneen tutoriaalin kautta (Azure Logic Apps 2024b). Scopen ensimmäiseksi kohteeksi playbookissa valittiin salasanan vaihdon pakotuksen API-kutsu. Luotiin uusi Scope-toiminto API-kutsun yläpuolelle, ja raahattiin API-kutsu sen sisälle, joka on havainnollistettu kuvassa 21.



Kuva 21. Havainnollistus "HTTP - Force password reset"-toiminnon siirtämisestä "Scope"-toiminnon sisälle

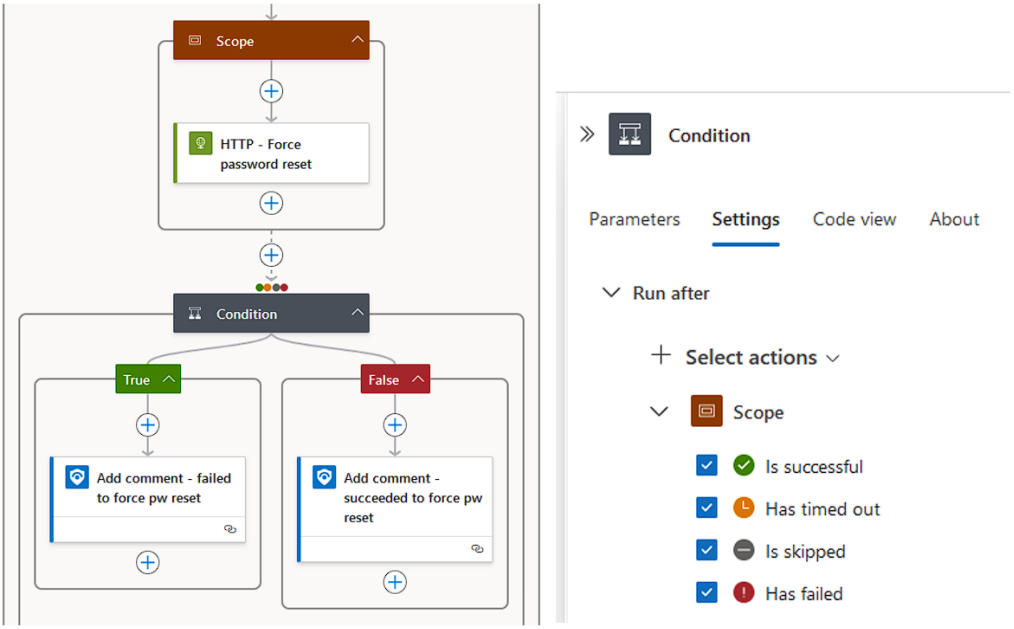
Tämän jälkeen Scope-toiminnon jälkeiseksi toiminnoksi lisättiin ”Condition”. Siihen tuli tutoriaalin mukaan laittaa ehtolausekkeeksi:
”action(’Scope’)” is equal to ”Failed” or ”Aborted”

Tätä syöttäessä huomattiin, ettei ”action”-nimistä funktiota tunnisteta olemassa olevaksi funktioksi, vaan ainoastaan ”actions” tunnistetaan, kuten näkyy kuvassa 22. Siitä huolimatta päätettiin ensin testata täsmälleen tutoriaalin mukaisesti, ja pidettiin funktio muodossa ”action(’Scope’)”



Kuva 22. Funktiota syöttäessä järjestelmä tunnistaa vain funktion "actions", ei funktiota "action"

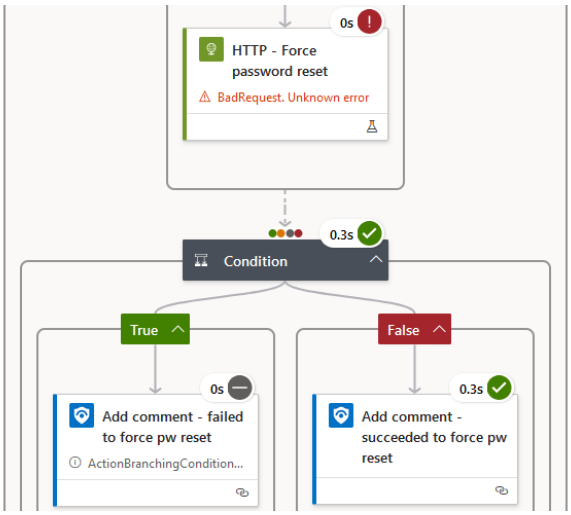
Tämän jälkeen tutoriaalissa neuvottiin, että täytyi laittaa Condition-toiminnon asetuksissa ”Run after” -asetukseen kaikki neljä valintaa päälle, joka tarkoittaa sitä, että playbookin suoritus jatkuu Scope-toiminnosta Condition-toimintoon ilman playbookin kaatamista, oli Scopen tulos mikä tahansa. Tämä tehtiin, ja lopuksi Condition-toiminnon True-polkuun lisättiin kommentin lisäys Sentinel-tapahtumaan, joka kertoo että salasanan vaihdon pakotus on epäonnistunut, sekä False-polulle vastaava kommentti onnistumisesta (kuva 23).



Kuva 23. Vasemmalla lisätyt toiminnot kommenttien lisäyksille, oikealla "Run after" -asetukset

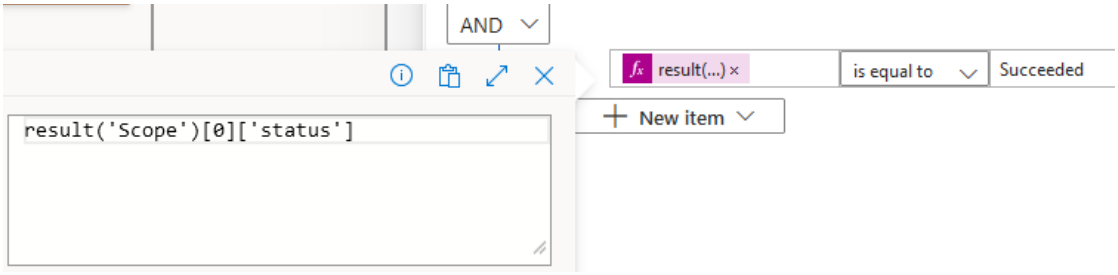
Playbookin ajoa testattiin, ja sen suoritus epäonnistui. Virheviestin perusteella action-funktiota ei tunnistettu. Päätettiin korjata funktio muotoon ”actions”, ja kokeilla playbookin ajoa uudestaan. Tällä kertaa playbookin suoritus onnistui ja Condition-toiminnossakin suoritus eteni oikealle eli False-polulle.

Tämän jälkeen päätettiin kokeilla, mitä käy jos salasanan vaihdon pakotuksen API-kutsu jostain syystä epäonnistuu. Tässä apuna käytettiin toiminnon asetuksissa olevaa Testing-osiota, jossa pystyy asettamaan staattisen tuloksen kyseiselle toiminnolle. Asetettiin tulokseksi ”Failed” virhekoodilla ”BadRequest”, ja kokeiltiin playbookia uudestaan. Suoritus eteni taas False-polulle, vaikka nyt sen olisi pitänyt mennä True-polulle (kuva 24).



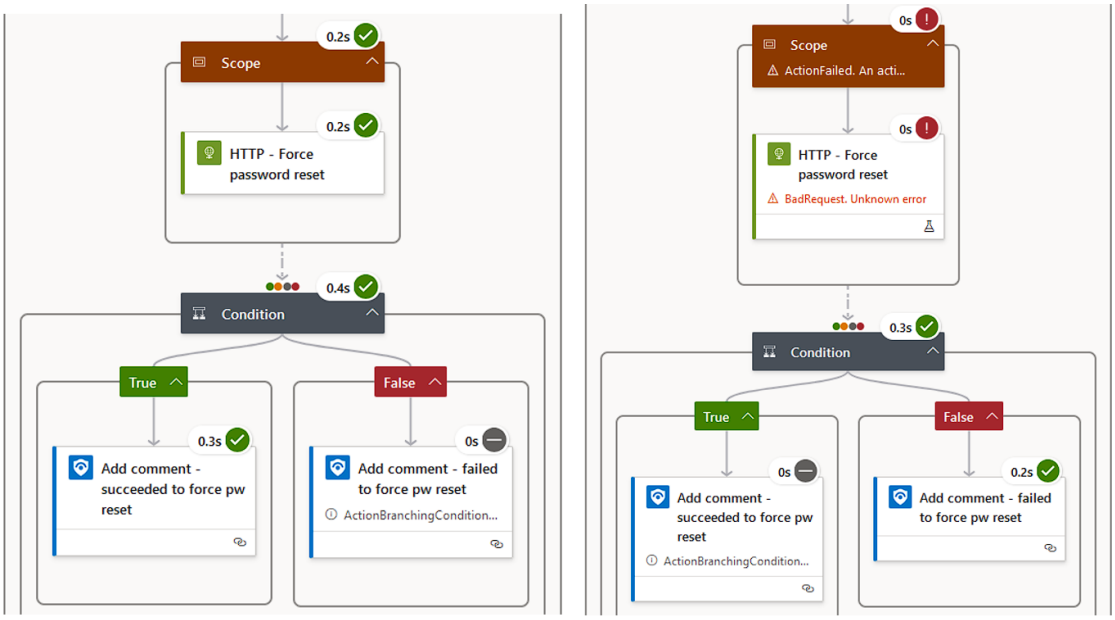
Kuva 24. Playbookin suoritus eteni False-polulle, vaikka sen olisi pitänyt mennä True-polulle

Vianselvitys pelkästä Microsoftin dokumentaatiosta ei tuottanut tulosta, vaan ratkaisua piti etsiä muualtakin verkosta. Lopulta pitkän tutkimisen päätteeksi ratkaisu löytyi Wilsonin blogiartikkelista (2022), jonka mukaan lausekkeella ”result('Scope')[0]['status']” saadaan Scopen tulos mukaan ehtolausekkeeseen. Tätä testatessa päätettiin myös vaihtaa ehtolauseke siten, että tuloksena on ”True” jos Scopen tulos on ”Succeeded”. Nämä molemmat muutokset näkyy kuvassa 25.



Kuva 25. Korjattu ehtolauseke

Tällä muutoksella Scope ja Condition saatiin vihdoin toimimaan oikein, kuten näkyy kuvassa 26.



Kuva 26. Vasemmalla suoritus onnistuneen ja oikealla epäonnistuneen salasanaresetoinnin jälkeen

Sama metodi lisättiin seuraavaksi myös playbookin istuntojen katkaisun API-kutsulle. Koska salasanan vaihdon pakotusta seurasi nyt kaksi vaihtoehtoista polkua, niin täytyi molemmille poluille lisätä oma istuntojen katkaisun API-kutsu. Tämä onnistuu playbookilla helposti, sillä toimintoja pystyy suoraan

kopioimaan ja liittämään graafisessa käyttöliittymässä. Lopputulemana syntyi siis neljä erillistä polkua, joille kaikille muotoiltiin omanlaisensa kuvaava kommentti, joka Sentinel-tapahtumalle lisätään (kuva 27).

Kuva 27. Playbookin konditiot ja jokaista skenaariota kuvaavat kommentit

Jokaista polkua testattiin vuorotellen staattisen tuloksen asettelun avulla, ja kaikki polut toimivat odotetusti.

4.9 Parent ja Child Logic Appit

Scope-toimintoon perehtyessä heräsi ajatuksia, että voisiko olla hyödyllistä jakaa tiettyjä toimintoja omiksi playbookeikseen, joita muut playbookit voisivat tarvittaessa kutsua? Tällaisella modulaarisella lähestymisellä voitaisi sulavoittaa playbookien kehitystä, sillä kun tekisi jollekin usein tarvitulle yksittäiselle toiminnolle oman playbookinsa, ei sitä toimintoa tarvitsisi konfiguroida joka kerta uudestaan uusia playbookeja luotaessa. Sen sijaan uudet playbookit voivat vain kutsua tätä yhtä playbookia, joka on valmiiksi konfiguroitu muun muassa tarvittavine oikeuksineen ynnä muine asetuksineen.

Asiaan löytyi hyödyllistä tietoa sekä Microsoftin dokumentaatiosta (Azure Logic Apps 2024a) että Rainan blogiartikkelista (2021), jossa tätä prosessia

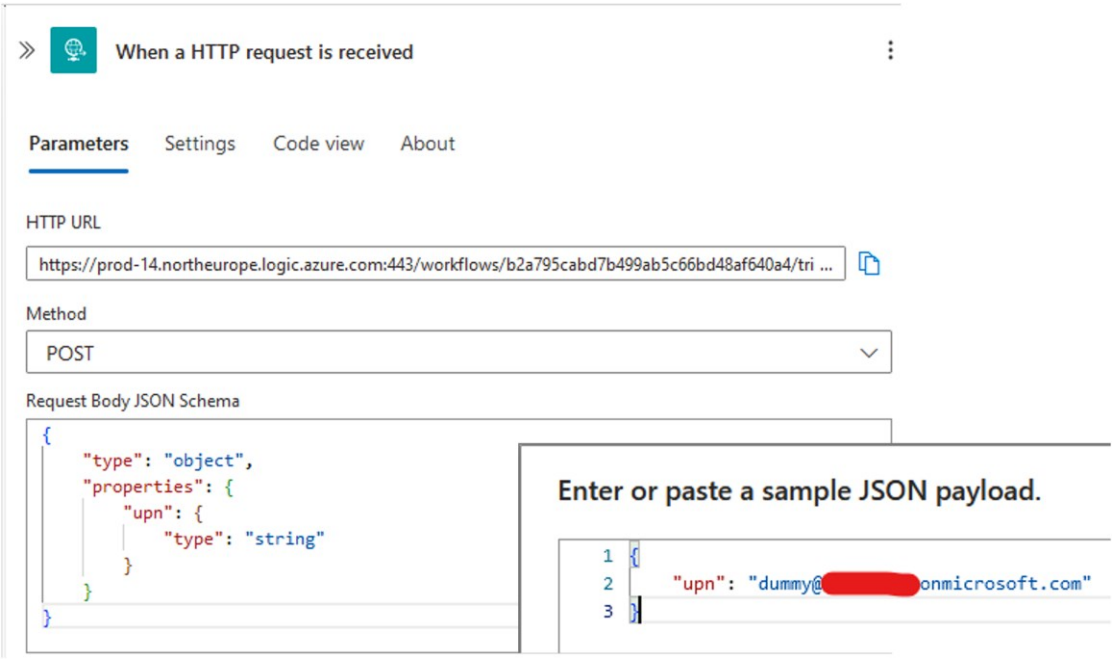
kuvailtiin osuvasti niin sanottujen Parent ja Child Logic Appien käsitteillä. Parent Logic App olisi esimerkiksi Sentinel-tapahtumasta käynnistynyt playbook, joka kutsuisi suorituksensa aikana Child Logic Appia, joka puolestaan käynnistyisi kyseisestä kutsusta. Child Logic Appin laukaisimen tyyppi olisi tässä tapauksessa http-pyyntö, jolle on Logic Appissa oma toimintonsa nimeltä ”When a HTTP request is received”. Kyseinen toiminto luo muiden laukaisimien tavoin itselleen niin sanotun callback URL -osoitteen, jota muut Logic Appit voivat kutsua. Ero Sentinelin omiin laukaisimiin on siinä, että tätä kyseistä URLia kutsuessa voidaan määrittää, mitä dataa sille lähetetään, siinä missä esimerkiksi Sentinelin entiteetin laukaisin kerää vain tiedot Sentinelin entiteetistä, eikä se ole muutettavissa.

Esimerkki käyttötapauksesta:

- Playbook 1:
  - Laukaisin: HTTP request
  - Toiminto: Katkaisee istunnot API-kutsulla
  - Oikeudet: ”User.RevokeSessions.All” application permission
- Playbook 2:
  - Laukaisin: HTTP request
  - Toiminto: Pakottaa salasanan vaihdon API-kutsulla
  - Oikeudet: ”Password Administrator” Entra rooli
- Playbook 3:
  - Laukaisin: Incident
  - Toiminto: Kutsuu playbookeja 1 ja 2, sekä lisää Sentinel-tapahtumaan kommentin
  - Oikeudet: ”Microsoft Sentinel Responder” Azure rooli
- Playbook 4:
  - Laukaisin: Entity
  - Toiminto: Kutsuu playbookia 1
  - Oikeudet: Ei tarvetta erityisoikeuksille

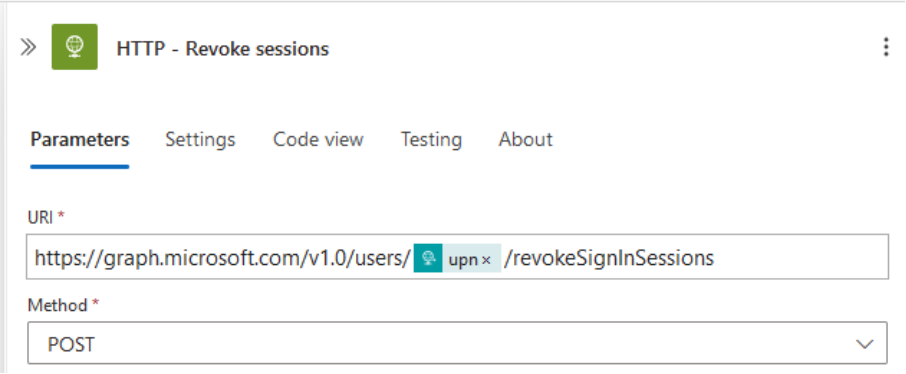
Asiaan perehtymisen jälkeen lähdettiin luomaan niin sanottua Childia, joka sisältäisi istuntojen katkaisun API-kutsun. Tällä olisi tarkoitus korvata kyseinen toiminto playbookissa ”Incident-RevokeSessions_ResetPassword-EntraID”, sekä tehdä sama myöskin salasanan vaihdon pakotuksen API-kutsun kohdalla. Näin ollen kyseisestä Sentinel-tapahtumasta laukaistavasta playbookista tulisi niin sanottu Parent, joka kutsuu kahta Childia. Molemmalle Childille asetettaisiin vain niiden tarvitsemat oikeudet, ja Parentilta kyseiset oikeudet voitaisiin ottaa pois.

Sentinelissä playbookia luodessa on mahdollista valita optio ”Blank playbook”, jolla voisi aloittaa playbookin luomisen täysin tyhjältä pöydältä, mutta koska sen kanssa tarvitsee alkuun päästäkseen konfiguroida enemmän asioita, niin on helpompi vain valita esimerkiksi ”Playbook with incident trigger” ja korvata editointinäkymässä tapahtumatyypin laukaisin toisella laukaisimella. Näin päätettiin tehdä, ja playbookin luomisen jälkeen laukaisimeksi asetettiin ”When a HTTP request is received”. Sen parametreihin täytyi määrittää, missä muodossa se vastaanottaa datan sekä millä http-metodilla. Datan saa käännettyä oikeaan JSON-muotoon antamalla mallin sisällöstä (sample payload). Koska tiedettiin, että tässä tarkoituksena on saada käyttäjän UPN (tai ID) istunnot katkaisevaan API-kutsuun, niin sisällön malliksi voitiin laittaa testikäyttäjän UPN. Sen myötä kenttä ”Request Body JSON Schema” täyttyi, ja tämä playbook oli nyt valmis vastaanottamaan dataa arvolle ”upn”. Kuvassa 28 näkyy edellä mainitut JSON-sisällön malli, sekä siitä koostettu ”Request Body JSON Schema”.



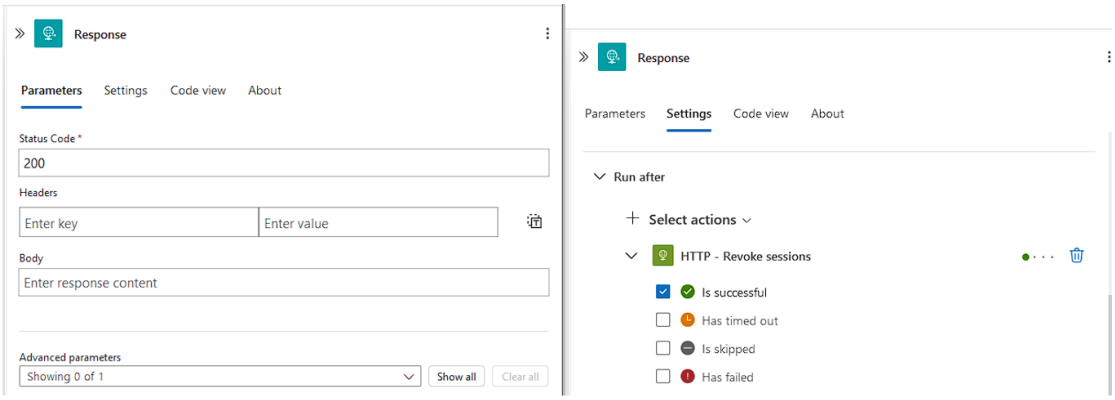
Kuva 28. Oikealla alhaalla annettu malli sisällöstä ja vasemmalla mallin myötä täytetty data

Playbookin seuraavassa vaiheessa lisättiin istunnot katkaiseva API-kutsu, jossa käytetään muuttujaa ”upn” käyttäjän kohdalla (kuva 29).



Kuva 29. Istunnot katkaisevan API-kutsun URI-osoitteeseen lisätty muuttuja "upn"

Viimeiseksi toiminnoksi playbookissa laitettiin ”Response”, johon asetettiin vain http-koodi ”200” vastaukseksi, jonka ”Run after” -asetuksiin jätettiin vakioasetus eli suoritetaan vain, jos edeltävä toiminto on suoriutunut onnistuneesti. Tällä tavalla tämä Child lähettää Parentille onnistumisesta indikoivan 200-koodin vain, jos edeltävä istunnot katkaiseva API-kutsu on ollut onnistunut. Nämä asetukset näkyy kuvassa 30.



Kuva 30. Vasemmalla "Response"-toiminnon 200-koodin asetus ja oikealla "Run after" -asetus

Lopuksi playbookin hallinnoidulle identiteetille lisättiin sen tarvitsema ”User.RevokeSessions.All”-oikeus. Tämän jälkeen Parentilta poistettiin Scope 1:n alta istunnot katkaiseva API-kutsu ja korvattiin se Childin kutsumisella. Parentin ajoa testattiin, ja sekä Parent että Child suoriutuivat onnistuneesti. Sen myötä samalla kaavalla luotiin Child myös salasanaresetoinnin API-kutsulle, ja korvattiin loputkin Parentin Scope-toimintojen sisällöt Childien kutsumisilla. Kuvassa 31 näkyy valmis playbook kokonaisuudessaan.

Kuva 31. Valmis Parent-playbook, johon määritelty Scope-toimintojen sisälle Child-playbookien kutsumiset

Enää täytyi poistaa turhat oikeudet Parentilta eli ”User.RevokeSessions.All” sekä Password Administrator, sillä nämä oikeudet olivat nyt lisättynä vastaaville Childeille. Parent tarvitsi enää vain Microsoft Sentinel Responder -roolijaon Azuren built-in rooleista Sentinel-tapahtumien kommenttien lisäämiseen. Kun oikeudet oli poistettu, oli aika jälleen testata playbookia.

Suoritus epäonnistui, mutta syyksi osoittautui jälleen kauttaviivan puuttuminen API-kutsun URI:sta. Näin voi käydä herkästi, sillä kauttaviivalla voi myös aloittaa funktion lisäämisen, joten URI-osoitteessa oleva kauttaviiva päätyy helposti funktion aloittajaksi, jos ei ole huolellinen. Kirjoitusvirheen korjauksen jälkeen playbookia testattiin uudelleen, jonka seurauksena sekä Parentin että Childien toiminnot näyttivät sujuneen hyvin. Myös Dummy-käyttäjällä eteni asiat odotetusti portaalissa, eli uloskirjautuminen -> salasana -> MFA -> salasanan vaihto. Ja kommenttien lisäykset Sentinel-tapahtumaan toimivat myös oikein. Näin oli saatu kehiteltyä ja testattua myös Parent ja Child Logic Appien konsepti käytännössä.

5 KENTTÄPÄIVÄKIRJAN ANALYSOINTI

Kenttäpäiväkirja tehtiin Word-tiedostoon ja sinne kuvattiin yksityiskohtaisesti kaikki kehitystyön vaiheet kuvankaappauksia hyödyntäen. Kehitystyöhön sisältyi paljon eri asioihin perehtymistä, joten kenttäpäiväkirjassa myöskin selitettiin olennaisia käsitteitä sitä mukaa, kun kehitystyössä kyseisiin käsitteisiin perehdyttiin, sekä lisättiin linkkejä lähteisiin.

Kenttäpäiväkirjan sisältö analysoitiin laadullisen sisällönanalyysin menetelmiä soveltaen, ja analyysin tavoitteena oli määritellä tutkimusongelman ja -kysymysten valossa olennainen sisältö ohjekirjaan. Ennen analysoinnin aloittamista kenttäpäiväkirja jäsenneltiin helppolukuiseen muotoon ja siihen perehdyttiin lukemalla sitä ajatuksella läpi, sekä verrattiin sen sisältöä tutkimuskysymyksiin. Siten varmistettiin, ettei kehityksessä ja testauksessa jäänyt mitään olennaista tekemättä. Tämä oli ns. valmisteluvaihe ennen analysointia ja se on olennainen osa sisällönanalyysin prosessia (Elo ym. 2022, 219).

Analyysivaihe alkoi kenttäpäiväkirjan sisällön luokittelulla ohjekirjan kannalta mielekkäisiin kategorioihin, joista puolestaan pyrittiin teemoittelemalla löytämään toistuvia piirteitä. Tässä olennaista on, ettei teemoja yritä määrittää etukäteen, vaan teemat nimenomaan analyysin aikana yritetään tunnistaa aineistosta (Kallinen & Kinnunen 2021). Luokittelun myötä muodostui kolme kategoriaa, joista jokainen visualisoitiin taulukoiksi, joiden avulla tunnistettiin ala- ja pääteemat ohjekirjaan lisättävälle sisällölle.

5.1 Luokittelu ja teemoittelu

5.1.1 Kategoria: Playbookien epäonnistuneet suoritukset ja niiden korjaukset

Ensimmäiseksi kategoriaksi määriteltiin ”Playbookien epäonnistuneet suoritukset ja niiden korjaukset”. Taulukkoon 1 on kerättynä jokainen testauksen aikana esiintynyt epäonnistunut playbookin suoritus, sekä sille tehty korjaava toimenpide.

Taulukko 1. Playbookien epäonnistuneet suoritukset ja niiden korjaukset

Playbookin nimi	Epäonnistumisen syy	Korjaus
Revoke-EntraIDSignin-Session-entityTrigger	Riittämättömät oikeudet suorittaa kutsu Graph API-päätepisteeseen "/revokeSignInSessions"	Oikeuden "User.RevokeSessions.All" lisääminen playbookin hallinnoidulle tunnukselle
Revoke-EntraIDSignin-Session-entityTrigger	Riittämättömät oikeudet suorittaa kutsu Graph API-päätepisteeseen "/revokeSignInSessions"	Playbookin uudelleen ajaminen
Revoke-EntraIDSignin-Session-entityTrigger	Riittämättömät oikeudet lisätä kommentti Sentinelin incidentille	Microsoft Sentinel Responder -roolin jakaminen playbookin hallinnoidulle tunnukselle
Incident-RevokeSessions_ResetPassword-EntraID	Kirjoitusvirhe toiminnon "HTTP - Force password reset" URI-osoitteessa; kauttaviiva puuttui polusta users-kansion ja käyttäjän UPN:n väliltä	Kauttaviivan lisäys URI-osoitteeseen users-kansion ja käyttäjän UPN:n väliin
Incident-RevokeSessions_ResetPassword-EntraID	Scope-toiminnon jälkeisessä Condition-toiminnossa käytettyä funktiota nimeltä "action" ei tunnistettu funktioksi	Funktion "action" korjaus muotoon "actions"
Incident-RevokeSessions_ResetPassword-EntraID	Scope-toiminnon jälkeisessä Condition-toiminnossa käytetty funktio nimeltä "actions" ei toiminut oikein; suoritus ajautui Condition-toiminnon False-polulle riippumatta Scope-toiminnon tuloksesta	Funktion "actions" korvaus funktiolla "result". Koko ehtolausekkeen rakenne muutettiin: actions('Scope') is equal to Failed or Aborted -> result('Scope')[0]['status'] is equal to Succeeded
ForcePasswordReset-EntraID	Kirjoitusvirhe toiminnon "HTTP - Force password reset" URI-osoitteessa; kauttaviiva puuttui polusta users-kansion ja käyttäjän UPN:n väliltä	Kauttaviivan lisäys URI-osoitteeseen users-kansion ja käyttäjän UPN:n väliin

Taulukkoon kerättyjen tietojen perusteella epäonnistumisten syyt ryhmiteltiin kolmeen alateemaan, joille jokaiselle määriteltiin huomioitavat seikat:

- Riittämättömät oikeudet
  - Perehdy ennalta Azuren oikeus- ja roolityyppeihin
- Kirjoitusvirhe Graph API-kutsun URI-osoitteessa
  - Tarkista funktiota lisättäessä, että kirjoitusmuoto on oikein
- Scope-toiminnon jälkeisessä Condition-toiminnossa käytetty funktio
  - Käytä lauseketta result(’Scope’)[0][’status’]

Nämä alateemat määriteltiin pääteemaksi ”Yleiset virheet”. Siihen kuuluu vältettävissä olevia virheitä, joiden huomioitavat seikat on syytä mainita ohjekirjassa, jotta ohjekirjan käyttäjät välttyvät kompastumasta niihin.

5.1.2 Kategoria: Ongelmatilanteet ja niiden ratkaisut

Seuraavaksi kategoriaksi kenttäpäiväkirjan sisällössä tunnistettiin ”Ongelmatilanteet ja niiden ratkaisut”, johon poimittiin kehityksen ja testauksen vaiheista sellaisia tilanteita, jotka tuottivat muihin vaiheisiin nähden jollain tapaa poikkeuksellisia ongelmia, ja joiden selvittely oli joko aikaa vievää tai muuten epäselvää (taulukko 2).

Taulukko 2. Ongelmatilanteet ja niiden ratkaisut

Ongelmatilanne	Haasteet	Ratkaisu
Applikaatio-oikeuden lisääminen playbookin hallinnoidulle identiteetille	Microsoftin dokumentaatiosta ei löytynyt yksiselitteistä vastausta, kuinka tämä tehdään. Tätä ei pysty tekemään Azuren portaalista selaimella, toisin kuin useimmat muut asiat Azuressa. Verkkolähteistä löytyneissä ohjeissa olevat skriptit käyttivät deprekoitunutta Azure AD PowerShell -moduulia. Vaati aikaa perehtyä PowerShelliin ja kehitellä eri ohjeita yhdistelemällä oma skripti.	Käytettiin mallina Anssi Päivisen blogiartikkelissa kuvattua skriptiä ja muutettiin se Microsoft Graph PowerShell -moduulia käyttävään muotoon Microsoftin dokumentaatiota apuna käyttäen.
Istuntojen katkaisun API-kutsun epäonnistunut suoritus tuntemattomasta syystä	Vaikka playbookilla oli samat oikeudet kuin edeltäneellä onnistuneella suorituskerralla, niin playbookin suoritus epäonnistui ja ilmoitti virheviestissä syyksi puutteelliset oikeudet suorittaa istuntojen katkaisun API-kutsu.	Uudella yrityksellä playbookin ajo onnistui, ilman tehtyjä muutoksia playbookiin. Syytä epäonnistumiselle ei selvinnyt. Ongelman havaitseminen korosti tarvetta virheiden ja poikkeamien käsittelylle Scope-toiminnolla.
Matalimpien tarvittavien oikeuksien selvittäminen	Istuntojen katkaisun osalta Sentinelin valmiiden playbook-mallien ohjeissa neuvottiin antamaan korkeammat oikeudet, kuin mikä käytännössä riitti. Puolestaan salasanaresetoinnin kohdalla Microsoftin dokumentaatiossa neuvottiin antamaan korkeammat oikeudet, kuin mikä käytännössä riitti. Matalimpien tarvittavien oikeuksien löytäminen haastavaa, kun luotettavaa lähdettä ei löydy.	Matalimmat tarvittavat oikeudet selvisivät käytännön testaamisen kautta.
Salasanan vaihdon pakotuksen API-kutsun poikkeava toiminta	Joskus käyttäjä pakotettiin kirjautumaan uudestaan ja joskus ei. MFA pyydetään käyttäjältä riippumatta siitä, asetetaanko "forceChangePasswordNextSignInWithMfa" arvoon "true" vai "false"	Luotiin playbook, joka tekee erillisillä API-kutsuilla sekä istuntojen katkaisun, että salasanan vaihdon pakottamisen
Scope-toiminnon jälkeisen ehtolausekkeen muodostaminen	Microsoftin dokumentaatiosta ei löytynyt toimivaa tapaa kerätä Scope-toiminnon tulosta sen jälkeiselle ehtolausekkeelle. Microsoftin tutoriaalissa käytetty funktio ei ollut toiminnallinen käytännössä. Myös muista verkkolähteistä toimivan tavan löytäminen oli haastavaa.	Andrew Wilsonin artikkelista löytyi toimiva lauseke tarkoitukseen.

Ongelmatilanteissa esiintyneitä haasteita tarkastellessa tunnistettiin kaksi alateemaa, sekä niiden osalta huomioitavat seikat:

- Microsoftin dokumentaation epäluotettavuus
    - Vaatii validointia käytännön testauksella
- Playbookin toiminnan epäluotettavuus
    - Vaatii virheiden ja poikkeamien käsittelyä

Näille alateemoille pääteemaksi nimettiin ”Huomioitavat haasteet”. Sillä kuvataan haasteita, jotka eroavat ensimmäisestä pääteemasta ”Yleiset virheet” siten, että niitä ei varsinaisesti kykene ennakoimalla kokonaan ohittamaan. Mutta niiden negatiivinen vaikutus voidaan kuitenkin minimoida niiden huomioitavat seikat tiedostamalla.

5.1.3 Kategoria: Työn aikana kehitetyt ratkaisut

Kolmanneksi kategoriaksi valittiin ”Työn aikana kehitetyt ratkaisut”. Tähän kategoriaan poimittiin kaikki sellaiset konkreettiset asiat, jotka kehityksen ja testauksen vaiheen aikana kehitettiin ja jotka todettiin relevanteiksi ohjekirjan kokoamisen kannalta. Playbookien osalta Sentinelin valmiista malleista suoraan luotuja playbookeja ei luettu osaksi tätä kategoriaa, vaan mukaan otettiin ainoastaan työn aikana alusta loppuun asti tehdyt playbookit. Taulukossa 3 on kuvattu jokainen kehitetty ratkaisu, sekä syyt niiden kehitykselle.

Taulukko 3. Työn aikana kehitetyt ratkaisut

Kehitetty ratkaisu	Kuvaus	Syy kehitykselle
Analytiikkasääntö: Dummy signs in	Luo Sentineliin incidentin testikäyttäjän kirjautumisesta.	Playbookien ja automaatiosääntöjen testaaminen.
Automaatiosääntö: Revoke sessions and reset password on Entra ID	Käynnistyy Sentinelin incidentistä, jonka otsikko on "Dummy signs in". Käynnistää playbookin "Incident-RevokeSessions_ResetPassword-EntraID", sekä päivittää incidentin tietoja viidellä muulla toiminnolla.	Automaatiosäännön toimintojen testaaminen ja demonstrointi.
Nimeämiskäytäntö	(Asiakastunniste)-Laukaisintyyppi-Päätoiminto-(Kohdealusta/laajuus)	Selkeyttää playbookien järjestelyä.
Playbook: Incident-RevokeSessions_ResetPassword-EntraID	Käynnistyy Sentinelin incidentistä. Katkaisee incidentin käyttäjien istunnot ja pakottaa salasanan vaihdon Entra ID:ssä, kutsumalla kahta muuta Logic Appia "RevokeSessions-EntraID" ja	Playbookin toimintojen testaaminen ja demonstrointi.
Playbook: ForcePasswordReset-EntraID	Käynnistyy HTTP-kutsusta. Tekee HTTP-kutsun Graph API:ssa käyttäjän tietoihin, joka pakottaa käyttäjän vaihtamaan	Parent/Child -konseptin testaaminen playbookeilla ja demonstrointi.
Playbook: RevokeSessions-EntraID	Käynnistyy HTTP-kutsusta. Tekee HTTP-kutsun Graph API-päätepisteeseen, joka katkaisee käyttäjän istunnot.	Parent/Child -konseptin testaaminen playbookeilla ja demonstrointi.
Playbook-metodi: Parent/Child -konseptin hyödyntäminen	Tiettyjä erityisoikeuksia vaativien toimintojen siirtäminen erillisiksi playbookeiksi, joita muut playbookit voivat tarvittaessa kutsua.	Modulaarisuuden lisääminen playbookien kehitykseen, toistuvan konfiguroinnin vähentämiseksi.
Playbook-metodi: Virheiden ja poikkeamien käsittely Scope-toiminnolla	Scope-toiminnon sisälle laitetaan toiminto, ja Scope tuottaa tuloksen	Jottei playbookin yksittäiset virheet ja poikkeamat kaada koko playbookin
PowerShell-skripti: oikeuden lisääminen hallinnoidulle identiteetille	Lisää applikaatio-oikeuden playbookin hallinnoidulle identiteetille.	Helpottaa tekemistä, kun on valmis skripti saatavilla.
PowerShell-skripti: oikeuden poistaminen hallinnoidulta identiteetiltä	Poistaa applikaatio-oikeuden playbookin hallinnoidulta identiteetiltä.	Helpottaa tekemistä, kun on valmis skripti saatavilla.
PowerShell-skripti: oikeuksien tarkistaminen hallinnoidulta identiteetiltä	Tarkistaa applikaatio-oikeudet playbookin hallinnoidulta identiteetiltä.	Helpottaa tekemistä, kun on valmis skripti saatavilla.

Tästä kategoriasta ei koettu tarpeelliseksi muodostaa teemoja. Ratkaisut itsessään toimivat runkona ohjekirjan läpikäyntiohjeille. Testauksen kautta ne todettiin toimiviksi ja siten luotettaviksi esimerkkiratkaisuiksi ohjekirjaan.

6 OHJEKIRJAN KOKOAMINEN

Ohjekirja koottiin Word-tiedostoon. Kenttäpäiväkirjan analyysin pohjalta lähdettiin aluksi hahmottamaan ohjekirjan sisältöä. Analyysissä määritettiin teemat ”Yleiset virheet” ja ”Huomioitavat haasteet”, jotka tulisi sisällyttää ohjekirjaan jollain tapaa. Läpikäyntiohjeiden runkona määritettiin olevan ”Työssä kehitetyt ratkaisut”. Tässä vaiheessa päätettiin, että ”Yleiset virheet” voidaan myös sisällyttää läpikäyntiohjeisiin; sen alateemat sekä niiden osalta

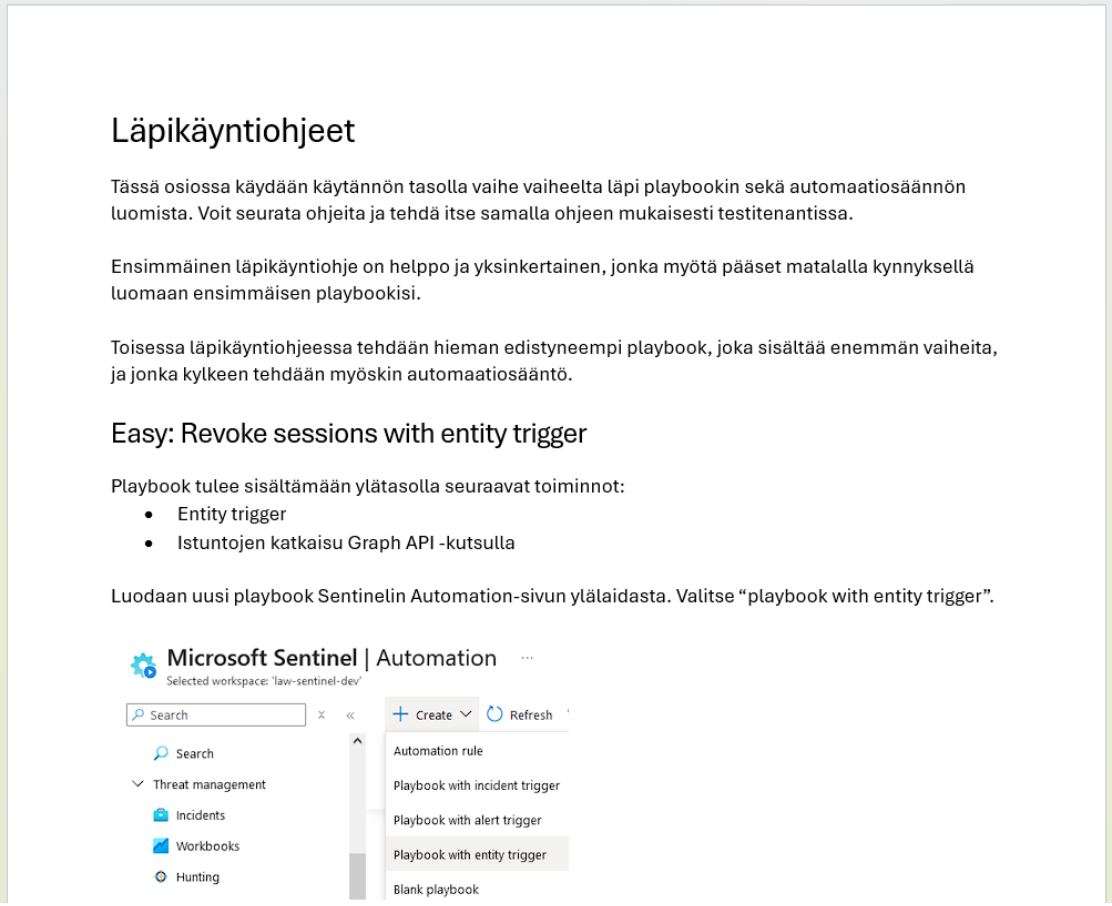
huomioitavat seikat voidaan mainita läpikäyntiohjeessa niitä koskettavissa kohdissa. Tämän myötä ohjekirjan sisältö tulisi sisältämään ainakin seuraavanlaiset rakenteet:

- Läpikäyntiohjeet
    - Työn aikana kehitetyt ratkaisut
    - Yleiset virheet
- Huomioitavat haasteet

Tästä lähtökohdasta aloitettiin ohjekirjan kasaaminen läpikäyntiohjeiden tekemisellä.

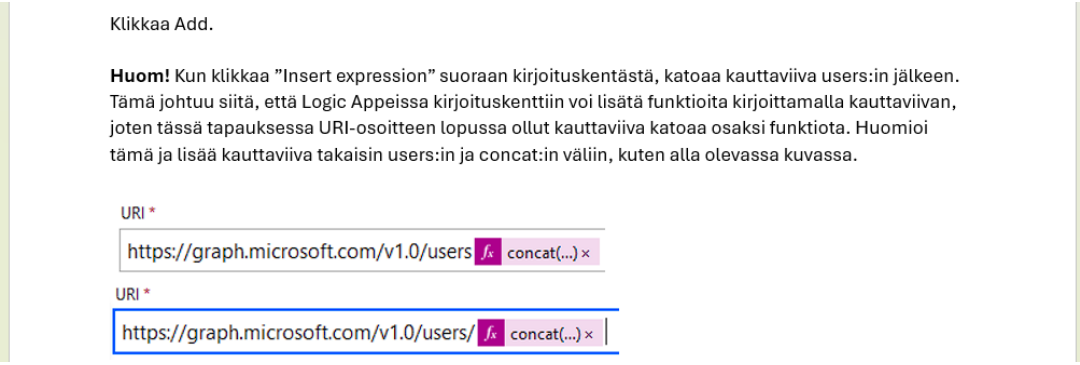
6.1 Läpikäyntiohjeet

Alun perin oli suunniteltu, että ohjekirjaan tulisi vain yksi läpikäyntiohje, mutta tässä vaiheessa päätettiinkin, että tehdään kaksi ohjetta; yksi helpompi läpikäyntiohje, jonka avulla ohjekirjan käyttäjä pääsee matalammalla kynnyksellä tekemään ensimmäisen playbookinsa (”Easy: Revoke sessions with entity trigger”), sekä monivaiheisempi versio, johon sisältyisi työn aikana kehitetyistä ratkaisuista suurempi osa (”Advanced: Revoke sessions and force password reset with incident trigger, along with an automation rule”). Vain yksi kaiken sisältävä läpikäyntiohje olisi monelle todennäköisesti liian työläs lähestyttävä, jolloin ohjekirja epäonnistuisi siirtämään tietotaitoa käytäntöön. Kuvassa 32 on katkelma ohjekirjasta läpikäyntiohjeiden ensimmäiseltä sivulta, jossa näkyy myöskin helpomman ohjeen alkuvaihetta.



Kuva 32. Katkelma ohjekirjan Läpikäyntiohjeet-osion ensimmäiseltä sivulta, jossa näkyy myös helpomman ohjeen alkuvaihetta

Läpikäyntiohjeisiin sisällytettiin kenttäpäiväkirjan analyysista tunnistettuja työn aikana kehitettyjä ratkaisuja, sekä sopiviin kohtiin huomautuksia yleisistä virheistä, kuten kuvassa 33 esitetty huomautus. Ohjeisiin jäi lopulta lisäämättä Parent/Child Logic Apps -konseptin demonstrointi, sillä ”Advanced”-ohjeesta alkoi tulla ilman sitäkin jo melko pitkä, ja aikarajoitteiden takia ei kolmatta läpikäyntiohjettä ehditty tehdä. Ohjekirjaa voi kuitenkin jatkossa päivittää ja lisätä sinne lisää läpikäyntiohjeita.



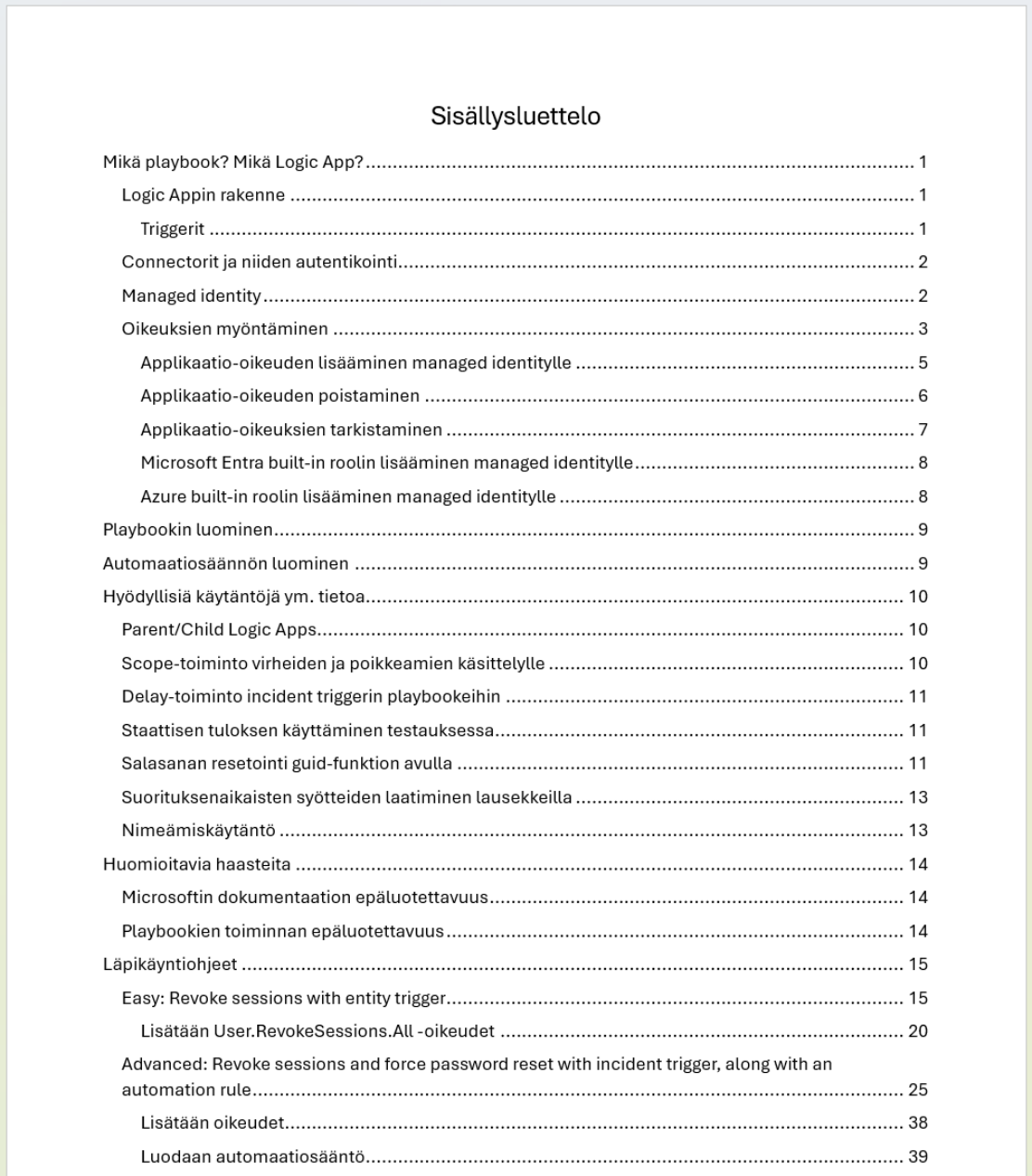
Kuva 33. Katkelma ohjekirjan Läpikäyntiohjeet-osiosta, jossa näkyy huomautus yleisestä virheestä

6.2 Muu sisältö

Kun läpikäyntiohjeet oli tehty, siirryttiin määrittämään, että mitä muuta täydennystä ohjekirja kaipaisi läpikäyntiohjeiden tueksi. Ohjeet tehtiin siten, että niitä täsmälleen seuraamalla voi luoda playbookin ja automaatiosäännön, vaikkei ymmärtäisi mitään niiden toiminnasta, mutta tämä ei yksinään riitä opettamaan playbookien ja automaatiosääntöjen luomista ja hyödyntämistä yleisesti. Ohjekirjaan tarvittiin myös selityksiä olennaisista käsitteistä, jotta tietotaito todella välittyisi. Myöskin kenttäpäiväkirjan analyysissä tunnistettu teema ”Huomioitavat haasteet” olisi syytä mainita erikseen läpikäyntiohjeiden ulkopuolella ohjekirjassa. Ohjekirjassa selitettäviksi käsitteiksi tunnistettiin:

- Logic Appien rakenne
    - Workflow Definition Language
    - Logic Appien laukaisintyypit
    - Suorituksenaikaiset lausekkeet
- API-yhteyksien autentikointi
- Managed identity
- Erityyppiset oikeudet ja niiden myöntäminen
- Virheiden ja poikkeamien käsittely (Scope)
- Parent/Child Logic Apps
- Nimeämiskäytäntö
- Testaus staattisella tuloksella

Ohjekirjan sisällön kirjoittamista helpotti se, että jo kehitystyön aikana oli kenttäpäiväkirjaan lisätty monien olennaisten käsitteiden selityksiä, joten suuri osa tästä sisällöstä oli jo valmiiksi kirjoitettu. Sisällön kirjoittamisen jälkeen ohjekirjaan lisättiin sisällysluettelo (kuva 34) ja kansilehti otsikolla ”Ohjekirja Sentinelin playbookien ja automaatiosääntöjen kehitykselle”. Lopuksi jäsenneltiin ohjekirjan osiot loogiseen järjestykseen ja muotoiltiin sisältö käyttökelpoiseksi. Tämän myötä ohjekirja saatiin valmiiksi, jonka jälkeen se lisättiin SOCin dokumentaatioalustalle.



Sisällysluettelo

Kuva 34. Ohjekirjan sisällysluettelo

7 TULOKSET

Työn alussa määriteltiin, että ohjekirjan tavoitteena oli ratkaista tutkimusongelma eli automaation hyödyntämisen puute Microsoft Sentinelin hälytysten käsittelyssä SOCissa, ja siihen tavoitteeseen pääsemiseksi ohjekirjan sisällöllä tulisi kyetä vastaamaan tutkimuskysymyksiin. Tarkastellaan siis saavutettuja tuloksia tutkimuskysymysten kannalta.

- Miten Microsoft Sentinelin automaatioratkaisuja hälytysten käsittelyyn voidaan kehittää?

Työn aikana onnistuttiin selvittämään kehittämisen ja testaamisen avulla, kuinka hälytyksiä käsitteleviä automaattisia toimenpiteitä voidaan playbookeilla ja automaatiosäännöillä kehittää Microsoft Sentinelissä. Nämä keinot on kuvattu ohjekirjassa läpikäyntiohjeissa, sekä niitä tukevissa käsitteiden selityksissä.

- Millaisia automaatioratkaisuja hälytysten käsittelyyn Microsoft Sentinelissä voidaan kehittää?

Ohjekirjassa demonstroidaan useita ratkaisuja automaattiseen hälytysten käsittelyyn playbookien ja automaatiosääntöjen avulla, jotka sisältävät toimintoja kuten käyttäjän istuntojen katkaisu, salasanan resetointi, sekä Sentinel-tapahtumaan kommentin lisääminen. Kaikki ohjekirjaan kerätyt ratkaisut todettiin testauksen kautta toimiviksi.

- Kuinka Microsoft Sentinelin automaatioratkaisujen kehittämisen tietotaitoa voidaan kasvattaa SOCissa?

Ohjekirjaan tehtiin yksityiskohtaiset läpikäyntiohjeet kuvankaappausten havainnollistamana, joita askel askeleelta seuraamalla kuka tahansa SOCin työntekijä kykenee luomaan toimivaksi todettuja automaatioratkaisuja Sentinelissä. Läpikäyntiohjeista tehtiin myöskin kaksi versiota, joista yksinkertaisemman ohjeen on tarkoitus madaltaa kynnystä aloittaa playbookien teko, ja monivaiheisempi ohje puolestaan demonstroi playbookien ja automaatiosääntöjen toiminnallisuuksia laajemmin. Lisäksi ohjekirjassa selitettiin olennaisia käsitteitä sekä muita huomioitavia seikkoja Sentinelin automaatioratkaisujen kehityksessä. Tällä yhdistelmällä on todennäköistä saavuttaa kyseisen kehityksen tietotaidon onnistunut levitys SOCin henkilöstössä.

Tutkimuskysymyksiin siis onnistuttiin vastaamaan ohjekirjan sisällöllä.

8 JOHTOPÄÄTÖKSET

Interventionistisena tutkimuksena työn tavoitteena oli saavuttaa muutos, joka tässä tapauksessa oli automaation hyödyntämisen puutteen parantaminen Microsoft Sentinelin hälytysten käsittelyssä SOCissa. Tällaisen muutoksen

toteutuminen vaatisi pitempää tarkastelujaksoa; täytyy antaa aikaa tietotaidon siirtymiselle ohjekirjasta SOCin henkilöstöön, sekä sen jälkeen omaksutun tietotaidon pitäisi vielä siirtyä käytäntöön SOCin asiakasympäristöihin toteutuneina ratkaisuina.

Ohjekirjan sisältöä tutkimuskysymysten valossa arvioimalla voidaan kuitenkin todeta, että ohjekirjasta onnistuttiin tekemään tarpeeksi kattava ja lähestyttävä, jotta tavoiteltuun muutokseen tullaan sen avulla todennäköisesti jatkossa pääsemään. Näin ollen tavoitteisiin pääasiallisesti päästiin, vaikkei tutkimusongelman todellista ratkeamista päästykään vielä tutkimuksen aikana seuraamaan.

9 POHDINTA

Opinnäytetyö oli pääosin onnistunut. Työn aikana onnistuttiin selvittämään, kuinka Sentinelin automaatioratkaisuja tehdään playbookeilla ja automaatiosäännöillä, ja sen pohjalta saatiin koostettua pitkälti sen kaltainen ohjekirja, kuin mitä työn alussa suunniteltiin. Tutkimusongelman olisi ehkä voinut alussa muotoilla konkreettisemmaksi tavoiteltavaksi muutokseksi, sillä ohjekirjan todellista vaikutusta määriteltyyn tutkimusongelmaan ei vielä tämän työn aikana selvinnyt. Ohjekirjan vaikutusta olisi pitänyt jäädä vielä tarkastelemaan pidemmäksi aikaa, ja analysoida sitä esimerkiksi ohjekirjan käyttäjien haastatteluiden pohjalta ja/tai tarkastelemalla tilastollisesti muutosta manuaalisesti käsiteltävien hälytysten määrässä ohjekirjan lisäämisen jälkeen. Siinä voisi olla aihetta jatkotutkimukselle.

Ohjekirjassa kuvattujen ohjeiden toimivuuteen voidaan luottaa, sillä siinä kuvatut automaatioratkaisut testattiin käytännössä toimiviksi ja nämä testaukset dokumentoitiin yksityiskohtaisesti kenttäpäiväkirjaan, jonka sisällöstä puolestaan laadullisen analyysin avulla koostettiin tarvittava sisältö ohjekirjaan. Tutkimuksen päätyttyä ilmaan jääneitä kysymyksiä ovat kuitenkin edelleen, että kuinka hyvin ohjekirja otetaan vastaan ja kuinka moni SOCin työntekijä sitä oikeasti tulee hyödyntämään. Tutkimuksen tulosten luotettavuutta voidaan näin ollen pitää vain korkeintaan kohtalaisena.

Suurimpia haasteita ohjekirjan tekemisessä oli keskeisten asioiden tiivistäminen ohjekirjaan ilman, että siitä tulisi liian pitkä ja monimutkainen. Ohjekirjasta kokonaisuudessaan tuli lopulta melko mittava (43 sivua), mutta yli puolet sen sisällöstä koostuu läpikäyntiohjeista, jotka sisältävät paljon kuvia havainnollistamisen vuoksi. Ohjekirjaan lisätyssä sisällössä yritettiin olla menemättä liian teknisiin yksityiskohtiin, jotta siitä ei tulisi liian sekava ymmärtää. Tavoitteena oli saada perusasiat avattua yksinkertaisesti mutta riittävästi, jotta ohjekirjan perusteella SOCin työntekijä, joka ei ole aiemmin Sentinelin playbookeja tehnyt, oppisi niitä matalalla kynnyksellä tekemään. Tässä onnistuttiin arviolta melko hyvin, mutta arvion paikkansapitävyys selviää käytännössä vasta myöhemmin ohjekirjan käyttöönoton jälkeen.

Yksi valinta helppolukuisuuteen liittyen oli myöskin valittu kieli ja kieliasu. Ohjekirjassa päätettiin olla kääntämättä kaikkia termejä englannista suomeksi, sillä arvion mukaan ohjekirjassa kuvatut asiat on helpompi ymmärtää, kun siinä käytetään kirjaimellisesti samoja termejä, kuin kohteena olevassa alustassa eli Sentinelissä. Esimerkiksi jos kääntää sanan ”incident” sanaksi ”tapahtuma”, voi mennä helposti sekaisin, että mihin tapahtumiin oikein viitataan. Kun ohjekirjaa käyttää ja soveltaa niitä asioita käytäntöön, esimerkiksi tehdessä playbookia läpikäyntiohjeen mukaisesti, niin todennäköisesti silmät löytää helpommin termit ohjekirjan ja Sentinelin välillä, sekä kontekstin omaksuu välittömämmin, kun käytetään täsmälleen samoja termejä suomennosten sijaan. Luultavasti vastaavista syistä johtuen niin sanottu finglish eli suomen ja englannin kielten sekoittaminen on muutenkin yleinen tapa kommunikoida, etenkin IT-alalla. Tässä harkittiin myös puhtaasti englannin kielen käyttämistä ohjekirjan kokoamisessa, mutta arvioitiin lopulta kuitenkin suomen kielen olevan helpompi ymmärtää valtaosalle. Nähtäväksi jää, oliko tämä oikea valinta, ja tähänkin selkeämmän vastauksen voisi saada jatkotutkimuksella, jossa SOCin työntekijöitä haastateltaisiin ohjekirjan käytön jälkeen.

Suurin yksittäinen haaste koko opinnäytetyössä oli sen tekeminen kokopäivätyön ohella. On raskasta olla saman työkoneen ääressä 7 päivänä viikossa lähes aamusta iltaan usean viikon ajan, ja sitä on vaikea pysyä jatkuvasti produktiivisena, kun se pelkkä päivätyö vaatii jo paljon ajatustyötä. Opinnäytetyötä oli aiheen valinnan jälkeisten ensimmäisten kuukausien

aikana kankeaa saada kunnolla aloitettua, mutta tuona aikana kuitenkin päivätöiden puolella tuli työskenneltyä aiempaa enemmän Sentinelin parissa, joka toisaalta helpotti opinnäytetyössä käsiteltävien asioiden omaksumista. Lopulta kun vauhtiin päästiin, niin työ eteni melko nopealla tahdilla loppuun asti.

Tarkoitus oli lisätä myöskin Parent ja Child Logic Appien demonstrointi läpikäyntiohjeisiin, mutta aikarajoitteiden takia sitä ei ehditty tehdä. Mielessä myös oli, että olisi voinut testata ja demonstroida KQL-hakujen tekemistä playbookilla, esimerkiksi hälytyksen kohteena olevan käyttäjän lokeihin. Sillä tavalla voisi kehitellä playbookin tekemään ensimmäisen analyysin tapahtumasta, jonka perusteella voisi tiettyjen ehtojen täyttyessä luokitella sen esimerkiksi false positive -tapahtumaksi, ja/tai muuttaa Sentinel-tapahtuman severiteettiä sopivammaksi, sekä esimerkiksi lisätä sen tietoihin kommentin, joka kertoo KQL-hausta saadun lisäkontekstin. Tämänkaltaisilla keinoilla voitaisiin karsia SOC-analyytikon taakkaa ainakin kahdella eri tavalla: karsimalla turhien hälytysten manuaalisen käsittelyn määrää, sekä lisäämällä hälytysten rikastusta automaattisella analysoinnilla.

Toisaalta ohjekirjan päätarkoituksena on kuitenkin vain opettaa automaatioratkaisujen kehittämisen perusteet, ja edellä mainitut ideat menevät edistyneempien ratkaisujen joukkoon. Nämä ovat kuitenkin hyviä ideoita jatkokehitykselle, mitä ohjekirjaan voidaan tulevaisuudessa lisätä. Myös aiheeseen liittyvät jatkotutkimukset voisivat perehtyä nimenomaan uusien ja tässä tutkimuksessa kehitettyjä ratkaisuja edistyneempien ratkaisujen kehitykseen, joilla saataisiin suorempi vaikutus aikaiseksi.

LÄHTEET

Azure Logic Apps. 2024a. Create workflows that you can call, trigger, or nest using HTTPS endpoints in Azure Logic Apps. WWW-dokumentti. Päivitetty 13.2.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-http-endpoint?tabs=standard [viitattu 16.11.2024].

Azure Logic Apps. 2024b. Run actions based on group status by using scopes in Azure Logic Apps. WWW-dokumentti. Päivitetty 25.6.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-control-flow-run-steps-group-scopes [viitattu 10.11.2024].

Azure Logic Apps. 2024c. Schema reference guide for the Workflow Definition Language in Azure Logic Apps. WWW-dokumentti. Päivitetty 13.6.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-workflow-definition-language [viitattu 2.11.2024].

Azure Logic Apps. 2024d. What is Azure Logic Apps? WWW-dokumentti. Päivitetty 14.6.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview [viitattu 14.9.2024].

CGI. 2024a. CGI yrityksenä. WWW-dokumentti. Saatavissa: https://www.cgi.com/fi/fi/cgi-yrityksena [viitattu 14.9.2024].

CGI. 2024b. Cyber Security Operations Center. WWW-dokumentti. Saatavissa: https://www.cgi.com/fi/fi/tietoturva/SOC [viitattu 14.9.2024].

Crowley, C. 2024. SANS 2024 SOC Survey: Facing Top Challenges in Security Operations. SANS Institute. PDF-dokumentti. Saatavissa: https://sansorg.egnyte.com/dl/3R2G0PVVWW [viitattu 29.9.2024].

Elo, S., Kajula, O., Tohmola, A. & Kääriäinen, M. 2022. Laadullisen sisällönanalyysin vaiheet ja eteneminen. Hoitotiede 2022, 34 (4), 215–225. Verkkolehti. Saatavissa: https://journal.fi/hoitotiede/article/view/128987/78028 [viitattu 23.11.2024].

Hinchy, E. 2023. Voice of the SOC. Tines. WWW-dokumentti. Saatavissa: https://www.tines.com/reports/voice-of-the-soc-2023/#key-findings [viitattu 29.9.2024].

Kallinen, T. & Kinnunen, T. 2021. Teemoittelu. Teoksessa Juhila, K. (toim.) Laadullisen tutkimuksen verkkokäsikirja. Tampere: Yhteiskuntatieteellinen tietoarkisto [ylläpitäjä ja tuottaja]. WWW-dokumentti. Saatavissa: https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/ [viitattu 23.11.2024].

Kasa, C. 2024. Major Challenges facing by SOC operations. LinkedIn. WWW-dokumentti. Päivitetty 18.2.2024. Saatavissa: https://www.linkedin.com/pulse/major-challenges-facing-soc-operations-chethan-kumar-reddy-kasa-fgnff/ [viitattu 15.9.2024].

Microsoft Azure. 2024. What is Azure? WWW-dokumentti. Saatavissa: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure [viitattu 14.9.2024].

Microsoft Entra. 2024a. Planning for mandatory multifactor authentication for Azure and other admin portals. WWW-dokumentti. Päivitetty 5.11.2024. Saatavissa: https://learn.microsoft.com/en-us/entra/identity/authentication/concept-mandatory-multifactor-authentication [viitattu 9.11.2024].

Microsoft Entra. 2024b. What are managed identities for Azure resources? WWW-dokumentti. Päivitetty 23.10.2023. Saatavissa: https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview [viitattu 2.11.2024].

Microsoft Entra ID. 2024. What is Microsoft Entra ID? WWW-dokumentti. Päivitetty 25.8.2024. Saatavissa: https://learn.microsoft.com/en-us/entra/fundamentals/whatis [viitattu 14.9.2024].

Microsoft Graph. 2024a. Overview of Microsoft Graph. WWW-dokumentti. Päivitetty 16.3.2023. Saatavissa: https://learn.microsoft.com/en-us/graph/overview [viitattu 2.11.2024].

Microsoft Graph. 2024b. Overview of Microsoft Graph permissions. WWW-dokumentti. Päivitetty 31.10.2024. Saatavissa: https://learn.microsoft.com/en-us/graph/permissions-overview?tabs=http [viitattu 2.11.2024].

Microsoft Graph. 2024c. Revoke sign-in sessions. WWW-dokumentti. Päivitetty 17.4.2024. Saatavissa: https://learn.microsoft.com/en-us/graph/api/user-revokesigninsessions?view=graph-rest-1.0&tabs=http [viitattu 2.11.2024].

Microsoft Graph. 2024d. Tutorial: Grant and revoke app roles in Microsoft Entra ID. WWW-dokumentti. Päivitetty 5.3.2024. Saatavissa: https://learn.microsoft.com/en-us/powershell/microsoftgraph/tutorial-grant-app-only-api-permissions?view=graph-powershell-1.0 [viitattu 2.11.2024].

Microsoft Graph. 2024e. User resource type. WWW-dokumentti. Päivitetty 17.10.2024. Saatavissa: https://learn.microsoft.com/en-us/graph/api/resources/user?view=graph-rest-1.0 [viitattu 9.11.2024].

Microsoft Security. 2024a. What is SIEM? WWW-dokumentti. Saatavissa: https://www.microsoft.com/fi-fi/security/business/security-101/what-is-siem [viitattu 14.9.2024].

Microsoft Security. 2024b. What is SOAR? WWW-dokumentti. Saatavissa: https://www.microsoft.com/fi-fi/security/business/security-101/what-is-soar [viitattu 14.9.2024].

Microsoft Sentinel. 2024a. Authenticate playbooks to Microsoft Sentinel. WWW-dokumentti. Päivitetty 21.5.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/sentinel/automation/authenticate-playbooks-to-sentinel [viitattu 3.11.2024].

Microsoft Sentinel. 2024b. Azure Logic Apps for Microsoft Sentinel playbooks. WWW-dokumentti. Päivitetty 17.8.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/sentinel/automation/logic-apps-playbooks [viitattu 14.9.2024].

Microsoft Sentinel. 2024c. Quick threat detection with near-real-time (NRT) analytics rules in Microsoft Sentinel. WWW-dokumentti. Päivitetty 28.5.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/sentinel/near-real-time-rules [viitattu 3.11.2024].

Microsoft Sentinel. 2024d. What is Microsoft Sentinel? WWW-dokumentti. Päivitetty 21.5.2024. Saatavissa: https://learn.microsoft.com/en-us/azure/sentinel/overview?tabs=azure-portal [viitattu 14.9.2024].

Päivinen, A. 2024. Revoke user sign-in sessions from Entra ID using a Logic App in Azure Sentinel. Blogi. Päivitetty 13.1.2024. Saatavissa: https://www.anssipaivinen.fi/posts/Revoke-User-Sign-In-Sessions-by-Logic-App-Sentinel-Playbook/ [viitattu 2.11.2024].

Raina, S. 2021. How to trigger a Logic App from another Logic app. Blogi. Päivitetty 15.3.2021. Saatavissa: https://missionpossiblesneha.medium.com/how-to-trigger-a-logic-app-from-another-logic-app-5c39777ed9ac [viitattu 16.11.2024].

Tilbury, J. & Flowerday, S. 2024a. Automation Bias and Complacency in Security Operation Centers. Computers 13, 165. Verkkolehti. Saatavissa: https://doi.org/10.3390/computers13070165 [viitattu 21.9.2024].

Tilbury, J. & Flowerday, S. 2024b. Humans and Automation: Augmenting Security Operation Centers. Journal of Cybersecurity and Privacy 4, 388-409. Verkkolehti. Saatavissa: https://doi.org/10.3390/jcp4030020 [viitattu 21.9.2024].

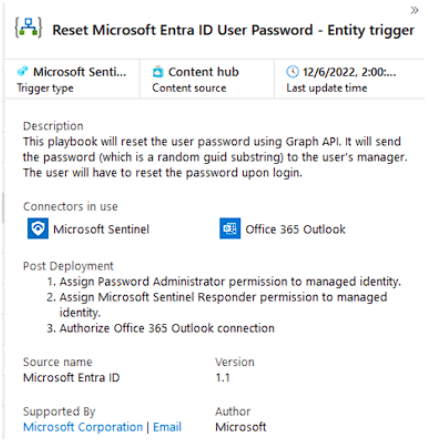
Wilson, A. 2022. Azure Logic App | Parallel Terminates & Action State Checking. Blogi. Päivitetty 4.9.2022. Saatavissa: https://andrewilson.co.uk/post/2022/09/azure-logic-app-parallel-terminate/ [viitattu 10.11.2024].

Katkelmat kenttäpäiväkirjasta

9.11.2024

Seuraavaksi aloin miettiä muita playbookilla ajettavia toimintoja, kuin istuntojen katkaisu ja kommentin lisääminen. Totesin, että salasanan resetointi on seuraava looginen askel, sen ollessa yhdessä istuntojen katkaisun kanssa yksi keskeisimmistä toiminnoista SOCille. Katsoin jälleen, miltä valmiit Sentinelin tarjoamat mallit näytti.

Valmiista pohjista löytyi salasanan resetointiin playbook päällisinpuolin samankaltaisella kaavalla, kuin aiemmin käyttöönotettu istuntojen katkaisun playbook.



Tässä kohtaa huomasin, että tuossahan oli selkeästi mainittu Post Deployment -ohjeissa, että mitkä oikeudet tulee managed identitylle antaa. Palasin katsomaan, että samalla tavalla nuo ohjeet oli playbookissa ” Revoke Entra ID Sign-in session using entity trigger”. Olin vain ne fiksusti sivuuttanut.

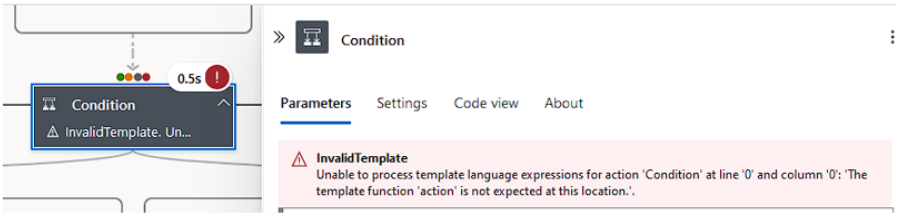
Opinpahan paremmin, kun piti perehtyä sopiviin oikeuksiin muuta kautta.

Huomiolle pantavaa myöskin on, että ”Revoke Entra ID Sign-in session using entity trigger” playbookin ohjeissa sanotaan, että olisi pitänyt antaa ”User.ReadWrite.All” ja ”Directory.ReadWrite.All” API-oikeudet, vaikka sekä Microsoftin kyseisen API-päätepisteen dokumentaatiosivulta, että käytännön testaamisen kautta selvisi, että riittävä oikeus on ”User.RevokeSessions.All”.

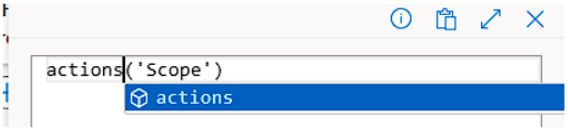
On todennäköistä, että kyseisen playbook-mallin ohje on vanhentunut, ja ”User.RevokeSessions.All”-oikeutta ei mahdollisesti edes ollut vielä olemassa tuolloin, kun kyseinen ohje kirjoitettiin, ilmeisesti vuonna 2022. Tämä mielessä pitäen, täytyy varmistaa tämän salasanaresetoinnin playbookinkin ohjeen paikkansapitävyys vielä muualta.

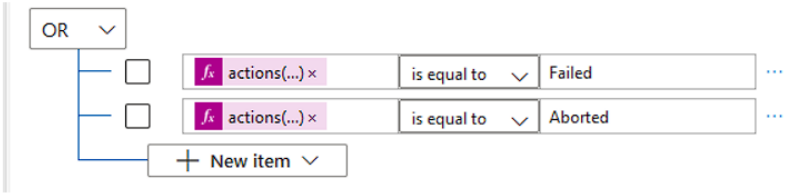
Loin kyseisen playbookin vakioasetuksilla ja menin katsomaan, miltä näyttää.

Testasin ajaa playbookia, ja menin katsomaan kumman kommentin se incidentille antoi. Suoritus epäonnistui. Virheviestin perusteella konditiossa käytetty action-funktio ei sovellu tähän käyttöön.



Epäilin tämän liittyvän siihen, ettei action-funktiota edes tunnistettu olemassa olevana funktiona playbookissa. Päätin kokeilla, että toimisiko sitten se saatavilla ollut actions-funktio.





Muokkasin molemmat kohdat muotoon ”actions(’Scope’)”, ja ajoin playbookin. Playbook suoriutui tällä kertaa onnistuneesti, ja konditiokin meni oikealle polulle eli tuloksena False.

Nyt pystyin poistamaan Parentilta eli ”Incident-RevokeSessions_ResetPassword-EntraID”-playbookilta turhat ”User.RevokeSessions.All”- sekä Password Administrator -oikeudet.

”User.RevokeSessions.All”-oikeuden poisto:

```
PS H:\> # Enter the name of the Logic App
$LogicAppName = "Incident-RevokeSessions_ResetPassword-EntraID"

# Enter the name of the permission to remove
$PermissionName = "User.RevokeSessions.All"

# Connect to Microsoft Graph
Connect-MgGraph -Scopes "AppRoleAssignment.ReadWrite.All", "Application.Read.All"

# Get Managed Identity Service Principal
$MI = Get-MgServicePrincipal -Filter "displayName eq '$LogicAppName'"

# Get Microsoft Graph Service Principal
$GraphServicePrincipal = Get-MgServicePrincipal -Filter "appId eq '00000003-0000-0000-c000-000000000000'"

# Get the app role
$AppRole = $GraphServicePrincipal.AppRoles |
    Where-Object {$_.Value -eq $PermissionName -and $_.AllowedMemberTypes -contains "Application"}

# Find the app role assignment to delete
$AppRoleAssignment = Get-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $MI.Id |
    Where-Object { $_.AppRoleId -eq $AppRole.Id -and $_.ResourceId -eq $GraphServicePrincipal.Id }

# Remove the app role assignment
Remove-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $MI.Id -AppRoleAssignmentId $AppRoleAssignment.Id
```

Tarkistus, ettei kyseistä oikeutta enää ole:

```
PS H:\> # Enter the name of the Logic App
$LogicAppName = "Incident-RevokeSessions_ResetPassword-EntraID"

# Connect to Microsoft Graph
Connect-MgGraph -Scopes "AppRoleAssignment.ReadWrite.All", "Application.Read.All"

# Get the Managed Identity Service Principal of the Logic App
$MI = Get-MgServicePrincipal -Filter "displayName eq '$LogicAppName'"

# Retrieve the list of app role assignments for the Logic App's managed identity
$Permissions = Get-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $MI.Id

# Display each assigned permission with its details
$Permissions | ForEach-Object {
    $AppRoleId = $_.AppRoleId
    $ResourceId = $_.ResourceId

    # Get the service principal for the resource (API) to which the permission belongs
    $ResourceSP = Get-MgServicePrincipal -ServicePrincipalId $ResourceId
    $AppRole = $ResourceSP.AppRoles | Where-Object { $_.Id -eq $AppRoleId }

    # Display permission information
    [PSCustomObject]@{
        "API" = $ResourceSP.DisplayName
        "Logic App" = $LogicAppName
        "Permission" = $AppRole.Value
        "Description" = $AppRole.Description
        "Resource ID" = $ResourceId
        "App Role ID" = $AppRoleId
    }
}
Welcome to Microsoft Graph!

Connected via delegated access using 14d82eec-204b-4c2f-b7e8-296a70dab67e
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs

NOTE: You can use the -NoWelcome parameter to suppress this message.


PS H:\>
```

Ei löydy myönnettyjä application permissioneita enää.

Liite 2/1

Katkelmat ohjekirjasta

Mikä playbook? Mikä Logic App?

Azuressa on alusta nimeltä Logic Apps, jota voidaan käyttää automaattisten prosessien kehittämiseen. Playbookit Sentinelissä pohjautuu Logic Appeihin; käytännössä ne siis ovat Logic Appeja, joita vain Sentinelissä kutsutaan playbookeiksi. Niillä on kaikki samat toiminnallisuudet kuin millä tahansa muulla Logic Appilla Azuressa.

Playbookeilla voi Sentinelissä luoda prosesseja, joita voi laittaa suoriutumaan joko manuaalisesti tai automaatiosäännöillä automaattisesti. Tämä ohjekirja tarjoaa ohjeet playbookien ja automaatiosääntöjen luomiselle.

Tässä ensimmäisessä osiossa perehdytään Logic Appien perusteisiin, joka toivottavasti auttaa ymmärtämään, miten playbookit toimii ja miten niiden sisältöä voidaan luoda.

Logic Appin rakenne

Logic Appin prosessit seuraavat JSON-pohjaista kieltä nimeltä Workflow Definition Language (WDL), joka määrittää sen sisältämien toimintojen järjestyksen. Logic App koostuu eri vaiheista: alussa on trigger, eli toiminto, joka käynnistää Logic Appin suorituksen. Esimerkiksi uusi hälytys Sentinelissä voi toimia triggerinä. Tätä seuraa muut toiminnot, jotka suoritetaan playbookin aikana, kuten vaikkapa jonkin vastatoimen suorittaminen ja/tai sähköpostin lähettäminen. Toimintojen kanssa voidaan hyödyntää erilaisia ohjelmoinnista tuttuja toimia, kuten ehtoja ja silmukoita. Toiminnoille voidaan määrittää parametrejä, jotka sallii myöskin dynaamisia suorituksenaikaisia syötteitä.

Microsoftin dokumentaatiosta löytyy tietoa WDL:n hyödyntämiseksi, kuten mitä funktioita ja operaattoreita WDL-lausekkeissa voidaan käyttää ja miten.

Triggerit

Playbookissa on aina oltava alussa jokin trigger, joka käynnistää sen toiminnan. Keskeisimmät triggerit ymmärtää ovat incident ja entity trigger.

Incident trigger

- Käynnistyy incidentistä.
- Kyseisen incidentin tiedot lähetetään playbookille.
- Jos incidentissä on useita entiteettejä, incident triggerin playbook voidaan määrittää toimimaan kaikkien entiteettien osalta.
- Incident triggerin playbook voidaan myös liittää automaatiosääntöön, jolloin playbook käynnistyy automaattisesti, kun tietyn tyyppinen incident luodaan tai päivitetään.

Entity trigger

- Käynnistyy entiteetistä (kuten käyttäjä, laite, IP-osoite).
- Entiteetin tiedot lähetetään playbookille.
- Jos playbook käynnistetään entiteetin kohdalta jonkin incidentin yhteydessä, niin myös kyseisen incidentin tiedot lähetetään playbookille.
- Ei voida liittää automaatiosääntöön.

Liite 2/2

Applikaatio-oikeuden lisääminen managed identitylle

Applikaatio-oikeuden lisääminen managed identitylle on sellainen toimi, mitä ei pysty Azuren portaalista tekemään. Sen voi tehdä PowerShellillä seuraavan ohjeen mukaisesti:

Step 1
Asenna Azure CLI koneellesi (ellei jo ole asennettuna) ja avaa PowerShell.

Step 2
Kirjaudu Azureen komennolla “az login”. Käytä kirjautumisessa tunnusta, jolla on vähintään Priviledged Role Administrator aktivoituna Entrassa.

Step 3
Suorita seuraava skripti, korvaten muuttujat kohdissa ”LogicAppName” ja ”PermissionName” tapauksellesi sopivilla muuttujilla:

```
---------------------------------------------------------------------------------------------------------------------------
# Enter the name of the Logic App
$LogicAppName = "RevokeSessions-EntraID"

# Enter the name of the permission
$PermissionName = "User.RevokeSessions.All"

# Connect to Microsoft Graph
Connect-MgGraph -Scopes "AppRoleAssignment.ReadWrite.All", "Application.Read.All"

# Get the managed identity service principal of the Logic App
$MI = Get-MgServicePrincipal -Filter "displayName eq '$LogicAppName'"

# Get the Microsoft Graph service principal
$GraphServicePrincipal = Get-MgServicePrincipal -Filter "appId eq '00000003-0000-0000-c000-
000000000000'"

# Get the app role
$AppRole = $GraphServicePrincipal.AppRoles |
  Where-Object {$_.Value -eq $PermissionName -and $_.AllowedMemberTypes -contains
"Application"}

# Create new app role assignment
$params = @{
  PrincipalId = $MI.Id
  ResourceId = $GraphServicePrincipal.Id
  AppRoleId = $AppRole.Id
}

New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $MI.Id -BodyParameter $params
---------------------------------------------------------------------------------------------------------------------------
```

Playbookin luominen

Step 1

- Optio 1: Tee se itse
  - Automation -> Create -> Valitse playbook joko incident, entity tai alert triggerillä *
- Optio 2: Valmis malli Sentinelistä
  - Automation -> Playbook templates (Preview) -> Valitse playbook -> Create playbook
- Optio 3: Valmis malli GitHubista
  - Mene osoitteeseen https://github.com/Azure/Azure-Sentinel/tree/master/Playbooks -> Valitse playbook -> Avaa "Deploy to Azure" -> Kirjaudu tilille, jolla on pääsy kyseiseen Sentineliin

*:
1) Alert triggeriä harvemmin käytetään, sillä lähtökohtaisesti Sentinelissä käsitellään incidentejä eikä vain yksittäisiä alerteja, ja alert triggeriä on myöskin huomattavasti rajoittavampaa käyttää automaatiosääntöjen kanssa, kuin incident triggeriä.
2) Jos aiot luoda playbookin jollain toisella triggerillä ja haluat päästä nopeasti alkuun, voit valita minkä tahansa (incident, entity, alert) triggerin ja korvata sen editoidessa playbookia. Onnistuu helpommin näin kuin valitsemalla ”Blank playbook”, joka vaatii enemmän konfigurointia alkuun.

Step 2

1. Valitse oikea subscription ja resource group
2. Jos olet tekemässä kokonaan uutta playbookia, etkä valmiista mallista: Nimeä playbook nimeämiskäytännön mukaisesti *
3. Tarkista, että connectorit sekä muut asetukset ovat ok -> Create
4. Saavut playbookin editointisivulle. Tähän näkymään pääsee myös: Sentinel workspace -> Automation -> Active playbooks -> Valitse playbook -> Edit
5. Tarkista playbookin sisältö ja/tai muokkaa sitä

*:
Nimeämiskäytäntö:
(Asiakastunniste)-Laukaisintyyppi-Päätoiminto-(Kohdealusta/laajuus)
Esimerkki: ASI-Incident-RevokeSessions_ResetPassword-EntraID

Automaatiosäännön luominen

Automation -> Create -> Automation rule

Automaatiosäännön luonti ja konfigurointi on hyvin yksinkertaista, eikä kaipaa tarkempaa selitystä. Olennainen kohta on Actions, jossa voidaan mm. valita automaatiosääntö käynnistämään playbook. Tämän ohjekirjan Läpikäyntiohjeet-osiossa käydään käytännössä automaatiosäännön luonti, jossa demonstroidaan myöskin jokainen Actions-kohdan toiminto.

Hyödyllisiä käytäntöjä ym. tietoa

Parent/Child Logic Apps

Logic Appeilla on mahdollista hyödyntää triggeriä nimeltä ”When a HTTP request is received”, joka toimii siten, että se luo itselleen ns. callback URLin, jota muut Logic Appit voivat kutsua triggeröidäkseen tämän triggerin omaavan playbookin suorituksen. Artikkelissa How to trigger a Logic App from another Logic app. | by Sneha Raina | Medium tätä prosessia kuvailtiin osuvasti Parent ja Child Logic Appien käsitteillä.

Eli Parent Logic App olisi esimerkiksi Sentinelin incidentistä startannut playbook, joka kutsuisi suorituksessaan Child Logic Appia, joka starttaisi kyseisestä kutsusta.

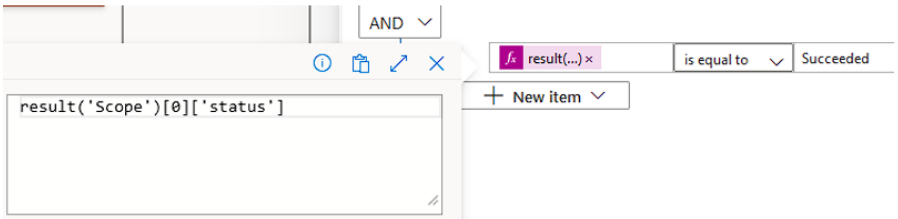
Tällaisella modulaarisella lähestymisellä voi olla monia erilaisia hyödyllisiä käyttötapauksia, mutta ainakin yksi selkeä geneerinen hyöty on siinä, että kun tekee jollekin usein tarvitulle yksittäiselle toiminnolle oman playbookinsa, ei sitä toimintoa tarvitse konfiguroida joka kerta uudestaan uusia playbookeja luodessa. Sen sijaan uudet playbookit voivat vain kutsua tätä yhtä playbookia, joka on valmiiksi konfiguroitu mm. tarvittavine oikeuksineen.

Esimerkki käyttötapauksesta:

- Playbook 1: HTTP request trigger - katkaisee istunnot API-kutsulla.
  - Tälle playbookille lisättäisiin vain ”User.RevokeSessions.All” application permission.
- Playbook 2: HTTP request trigger - pakottaa salasanan vaihdon API-kutsulla.
  - Tälle playbookille lisättäisiin vain ”Password Administrator” Entra rooli.
- Playbook 3: Incident trigger - kutsuu playbookeja 1 ja 2, sekä lisää incidentille kommentin.
  - Tälle playbookille lisättäisiin vain ”Microsoft Sentinel Responder” Azure rooli.
- Playbook 4: Entity trigger - kutsuu playbookia 1.
  - Tälle playbookille ei tarvitsisi lisätä ollenkaan oikeuksia.

Scope-toiminto virheiden ja poikkeamien käsittelylle

Scope-toiminnon sisälle voidaan laittaa yksi tai useampi toiminto, ja Scope tuottaa tuloksen sisältämiensä toimintojen suorituksista. Tuloksena voi olla ”Succeeded”, ”Failed”, ”Cancelled” tai ”Skipped”. Scopen jälkeen voi laittaa Condition-toiminnon, johon voidaan määrittää esimerkiksi, että jos edeltävän Scopen tulos on ”Succeeded”, niin tulos on ”True”, ja muutoin ”False”. Tätä metodia on kätevä hyödyntää muun muassa virheiden ja poikkeamien käsittelyssä.
Kun tätä metodia halutaan hyödyntää, voidaan Condition-toimintoon laittaa arvoksi ”result('Scope')[0]['status']” jossa ’Scope’ täytyy olla kyseisen Scope-toiminnon nimi, johon viitataan.



Huomioitavia haasteita

Microsoftin dokumentaation epäluotettavuus

Microsoftin dokumentaatio on yleisesti ottaen hyvin kattava ja hyödyllinen lähde, jota ehdottomasti kannattaa ja täytyy hyödyntää playbookien ja muun automaation kehityksessä Azuressa. Mutta sieltä ei välttämättä aina löydy tarvittavaa tietoa, ja toisinaan siellä voi olla vanhentunutta tai muusta syystä virheellistä tietoa, esimerkiksi matalimpiin vaadittaviin oikeuksiin tai johonkin Logic App-toiminnallisuuteen liittyen. Tästä syystä siihen ei pidä sokeasti luottaa, ja kannattaa aina validoida sieltä, kuten muualtakin poimitut ohjeet käytännön testauksella.

Playbookien toiminnan epäluotettavuus

Playbookit, kuten muutkin sovellukset, ovat alttiita bugeille eli selittämättömille toiminnan poikkeavuuksille. Playbookeja kehittäessä kannattaa huomioida virheiden ja poikkeamien käsittely, esimerkiksi Scope-toimintoa käyttäen, kuten on kuvattu tämän ohjekirjan kohdassa ” Scope-toiminto virheiden ja poikkeamien käsittelylle”. Aina kannattaa miettiä vähintäänkin, että playbookin virheellisestä toiminnasta tulisi johonkin näkyville ilmoitus, jotta se tiedostettaisi mahdollisimman pian ja vältyttäisi katastrofeilta. Automaatiolla voidaan saavuttaa suuria etuja manuaalisen työn vähentämisessä, mutta siihen sokea nojaaminen voi johtaa entistä suurempiin ongelmiin, jos sen poikkeava toiminto jää tiedostamatta.

Läpikäyntiohjeet

Tässä osiossa käydään käytännön tasolla vaihe vaiheelta läpi playbookin sekä automaatiosäännön luomista. Voit seurata ohjeita ja tehdä itse samalla ohjeen mukaisesti testitenantissa.

Ensimmäinen läpikäyntiohje on helppo ja yksinkertainen, jonka myötä pääset matalalla kynnyksellä luomaan ensimmäisen playbookisi.

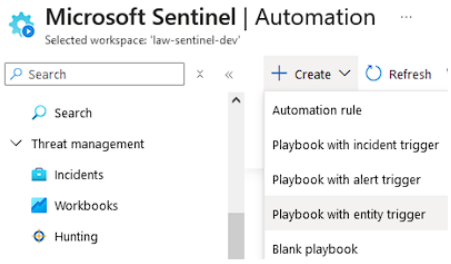
Toisessa läpikäyntiohjeessa tehdään hieman edistyneempi playbook, joka sisältää enemmän vaiheita, ja jonka kylkeen tehdään myöskin automaatiosääntö.

Easy: Revoke sessions with entity trigger

Playbook tulee sisältämään ylätasolla seuraavat toiminnot:

- Entity trigger
- Istuntojen katkaisu Graph API -kutsulla

Luodaan uusi playbook Sentinelin Automation-sivun ylälaidasta. Valitse “playbook with entity trigger”.



Valitse oikea subscription ja resource group playbookille, sekä nimeä playbook kuvaavasti.

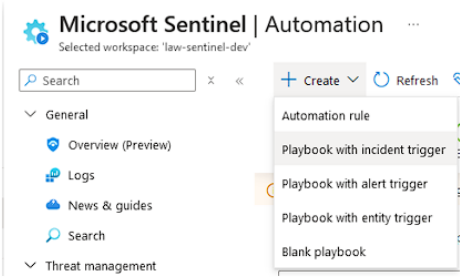
Etene viimeiselle sivulle, tarkista asetukset ja klikkaa Create playbook.

Advanced: Revoke sessions and force password reset with incident trigger, along with an automation rule

Playbook tulee sisältämään ylätasolla seuraavat toiminnot:

- Incident trigger
- Käyttäjätunnusten haku incidentiltä
- Istuntojen katkaisu Graph API -kutsulla
- Salasanan vaihdon pakotus Graph API -kutsulla
- Scope-toiminto virheiden ja poikkeamien käsittelyyn
- Kommentin lisäys Sentinelin incidentiin

Luodaan uusi playbook Sentinelin Automation-sivun ylälaidasta. Valitse playbook with incident trigger.



Valitse subscription ja resource group playbookille, sekä anna kuvaava nimi.

Etene viimeiselle sivulle, tarkista asetukset ja klikkaa Create playbook.