jamk

Azure Sentinel -palvelun SOAR-integraatio

Teemu Pätsi

Opinnäytetyö, AMK
Huhtikuu 2021
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Pätsi, Teemu

Azure Sentinel -palvelun SOAR-integraatio

Jyväskylä: Jyväskylän ammattikorkeakoulu. Huhtikuu 2021, 43 sivua.

Tietojenkäsittely ja tietoliikenne. Tieto- ja viestintätekniikka. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Verkkojulkaisulupa myönnetty: kyllä

Tiivistelmä

Toimeksiantona oli kehittää integraatio Security Orchestration, Automation and Response (SOAR) -järjestelmän ja Azure Sentinel -palvelun välille. Integraation oli tarkoitus mahdollistaa Sentinelin muodostamien tietoturvatapahtumien valvonta SOARin keskitetystä hälytysjonosta Security Operations Centerissä (SOC). Työn tavoitteena oli mahdollistaa Sentinelin tietoturvapoikkeamien käsittely SOAR-järjestelmän kautta.

Opinnäytetyö toteutettiin liitosohjelmiston kehitystyönä. Työssä kehitettiin SOARiin liitosohjelmisto, joka hakee tietoa Sentinelin tietoturvapoikkeamista Azure Sentinel REST API -ohjelmointirajapintaa hyödyntäen. Opinnäytetyön tietoperustassa esiteltiin kehitysongelma ja integroitavat palvelut sekä ohjelmiston kehityksessä käytettävä ohjelmointirajapinta. Toteutusvaiheessa esiteltiin liitosohjelmiston vaatimusmäärittely ja dokumentoitiin sen kehittämisprosessi sekä toteutetut testit.

Kehitystyön tuloksena oli tuotantoon käyttöönotettu liitosohjelmisto SOARissa. Liitosohjelmistolla Sentineliin muodostuvat tietoturvapoikkeamat saadaan automaattisesti SOARin hälytysjonoon. Integraation ansiosta SOCin analyytikot voivat seurata Sentinelin tietoturvapoikkeamia keskitetystä SOARin hälytysjonosta, johon muodostuvat eri palveluiden tietoturvahälytykset. Liitos myös mahdollistaa Sentinelin tietoturvapoikkeaman tilan muuttamisen ja kommentoimisen SOARin kautta.

Integraation ansiosta Sentinel-palvelun tietoturvapoikkeamia voi käsitellä saman käyttöliittymän kautta, josta muidenkin palveluiden muodostamia hälytyksiä käsitellään. Liitosohjelmiston avulla analyytikoiden ei tarvitse seurata kahden eri palvelun hälytysnäyttöjä. Opinnäytetyön takia tiimissä on kokemusta SOARin liitosohjelmiston kehittämisestä, joten tulevia palveluliitoksia varten ei tarvitse tehdä selvitystyötä liitosohjelmistojen toimintaperiaatteista.

Avainsanat (asiasanat)

SOC, SOAR, API, Azure Sentinel, integraatio, automatisointi

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Pätsi, Teemu

SOAR integration for Azure Sentinel

Jyväskylä: JAMK University of Applied Sciences, April 2021, 43 pages.

Information and Communications. Degree program in Information and Communication Technology. Bachelor's thesis.

Permission for web publication: Yes

Language of publication: Finnish

Abstract

The assignment was to develop the integration between the Security Orchestration, Automation and Response (SOAR) system and the Azure Sentinel service. The purpose of the integration was to enable the monitoring of security incidents generated by Sentinel from SOAR's centralized alert queue in the Security Operations Center (SOC). The objective of the thesis was to enable the handling of Sentinel's security incidents through the SOAR system.

The thesis was carried out as a software development work. A connector software was developed for SOAR, which retrieves incident information from Sentinel using the Azure Sentinel REST API. The theory section of the thesis introduced the development problem, the services to be integrated and the application programming interface used in the connector software. In the implementation section, the software's specification of requirements and its development process was presented, and the implemented tests were documented.

The result of the development work was the connector software at SOAR which was taken into production. Sentinel's security incidents are automatically fetched by the connector software which creates alerts from them into SOAR's alert queue. Thanks to the integration, SOC analysts can monitor Sentinel's security incidents from a centralized alert queue in SOAR where all the alerts from different sources are shown. The connector software also allows changing the status of Sentinel's incident and commenting it through SOAR.

The integration enables handling Sentinel's security incidents through the same graphical user interface from where the alerts generated by other sources are handled. This simplifies the work of SOC analysts as they do not have to monitor alarm queues of two different services. Due to the development process of the software, Security Operations Center now has experience in the development of connector software for SOAR. Therefore, future integrations will be easier to develop because the team already has experience with the operational principles of connector software for SOAR.

Keywords/tags (subjects)

SOC, SOAR, API, Azure Sentinel, integration, automation

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Lyhenteet

AD	Active Directory
API	Application Programming Interface
CEF	Common Event Format
CSIRT	Cyber Security Incident Response Team
CSOC	Cyber Security Operations Center
HTTPS	Hypertext Transfer Protocol Secure
HMAC	Hashed Message Authentication Code
IOC	Indicator of Compromise
IP	Internet Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
SIEM	Security Information and Event Management
SOAR	Security Orchestration, Automation and Response
SOC	Security Operations Center
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator
XOR	Exclusive OR
XSS	Cross-Site Scripting

1 Johdanto ja tavoitteet

1.1 Toimeksiantaja

Viria Security Oy on Viria-konserniin kuuluva suomalainen turvallisuusalan yritys ja Suomen suurimpia yritysturvallisuuspalveluiden toimittajia. Viria Security tuottaa sekä fyysisen, että digitaalisen turvallisuuden palveluita. Turvaliiketoiminnan kantava ajatus on ”yksi turvallisuus”, joka ilmenee Viria Securityn kykynä turvata ihmisiä, tietoa ja omaisuutta kokonaisuutena. Vuonna 2020 Viria-konsernin henkilöstön lukumäärä oli 714 ja liikevaihto 106,8 M€. Konserni tarjoaa turvaliiketoiminnan palveluiden lisäksi digitaalisen kehittämisen palveluita kuten data-analytiikan ja tiedolla johtamisen palveluita. Viria-konserniin kuuluvat emoyhtiö Viria Oyj:n lisäksi tytäryhtiöt Aureolis Oy, Bellurum BI-palvelut Oy, Bitfactor Oy, Oy Hibox Systems Ab, Tansec Oy, Spellpoint Group Oy, Viria Kiinteistöt Oy ja Viria Security Oy. (Viria Oyj Tilinpäätöstiedote 2021.)

1.2 Toimeksianto

Toimeksiantona oli kehittää Cyber Security Operations Centerille (CSOC) integraatio toimeksiantajan SOAR-järjestelmän (Security Orchestration, Automation and Response) ja Microsoftin Azure Sentinel (jatkossa Sentinel) -palvelun välille. Työn tavoitteena oli mahdollistaa Sentinelin muodostamien tietoturvapoikkeamien (engl. incident) käsittelyn SOAR-järjestelmän kautta. Lisäksi tavoitteena oli kehittää edellytykset poikkeamien tutkimisen automatisoinnille SOAR-palvelun pelikirjojen (engl. playbook) avulla.

Toimeksianto toteutetaan SOARin liitosohjelmiston kehitystyönä. Valmis integraatio mahdollistaa sen, että CSOC-tiimin asiantuntijat voivat käsitellä Sentinel-palvelun tietoturvapoikkeamia saman käyttöliittymän kautta, josta muidenkin palveluiden muodostamat tietoturvapoikkeamat ja -tapahtumat käsitellään keskitetysti. Integraation tavoitteena on myös mahdollistaa tietoturvapoikkeamien analysoimisen SOAR-palvelun kautta.

1.3 Kehitysongelma

Toimeksiantajan asiakkailla on käytössä palveluita eri palveluntarjoajilta. SOCin tavoite on, että asiakkaiden järjestelmien tietoturvatapahtumia saadaan valvottua mahdollisimman tehokkaasti palveluntarjoajasta riippumatta. Sentinel sisältää erinomaisen tuen ja säännöstön varsinkin Microsoftin tuotteiden valvomiseen. Näin ollen Sentinel on suosittu ratkaisu lokien ja tapahtumien hallintaan, mikäli käytössä on useita Microsoftin järjestelmiä. Käytössä oleva SOAR-palvelu ei sisällä tukea Sentinelin muodostamien tietoturvapoikkeamien käsittelyyn. Kehitettävän liitosohjelman avulla SOARiin tulee saada tietoa uusista hälytyksistä Sentinelissä, jotta analyytikot osaavat tutkia niitä ja reagoida niihin. Sen pitäisi myös mahdollistaa Sentinelin luomien tietoturvapoikkeamien käsittelyn SOARin kautta.

2 Integroitavat palvelut

2.1 Azure Sentinel

Sentinel on Microsoftin ylläpitämässä Azuren pilvessä toimiva SIEM- (Security Information and Event Management) ja SOAR-palvelu. Siellä voi säilyttää tietoa useista eri lähteistä ja se sisältää sisäänrakennettuna liitosohjelmia suureen osaan Microsoftin moderneista tuotteista, kuten Microsoft 365 Defender, Office 365 ja Azure AD. Sen lisäksi palvelu sisältää liitosohjelmistoja myös muiden palveluntarjoajien järjestelmiin. Sentinel osaa myös parsia ja tulkita Syslog- ja CEF (Common Event Format) lokimuotoja, joten mikäli lokilähteelle ei löydy valmista liitosohjelmistoa se voidaan kehittää luomalla Sentineliin lokimuotoa parsiva ohjelma. (What is Azure Sentinel? 2020.)

2.2 Security Orchestration, Automation and Response

2.2.1 Yleistä

SOAR-järjestelmä toimii CSOC-tiimissä keskitettynä tietoturvatapahtumien käsittelyalustana. Virian CSOC valvoo asiakkaiden tietoverkkoja ja -järjestelmiä kyberuhkien ja poikkeamien varalle. SOARin pitää pystyä tulkitsemaan tietoa hyvin erilaisilta lähdejärjestelmiltä ja -laitteilta, jotta erilaisten ympäristöjen kyberturvallisuudesta saadaan rakennettua tilannekuvaa. Valvottavien järjestelmien tapahtumat ja hälytykset tuodaan liitosohjelmistoja ja pelikirjoja hyväksikäyttäen SOARiin, joka tekee tapahtumista hälytyksiä SOCin (Security Operations Center) analyytikoiden tutkittavaksi. Hälytysten analysoiminen on nopeampaa yhden alustan kautta, sillä eri palveluiden tapahtumien oleellinen tieto löytyy SOAR-palvelusta yhtenevässä muodossa. Erilaisten palveluiden liittäminen SOC-palveluun helpottuu, kun analyytikkojen ei tarvitse opetella jokaisen uuden palvelun käyttöliittymää tai seurata useita erilaisia hälytysnäyttöjä.

2.2.2 Automaatio

Automatisoimisella tarkoitetaan SOARissa erilaisten tehtävien suorittamista ilman ihmisohjausta. Automatisoimisen tavoitteena on toteuttaa toistuvat, yksitoikkoiset tai aikaa vievät tehtävät koneen avulla (Jelen 2020). SOAR rikastaa automaattisesti hälytyksiin liittyviä tietoja ja uhkaindikaatioita (engl. Indicator of Compromise, IOC), jotta analyytikon on mahdollisimman helppoa ja nopeaa tehdä tulkinta tapauksen vakavuudesta ja tarvittavista toimenpiteistä. SOC käsittelee päivittäin valtavan määrän tietoturvahälytyksiä useiden asiakkaiden erilaisista järjestelmistä ja lokilähteistä,

joten analyytikoiden olisi mahdotonta tutkia manuaalisesti kaikki muodostuneet havainnot. Automatisoimalla hälytysten esikäsittelyä saadaan vähennettyä analyytikoiden nähtäväksi muodostuvien hälytysten määrää. Tämä ehkäisee hälytysväsymystä (engl. alert fatigue), koska hälytysjonoon tulee suhteessa vähemmän hälytyksiä, jotka analyytikon tulkinnan perusteella ilmenevät vääriksi hälytyksiksi. Aiheellisten hälytyksien vakavuutta voi olla vaikea huomata suuren hälytysmassan joukosta, mikäli analyytikko joutuu käsittelemään suuria määriä vääriä hälytyksiä (Townsend 2018).

2.2.3 Orkestrointi

Tietoturvatapahtumien orkestrointi tarkoittaa eri palveluiden ja työkalujen yhdistämistä niin, että eri komponenttien tehtävien automatisoiminen mahdollistuu. Ihminen on myös osa orkestrointia, kun tietoturvatapahtuman haitallisuutta tai kiireellisyyttä ei saada selvitettyä pelkästään koneen avulla. Orkestroimalla eri palveluiden toimia voidaan automatisoida esimerkiksi tietoturvatapahtumien uhkaindikaatioiden hakemisen, IP-osoitteiden maineiden tarkastamisen ja haittaohjelma-analyysin. (Imam 2019.)

SOAR-palveluun muodostettuihin hälytyksiin rikastetaan orkestroimalla analyytikkoa kiinnostavaa tietoa eri lähteistä. Analyytikko käyttää rikastettua tietoa hyväkseen tehdessään päätöstä sisältääkö tapahtuma oikeaa uhkaa ja minkälaisia toimia tapaus vaatii SOCilta tai asiakkaalta. Usein eri palveluiden ja järjestelmien orkestrointi nopeuttaa myös vastatoimien toteuttamista havaitun uhan tai aktiivisen kyberhyökkäyksen aikana.

2.2.4 Tietoturvapoikkeamiin vastaaminen

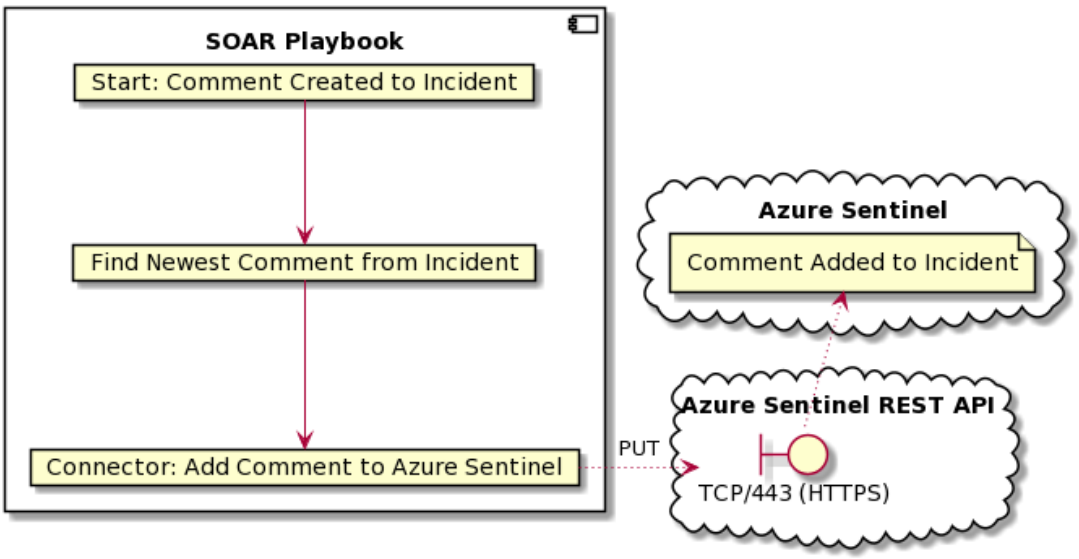
Imamin (2019) mukaan SOAR-järjestelmän poikkeamien vastatoimia edistävät tekniikat helpottavat analyytikon työtä muun muassa antamalla työkaluja poikkeamien hallintaan sekä keräämällä uhkatietoa hälytyksiin liittyen. SOAR-järjestelmässä on erilaisia moduuleja ja teknologioita, jotka helpottavat koordinoimaan tietoturvapoikkeamien käsittelyä ja mahdollistavat vastatoimenpiteitä. Moduulit ovat tietueiden kokoelmia, joilla on tiettyjä ominaisuuksia. Esimerkiksi hälytysmoduuli sisältää eri lähteistä muodostetut tietoturvahälytykset, jotka jakavat samat muokattavat parametrit. Eri tietueiden yhteneväisyys mahdollistaa yleispätevien automatisointien kehittämisen, joilla saadaan ratkaistua tietty ongelma tietueen lähdejärjestelmästä riippumatta.

Tiketöintimoduuli on tietoturvapoikkeamien hallinnan kannalta oleellinen. Mikäli analyytikko tulkitsee hälytyksen tarvitsevan toimenpiteitä, siitä tehdään tiketti (engl. incident ticket). Tikettiin kirjataan hälytykselle tehdyt toimenpiteet ja tutkimustulokset. Tiketin kautta saa ajettua kyseisen hälytyksen kannalta oleellisia pelikirjoja, jotka esimerkiksi rikastavat lisää tietoa hälytykseen eri lähteistä tai tekevät vastatoimenpiteitä hälytyksen kohteena olevalle laitteelle.

2.2.5 Pelikirjat

SOAR-järjestelmän automaattiset toimenpiteet ja eri palveluiden looginen orkestrointi perustuvat pelikirjoihin. Pelikirjat ovat visuaalisesti esitettyjä ohjelmia, jotka koostuvat toisiinsa yhdistetyistä askelista (engl. step). Askeleet voivat olla esimerkiksi eri palveluiden liitosohjelmistojen pelikirjatoimintoja tai SOAR-järjestelmän sisäänrakennettuja toimintoja. Sisäänrakennetuilla toiminnoilla voi esimerkiksi luoda ja muokata SOAR-järjestelmän hälytyksiä, kun taas liitosohjelmistojen pelikirjatoiminnot hakevat tai muokkaavat kohdejärjestelmän tietoa. Jokainen askel voi käsitellä pelikirjan aikana määriteltyjä muuttujia sekä edellisten askeleiden käsittelemää tietoa.

Integraatiota varten kehitetään liitosohjelmisto, joka mahdollistaa erilaisien toimenpiteiden toteuttamisen Sentinelin tietoturvapoikkeamille SOARin kautta. Kehitettävässä liitosohjelmistossa tulee siis olla erilaisia pelikirjatoimintoja, joita toteutetaan askelina pelikirjoissa. Kuviossa 1 on kuvattu esimerkki pelikirjasta, joka käyttää kehitettävän liitosohjelmiston pelikirjatoimintoa.



Kuvio 1. Pelikirja lisää SOARin kommentin Sentinelin tietoturvapoikkeamaan

2.3 Azure Sentinel REST API

Liitosohjelmiston toteuttamisessa päätettiin käyttää hyväksi Azure Sentinel REST API -ohjelmointirajapintaa. Tällä ohjelmointirajapinnalla voi hakea tietoa Sentinelin resursseista ja muokata niitä (Azure Sentinel 2020). Sen tärkeimmät ominaisuudet liitosohjelmiston kannalta on tietoturvapoikkeamien hakemisen helppous ja niiden muokkausmahdollisuudet. Näillä toiminnoilla saadaan jo toteutettua suuri osa vaatimusmäärittelyssä kappaleessa 3.3 esille tulevista ominaisuuksista, mukaan lukien pakolliset vaatimukset liitteen 2 toiminnallisista vaatimuksista.

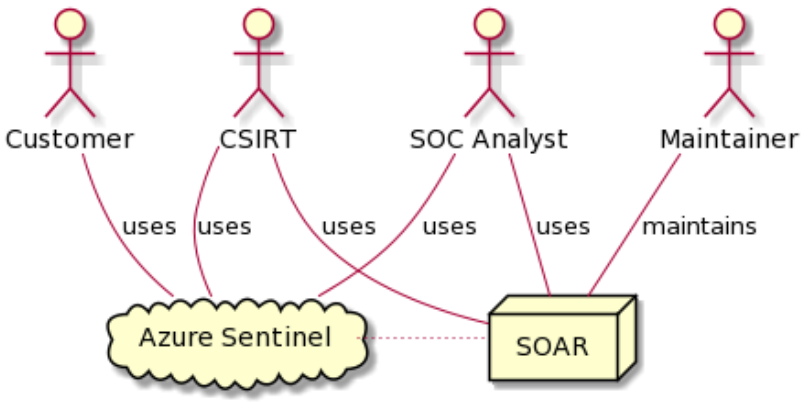
Azure Sentinel REST API toteuttaa nimensä mukaisesti REST-arkkitehtuuria (Representational State Transfer). REST-arkkitehtuuria toteuttavalle ohjelmointirajapinnan resursseille voi osoittaa komentoja lähettämällä HTTP-kyselyitä. REST-ohjelmointirajapinnan tapauksessa resurssi tarkoittaa samankaltaisten tietueiden joukkoa. Eri HTTP-metodeilla saa toteutettua eri toiminnallisuuksia. Esimerkiksi GET-metodilla saa yleensä haettua tietoja tiettyyn resurssiin liittyen. (Fielding 2000.)

REST-ohjelmointirajapinnat ovat tilattomia (engl. stateless). Tilattomuus tarkoittaa sitä, että palvelin ei muista asiakasohjelmien (engl. clients) lähettämiä kyselyitä. Jokainen ohjelmointirajapinnalle lähetettävä kysely tulee sisältää kaikki pyynnön suorittamiseksi tarvittava informaatio, mukaan lukien asiakasohjelman tunnistautumiseen ja autentikaatioon tarvittavat tiedot. (Fielding 2000.)

3 Vaatimusmäärittely

3.1 Käyttäjäryhmät

Toimeksiantajan SOC-analyytikot valvovat Sentinelin tietoturvapoikkeamia SOARin kautta ja käyttävät Sentineliä hyväkseen tapahtumien tutkimisessa ja vastatoimenpiteissä. CSIRT-tiimi (Cyber Security Incident Response Team) toimii SOCin apuna aktiivisessa kyberhyökkäystilanteessa tai kyberpoikkeaman forensiikkatutkimuksessa. Asiakas voi käyttää ja tarvittaessa ylläpitää omaa osiotaan Sentinelistä. SOARin ylläpitäjät ovat vastuussa siitä, että SOAR-järjestelmä ja sen integraatiot toimivat. Edellä mainittujen käyttäjäryhmien suhteet integroitaviin järjestelmiin havainnollistetaan kuviossa 2.



Kuvio 2. Integraation käyttäjäryhmät

3.2 Käyttäjätarinat

Käyttäjätarinat (engl. user story) ovat varsinkin ketterissä ohjelmistokehitysmenetelmissä käytetty menetelmä, jolla ohjelmistokehittäjä pyrkii sisäistämään asiakkaan tai palvelun loppukäyttäjien tarpeita (Lucassen, Dalpiaz, Van der Werf & Brinkkemper 2016). Käyttäjätarinat ovat nimestään huolimatta yksittäisiä lauseita, jotka kertovat loppukäyttäjän näkökulmasta ilmaistuna kyseiselle käyttäjälle tärkeästä ohjelman toiminnallisuudesta. Cohnin (2004) mukaan käyttäjätarinoiden olisi hyvä perustua jonkun tietyn käyttäjäryhmän edustajan rooliin, joka tarvitsee kyseistä ominaisuutta eniten. Siitä pitää käydä ilmi käyttäjän arvostama toiminnallisuus ja mahdollisesti myös syy toiminnallisuuden tarpeelle. Niiden olisi myös hyvä toteuttaa samankaltaista pohjaa, jotta oleellinen tieto toiminnallisuudesta ja sen merkityksestä käy helposti ilmi. (Cohn 2004.) Connextran pohja on yleisimmin käytössä oleva pohja, jossa korostuu käyttäjän rooli osana käyttäjätarinaa (Lucassen ym. 2016). Taulukossa 1 on esitelty SOAR-palvelun kanssa toimivien henkilöiden toiveisiin perustuvat käyttäjätarinat Connextran pohjaan muotoiltuna.

Taulukko 1. Käyttäjätarinat

ID	Käyttäjätarina
US001	SOC-analyytikkona haluan käsitellä Sentinelin tietoturvapoikkeamia samasta paikasta kuin muidenkin järjestelmien hälytyksiä
US002	SOC-analyytikkona haluan nähdä Sentinelin tietoturvapoikkeaman tapahtumien raakalokin SOARin kautta, jotta sitä ei tarvitsisi käydä katsomassa Sentinelistä
US003	SOC-analyytikkona haluan nähdä Sentinelin tietoturvapoikkeaman tapahtumista oleelliset tiedot samalla tavalla kuin muidenkin SIEM-palveluiden hälytyksistä
US004	SOC-analyytikkona haluan ajaa Sentinelin pelikirjoja SOARin kautta
US005	SOC-analyytikkona haluan päästä Sentinelin poikkeamaan hälytyksen linkistä
US006	SOAR-palvelun ylläpitäjänä haluan tehdä Azure Sentinelin ohjelmointirajapinnan mahdollistamia kyselyitä Sentineliin kehittäessäni SOARin pelikirjoja

3.3 Tärkeimmät toiminnallisuudet

Taulukossa 2 on esitelty liitosohjelmiston tärkeimmät toiminnallisuudet ja niille määritellyt prioriteetit. Ensimmäisen prioriteetin (P1) toiminnallisuudet ovat ohjelmistolle pakollisia, toiset ovat hyödyllisiä ja kolmannet toivottuja. Mikäli jotkut toiminnallisuuksista jäisivät toteuttamatta ohjelmiston kehittämisvaiheessa, niiden kehittämistä jatkettaisiin tuotantovaiheessa mikäli käyttäjät kokevat ominaisuuden tarpeelliseksi.

Taulukko 2. Tärkeimmät toiminnallisuudet

ID	Prioriteetti	Toiminnallisuus
FT001	P1	Uusien Sentinelin tietoturvapoikkeamien muodostaminen SOARiin hälytyksiksi automaattisesti
FT002	P2	Sentinelin tietoturvapoikkeamien kommentoiminen SOARin kautta
FT003	P2	Sentinelin tietoturvapoikkeamien tilan muutos SOARin kautta
FT004	P2	Sentinelin tietoturvapoikkeamaan liittyvien hälytysten tietojen rikastaminen SOARin hälytykseen
FT005	P3	Mielivaltaisten API-kyselyiden tekeminen Azure Sentinelin ohjelmointirajapintaan SOARin pelikirjan kautta
FT006	P3	Sentinelin tietoturvapoikkeamien käsittelyn automatisoimiseen liittyvien pelikirjatoiminnallisuuksien mahdollistaminen
FT007	P3	Sentinelin pelikirjojen ajaminen SOARin kautta

3.4 Tuotannolliset ja tekniset vaatimukset

Toiminnalliset vaatimukset

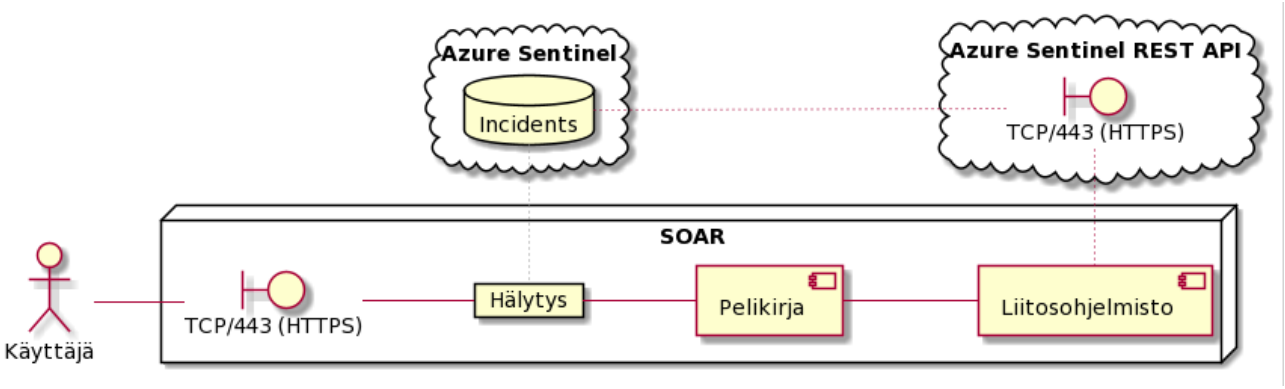
Ohjelmiston toiminnalliset vaatimukset määrittelevät toiminnallisuudet, jotka ohjelmistoon on kehitettävä käyttäjien tarpeiden täyttämiseksi. Liitteessä 2 esitellyt toiminnalliset vaatimukset on priorisoitu JUHTAn suosituksen JHS 173 (2018) mukaisesti kolmetasoisesti. Ensimmäisen prioriteetin vaatimukset ovat ohjelmiston käyttöönoton kannalta pakollisia, toiset ovat hyödyllisiä ja kolmannet toivottuja.

Ei-toiminnalliset vaatimukset

Ei-toiminnalliset vaatimukset määrittelevät ohjelmiston toiminnallisuuksille reunaehtoja ja rajoituksia. Näillä varmistetaan, että ohjelmiston käytettävyys on riittävän laadukasta ja että ohjelmisto on luotettava ja tietoturvallinen. (JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely 2018.) Integraation liitosohjelmiston ei-toiminnalliset vaatimukset on esitelty liitteessä 3.

3.5 Sijoittelunäkymä

Kaikki yhteydet SOARista Azure Sentinelin ohjelmointirajapintaan toteutetaan SOARin pelikirjojen avulla. SOARissa tulee olla pelikirja, joka lyhyen ajan välein tarkastaa liitosohjelmaa käyttäen APIsta onko Sentineliin muodostunut uusia tietoturvapoikkeamia. Jokaisesta uudesta poikkeamasta luodaan hälytys SOARiin, joka näkyy SOC-analyytikoille avatussa (engl. open) tilassa. Osa kehitettävistä pelikirjoista on ajettavissa suoraan SOARiin muodostuneesta hälytyksestä. Käyttäjä voisi hälytyksen kautta esimerkiksi ajaa pelikirjoja, joilla saa haettua Sentinelin ohjelmointirajapinnasta lisää tietoa poikkeamaan liittyen. Ohjelmiston sijoittelunäkymä on kuvattu kuviossa 3.



Kuvio 3. Liitosohjelmiston sijoittelunäkymä

3.6 Laadunvarmistus

Jokainen toteutettu pelikirjatoiminnallisuus on testattava, jotta varmistutaan liitteissä 2 ja 3 määriteltyjen liitosohjelmiston vaatimusten täyttymisestä. Testit on suunniteltu niin, että jokainen ohjelmistolle määritelty vaatimus tulee varmistetuksi, mikäli siihen liittyvä toiminnallisuus on toteutettu. Liitosohjelmistolle toteutetut testit ja niiden tulokset esitellään kappaleessa 5.

4 Palveluintegraation kehittäminen

4.1 Liitosohjelmiston alustaminen

SOAR-järjestelmän liitosohjelmistot käyttävät Python3-ohjelmointikieltä. Kehitettävän ohjelmiston pohjana käytettiin SOAR-palvelusta löytyvää Microsoft Graph API -liitosohjelmistoa, joka mahdollistaa esimerkiksi Azure Security Centerin hälytysten hakemisen ja erilaisten uhkaindikaattoreiden etsimisen lokeista. Kyseinen ohjelmisto oli toteutettu hyvin eri tavalla kuin miten kehitettävä liitosohjelmisto on suunniteltu, joten siitä ei saatu suoraan käytettyä mitään osaa koodista. Kuitenkin sen avulla saatiin helposti muokattua kuvion 4 tiedostoja niin, että niissä on kaikki tarvittavat sisällytykset (engl. imports) ja konfiguraatiot. Sain valmiista liitosohjelmistosta myös esimerkkejä, kuinka mahdollisten ajon aikaisten virheiden lokitus onnistuu käyttäen liitosohjelmistoille yhteisestä kirjastosta sisällytettyä *ConnectorError*-poikkeusta.

connectorname folder
--+ playbooks
---+ __init__.py
---+ playbooks.json
--+ connector.py
--+ info.json
--+ images
--+ requirements.txt
--+ packages
---+ <package_name>

Kuvio 4. Liitosohjelmiston kansiorakenne (SOAR-järjestelmän dokumentaatio 2020)

4.2 Kehitettävien pelikirjatoiminnallisuuksien rajaaminen

Azure Sentinel REST APIn avulla ei saa haettua Sentinelin poikkeamaan liittyvien hälytysten tietoja, sillä sen uusin *2020-01-01* -versio ei sisällä siihen liittyvää toiminnallisuutta (Azure Sentinel 2020). Nimimerkin azsec (2020) mukaan poikkeamaan liittyvien relaatioiden tiedot saisi haettua GET-pyynnöllä ohjelmointirajapinnan resurssiin */incidents/{incidentId}/relations*, mutta tämä toiminnallisuus on ilmeisesti otettu uudesta API-versiosta pois. Niesenin (2020) mukaan version *2019-01-01-preview* avulla saisi edelleen haettua poikkeaman relaatiot ja niistä liittyvien hälytysten tiedot käyttäen kuvion 5 API-kyselyä. Tämän kyselyn HTTP-leipätekstin (engl. message body) *expansionId*

parametri on kovakoodattu arvo, joka tarkoittaa ”kaikkia relaatioita”. Päätimme olla toteuttamatta liittyvien hälytysten hakuominaisuutta tämän ohjelmointirajapinnan avulla, koska kyseessä on näin erikoisesti toteutettu ja heikosti dokumentoitu vanhan APIn resurssi.

```
POST
https://management.azure.com/subscriptions/##SUBSCRIPTIONID##/resourceGroups/##RESOURCEGROUP##/providers/Microsoft.OperationalInsights/workspaces/{2}/providers/Microsoft.SecurityInsights/entities/##RelatedResourceName##/expand?api-version=2019-01-01-preview

The following HTTP-body should be used:
{
   "expansionId": "98b974fd-cc64-48b8-9bd0-3a209f5b944b",
}
```

Kuvio 5. Vanhan API-version kyselyllä voi hakea poikkeamaan liittyvien hälytysten tiedot (Niesen 2020)

Microsoft ylläpitää Azure Sentinel REST APIn lisäksi myös Microsoft Graph APIa, jolla voi hakea tietoa Azuren palveluista ja tehdä niille toimenpiteitä (Use the Microsoft Graph Security API 2021). Tehdyn selvityksen perusteella tämän ohjelmointirajapinnan käyttäminen ei olisi järkevää Sentinelin liitosohjelmistossa, koska Graph APIn kautta ei voi hakea tietoa Sentinelin tietoturvapoikkeamista. Vaikka Graph APIsta saa hyvin rikastettua tietoa eri Azuren palveluiden hälytyksiin liittyen, tätä tietoa ei saada käytettyä hyväksi Azure Sentinel REST APIn kanssa, koska ne käyttävät eri ID:tä samoista hälytyksistä. Tämän vuoksi varmaa korrelaatiota kuvion 5 API-kyselyn palauttamien ja Graph APIn palauttamien hälytysten välillä ei saada tehtyä.

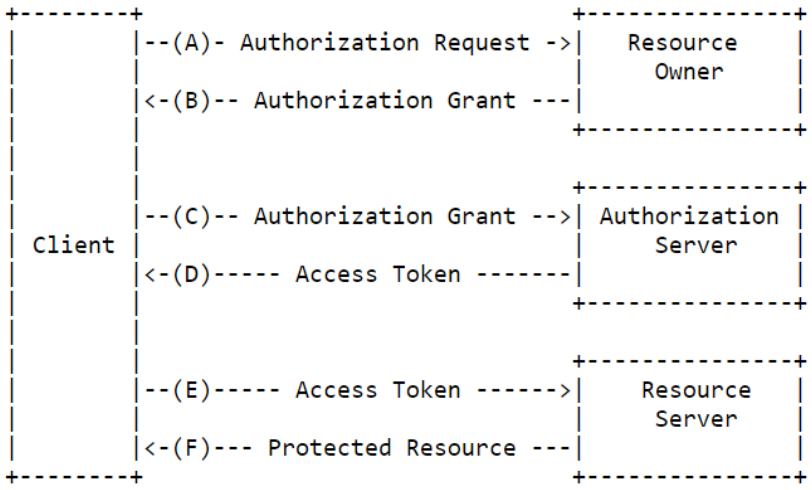
Kehitettävän liitosohjelmiston ensimmäiseen versioon ei toteuteta ominaisuutta FT004, koska käytettävillä ohjelmointirajapinnoilla ei saada tuotantokelpoisesti haettua tietoa Sentinelin tietoturvapoikkeamiin liittyvistä hälytyksistä. Myöskään ominaisuutta FT006 ei saada toteutettua liitosohjelmistoon, sillä SOARin hälytysten automatisoiminen vaatisi liittyvistä hälytyksistä rikastettua dataa. Kehitystyön aikana myös todettiin, että Sentinelin tapahtumien automatisoiminen on järkevämpää toteuttaa Sentinelin puolella käyttäen hyväksi sen olemassa olevia pelikirjoja mahdollisuuksien mukaan. Ominaisuutta FT007 ei voida toteuttaa, koska kumpikaan Azuren ohjelmointirajapinta ei mahdollista Sentinelin pelikirjojen ajamista ohjelmallisesti. (ks. taulukko 2.)

4.3 Tarvittavien API-kyselyiden selvittäminen

Liitosohjelmistossa tarvittavien API-kyselyiden selvittäminen toteutettiin Postman-ohjelmalla. Postman on ilmainen API-kyselyiden testaustyökalu, joka mahdollistaa useiden kyselyiden kokoamisen ja kyselyiden parametrien asettamisen muuttujiksi. Tämän ansiosta kyselyistä saa otettua kuvankaappauksia ilman, että salassa pidettävät parametrit olisivat vaarassa vuotaa. Tällöin ei myöskään haittaa, vaikka joku näkisi ruudun selän takaa. API-kyselyiden kokoelmat mahdollistavat kaikkien kokoelman kyselyiden testaamista yhtäaikaisesti ja edellisten kyselyiden vastauksia voi halutessaan käyttää seuraavien kyselyiden muuttujissa. (Postman Tutorial: How to use Postman Tool for API Testing 2020.)

4.4 Autentikaatio

Azure Sentinel REST APIn kyselyiden autentikaatiossa käytetään **bearer tokeneita** (Gallant 2017). Tokenin hallussapitäjä voi käyttää sitä tunnistautuakseen tiettyyn suojattuun resurssiin ilman erillistä kirjautumistietojen antamista. Bearer tokenin saa haettua kuvion 6 mukaisesti valtuutuspalvelimelta (engl. authorization server) identiteetin todentamista vastaan, mikäli kyseisellä identiteetillä on oikeudet haettuun resurssiin.



Kuvio 6. Asiakasohjelman vuorovaikutus resurssin omistajan, valtuutuspalvelimen ja resurssipalvelimen kanssa (Jones & Hardt 2012)

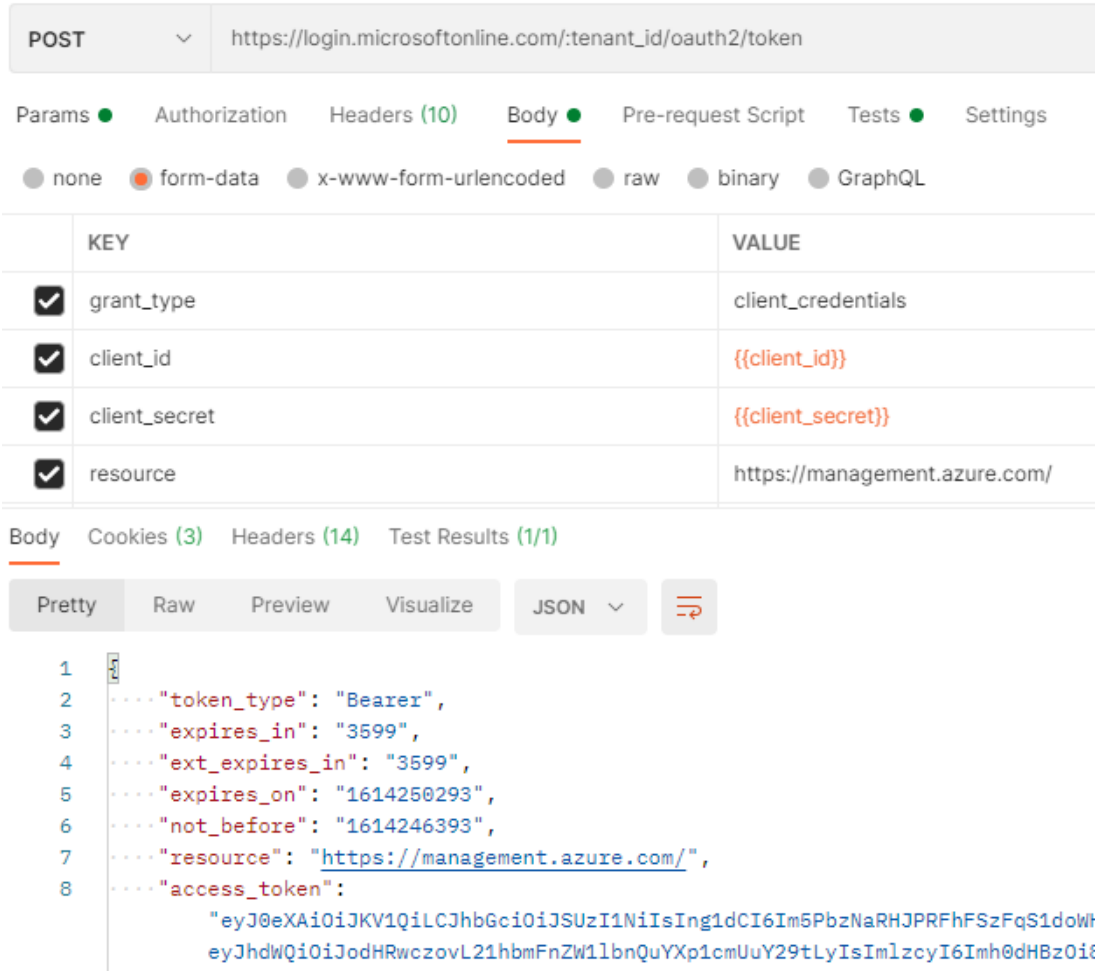
Bearer tokenia käytetään HTTP-pyynnön *Authorization*-otsikossa (engl. header). Tokenilla voi autentikoitua vain, jos yhteys on TLS-salattu (Transport Layer Security), koska selkokielellä lähetettynä se voisi joutua vääriin käsiin. (Jones & Hardt 2012.) TLS-yhteyden käyttäminen varmistaa tiedon salaamisen lisäksi myös tiedon eheyden. Salatut viestit sisältävät HMAC-osion (Hashed Message Authentication Code), joka muodostuu viestistä sekä TLS-kättelyssä (engl. handshake) muodostetusta yhteisestä avaimesta. (Dierks & Rescorla 2008.) HMAC-arvo saadaan yhteisestä avaimesta ja viestistä kaavalla

$$HMAC_{\mathrm{k}}(m) = H\big((k \oplus opad), H((k \oplus ipad), m)\big),$$

jossa *k* on yhteinen avain, *m* on viesti, *H* on hajautusfunktio, joka ottaa kaksi argumenttia ja *ipad* sekä *opad* ovat hajautetun lohkon kokoisia vakioita (Ryan & Galindo 2017).

HMAC arvon ottamiseksi viesti hajautetaan kahteen kertaan: ensin avaimella, joka on muunnettu eksklusiivisella disjunktiolla (XOR) *ipad* -vakion kanssa, ja sitten avaimella, joka on muunnettu eksklusiivisella disjunktiolla *opad* -vakion kanssa. Koska kerran hajautettu viesti hajautetaan toisen kerran eri avaimella, viestin alkuperäistä sisältöä on mahdotonta selvittää ilman yhteistä avainta. (Ryan & Galindo 2017.)

Azure Sentinel REST APIa varten luotu bearer token saatiin POST-kyselyllä Microsoftin OAuth2-palvelun resurssiin */{tenant_id}/oauth2/token*. Kyselyn otsikkokenttään määritellään mihin resurssiin tokenilla voi autentikoitua. (Microsoft identity platform and the OAuth 2.0 client credentials flow 2020.) Kuviossa 7 esitellään API-kysely, jolla liitosohjelmisto hakee tokenin OAuth2-palvelun ohjelmointirajapinnalta. Liitosohjelmisto ottaa kyselyn palauttamista tiedoista muistiin *access_token* rivin arvon, jota käytetään Azure Sentinel REST APIn tunnistautumiseen.



Kuvio 7. Bearer tokenin vastaanottaminen APIlta

4.5 Pelikirjatoiminnallisuuksien kehittäminen

Liitosohjelmiston tehtäviin kuuluu yhteyden muodostamisen ja autentikaation lisäksi mahdollistaa pelikirjoissa käytettävät toiminnallisuudet. Jokaisen *operations.py* tiedoston aliohjelman voi määritellä pelikirjatoiminnallisuudeksi ohjelmiston konfiguraatiotiedostossa *info.json*. Konfiguraatiossa pelikirjatoiminnallisuudelle annetaan SOARissa näkyvä nimi ja kuvaus, määritellään pelikirjassa käytettävät parametrit ja kerrotaan mitä JSON-rivejä pelikirja palauttaa. SOAR olettaa, että onnistunut pelikirjatoiminnallisuus palauttaa konfiguraatiotiedostossa ilmoitetut rivit, joten siellä ilmoitettuja JSON-rivejä voi käyttää muuttujina pelikirjassa. (SOAR-järjestelmän dokumentaatio 2020.)

Kaikki pelikirjatoiminnallisuudet hakevat ensimmäiseksi bearer tokenin resurssiin *https://management.azure.com/* Micorosftin OAuth2-palvelusta kuvion 7 mukaisesti. Tätä tokenia käytetään seuraavien API-kyselyiden *Authorization*-otsikossa. Pelikirjatoiminnallisuuksien loput API-kyselyt kohdistuvat Azure Sentinel REST APIin.

SOAR lokittaa automaattisesti jokaisen pelikirjan ajamisesta kuka käyttäjä sen on suorittanut, mitä toimenpiteitä pelikirjan askeleiden aikana on tehty sekä askeleiden palauttamat tiedot. SOARin lokitusominaisuuksien ansiosta vaatimusmäärittelyn ei-toiminnalliset vaatimukset SEC_REQ_001 ja SEC_REQ_002 täyttyvät, eikä niitä näin ollen tarvitse ottaa huomioon liitosohjelmiston kehittämisessä. Myös vaatimukset SEC_REQ_004 ja SEC_REQ_005 täyttyvät, sillä kaikki liikenne SOARin ja Microsoftin ohjelmointirajapintojen välillä on TLS-salattua. (ks. liite 3.)

4.5.1 Uusien tietoturvapoikkeamien hakeminen

Liitosohjelmiston tärkein toiminnallisuus on saada haettua ohjelmointirajapinnalta automaattisesti tietoa uusista tietoturvapoikkeamista (ks. taulukko 2). Tähän tarkoitukseen kehitetty pelikirjatoiminnallisuus hakee Sentinelistä kaikki tietoturvapoikkeamat viimeisen tunnin ajalta. Toiminnallisuuden ajava pelikirja toteutetaan kahden minuutin välein. Näin varmistetaan, että Sentinelistä saadaan haettua kaikki tietoturvapoikkeamat, vaikka yksittäiset kyselyt epäonnistuisivat tai niissä menisi aikaa. Jokaisesta tietoturvapoikkeamasta luodaan kuitenkin vain yksi hälytys SOARiin, vaikka pelikirja hakisi saman poikkeaman useaan kertaan tunnin aikana.

Toiminnossa käytetään GET-pyyntöä ohjelmointirajapinnan resurssiin */incidents*, joka palauttaa lähtökohtaisesti kaikki olemassa olevat tietoturvapoikkeamat Sentinelistä. Toimeksiantajan käyttötapauksessa riittää, että haetaan kerrallaan poikkeamat viimeisen tunnin ajalta. Tähän tarkoitukseen pelikirjatoiminnallisuus käyttää hyväkseen ohjelmointirajapinnan *$filter* parametria, joka suodattaa haetut tietoturvapoikkeamat luontiajan eli attribuutin *properties/createdTimeUtc* perusteella. (Azure Sentinel 2020.) Tällä kyselyllä saadaan riittävästi tietoa uusista poikkeamasta hälytyksen luomiseksi SOARiin, mutta ohjelmointirajapinnan palauttama data ei sisällä tietoa poikkeamaan liittyvistä hälytyksistä tai uhkaindikaatioista (ks. liite 6).

4.5.2 Tietoturvapoikkeaman tilan muutos

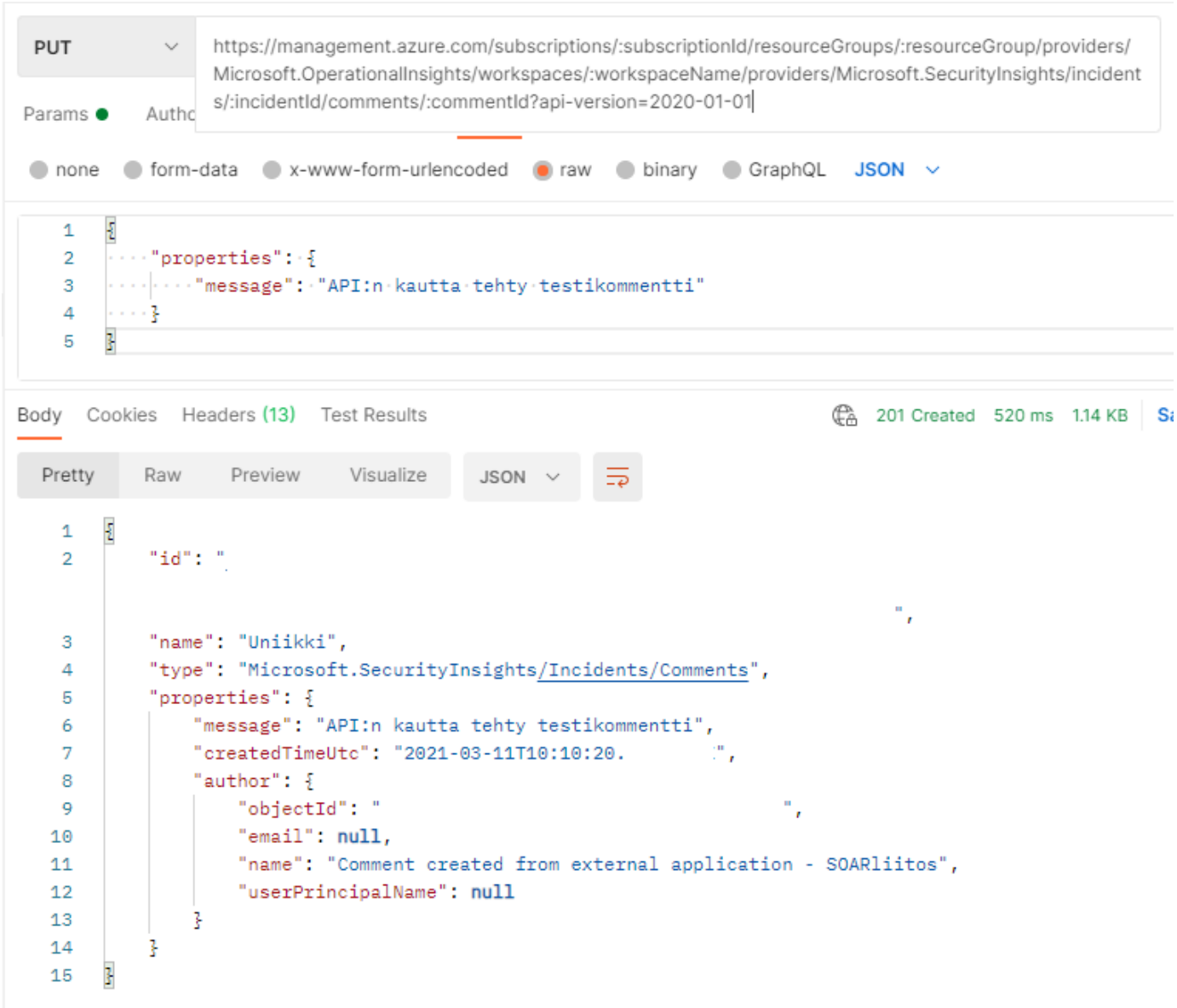
Tietoturvapoikkeamaan saa tehtyä muutoksia PUT-pyynnöllä ohjelmointirajapinnan resurssiin */incidents/{incidentId}*. Kaikki muutettavat tietueet pitää ilmoittaa HTTP-kyselyn leipätekstissä JSON-muodossa. Tietoturvapoikkeaman ID:n lisäksi pitää tietää poikkeaman sen hetkinen ETag-arvo. ETag on HTTP otsikko, jonka tarkoituksena on yksilöidä tietty resurssin versio (ETag 2021). API palauttaa HTTP tilakoodin *409 Conflict* mikäli ETag-otsikko puuttuu tai eroaa nykyisen poikkeaman versiosta. Muokattavan poikkeaman sen hetkinen ETag-arvo haetaan GET-pyynnöllä ohjelmointirajapinnan resurssiin */incidents/{incidentId}*, joka palauttaa kyseisen poikkeaman tiedot sisältäen ETag-rivin. Tämän arvo otetaan talteen ja se lähetetään muutoskyselyn leipätekstissä muutettavien tietueiden kanssa. API-kyselyn leipätekstissä on oltava määriteltynä ainakin kuviossa 8 esitellyt rivit. (Azure Sentinel 2020.)

```
{
  "etag": "{ETag}",
  "properties": {
    "severity": "{Severity}",
    "status": "{Status}",
    "title": "{Title}"
  }
}
```

Kuvio 8. API-kyselyn leipätekstin pakolliset JSON-rivit

4.5.3 Kommentin lisääminen tietoturvapoikkeamaan

Kommentin saa lisättyä olemassa olevaan tietoturvapoikkeamaan PUT-pyynnöllä APIn resurssiin */incidents/{incidentId}/comments/{commentId}*, jossa *{commentId}* on mielivaltainen, mutta uniikki nimi lisättävälle kommentille. Liitosohjelmaan kehitettiin pelikirjatoiminnallisuus, joka mahdollistaa kommentin lisäämisen SOARin hälytyksestä vastaavaan Sentinelin tietoturvapoikkeamaan. Pelikirjatoiminnallisuus tarvitsee toimiakseen parametreinä Sentinelin tietoturvapoikkeaman ID:n ja lisättävän kommentin tekstisisällön, jotta kuvion 9 API-kyselyn URL-parametri *:indicentId* sekä HTTP-kyselyn leipäteksti saadaan täytettyä.

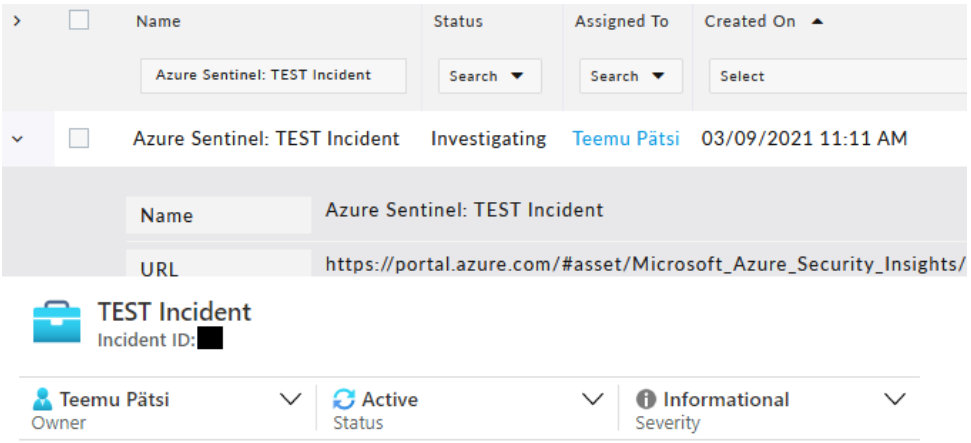


Kuvio 9. API-kysely luo uuden kommentin Sentinelin tietoturvapoikkeamaan

Kommentin lisäämisen mahdollistavaa pelikirjatoiminnallisuutta käytetään hyväksi pelikirjassa, joka siirtää Sentineliin liittyvään SOARin hälytykseen lisätyt kommentit automaattisesti vastaavaan Sentinelin poikkeamaan kuvion 1 mukaisesti. SOARin hälytys sisältää liittyvän Sentinelin poikkeaman ID:n ja kommentin sisältöön laitetaan SOARin kommentin sisältö. Testausvaiheessa pitää varmistaa, etteivät Sentinelin tekstikentät käsittele saamaansa tekstiä koodina tai komentoina, koska SOAR-käyttäjän syöte laitetaan sellaisenaan järjestelmästä toiseen.

4.6 Tietoturvapoikkeamien käsittely SOARissa

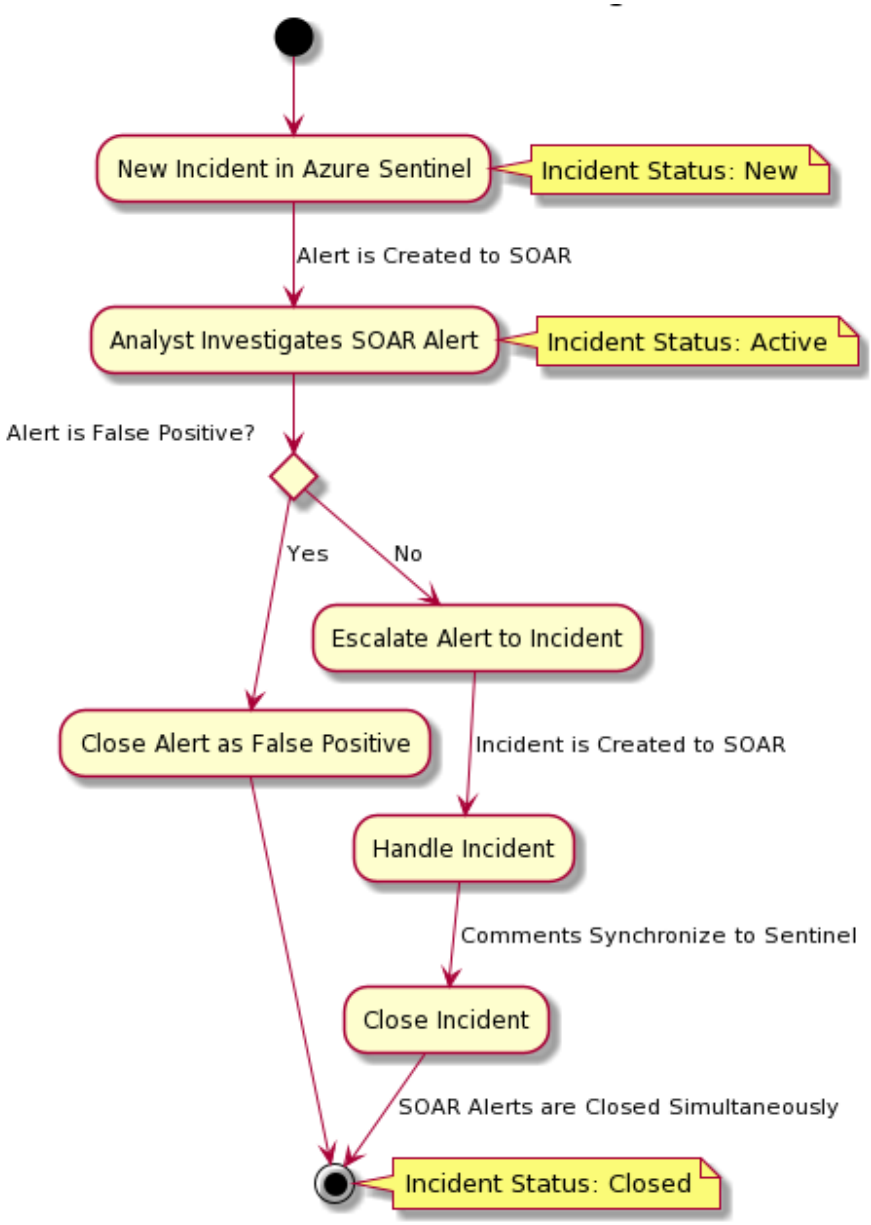
Sentinelin uudesta tietoturvapoikkeamasta muodostuu hälytys SOARiin *New*-tilassa. Uusi hälytys tulee SOC-analyytikoiden nähtäville hälytysjonoon. Kun analyytikko ottaa hälytyksen käsittelyyn se muuttuu *Investigating*-tilaan ja analyytikko määrittyy sen vastuuhenkilöksi. Aina kun SOARin hälytyksen tila muuttuu, tilatieto päivittyy Sentineliin kuvion 10 mukaisesti pelikirjalla, joka tekee Sentinelin tietoturvapoikkeamaan muutoksen kappaleessa 4.5.2 esitellyllä API-kyselyllä.



Kuvio 10. Käsittelyyn otettu SOARin hälytys asettaa Sentinelin poikkeaman aktiiviseksi

SOC-analyytikko pääsee hälytyksen URL-kentän linkistä suoraan Sentineliin tutkimaan poikkeamaa ja siihen liittyviä hälytyksiä. Mikäli kyseessä on väärä hälytys (engl. false positive) analyytikko voi sulkea hälytyksen SOARista, jolloin Sentinelin hälytys sulkeutuu samalla. Hälytys eskaloidaan SOARissa tietoturvapoikkeamaksi, mikäli hälytys vaatii toimenpiteitä SOCilta tai asiakkaalta. SOARin tietoturvapoikkeamaan dokumentoidaan tehdyt toimenpiteet. Poikkeaman voi myös tarvittaessa eskaloida asiakkaalle, jolloin hälytys näkyy asiakkaan palveluportaalissa tikettinä.

SOARin poikkeamaan merkatut kommentit synkronoituvat myös Sentinelin poikkeamaan kommenteiksi, jotta toimenpiteiden dokumentointi on saatavilla myös Sentinelissä. Tapahtuman käsittelyn jälkeen SOARin poikkeama suljetaan, joka sulkee automaattisesti myös siihen liitetyt hälytykset ja näin ollen myös hälytyksiin liittyvät Sentinelin tietoturvapoikkeamat. Sentinelin tietoturvapoikkeaman käsittelyn vaiheet on kuvattu kuvion 11 vuokaaviossa.



Kuvio 11. Vuokaavio Sentinelin tietoturvapoikkeaman käsittelystä SOARin kautta

5 Ohjelmiston testaaminen

5.1 Testaamisen tavoitteet ja tekniikat

Kehitetyn integraation testaamisen tavoitteena oli varmistaa vaatimusmäärittelyssä esiteltyjen toiminnallisuuksien toimiminen halutulla tavalla. Jokainen integraation ensimmäiseen versioon toteutettu toiminnallisuus testattiin. Toimeksiantajan edustaja oli mukana testeissä varmistaakseen, että toiminnallisuudet toimivat halutulla tavalla. Toiminnallisuuksien testeissä myös tarkastettiin, että liitteen 3 ei-toiminnallisten vaatimusten asettamat rajoitukset ja reunaehdot toteutuvat.

Palvelun testaaminen tuotantoympäristössä ei ole suotavaa mahdollisten toimintavirheiden vuoksi, joten testaamista varten käytettiin testaukseen sopivaa ympäristöä sekä SOARista, että Sentinelistä. Tuotantoon tuleviin pelikirjoihin lisättiin ylimääräisiä muuttujia, joilla testattiin ajon aikana minkälaisia arvoja pelikirjojen eri kohdat palauttavat. Liitosohjelmiston testaamista varten Sentineliin luotiin eri vakavuustason omaavia tietoturvapoikkeamia APIn kautta liitteen 7 koodilla. Myöhemmin kyseistä koodia käytettiin hyväksi pienin muutoksin erilaisten poikkeamien luomiseen. Kehitystyötä jatkettiin testaamisen aikana, kun virhetilanteita tuli vastaan, jotta testitulokset saatiin riittävän hyviksi lopulliseen versioon.

5.2 Tietoturvapoikkeamien hakeminen automaattisesti

Liitosohjelmiston taulukossa 2 esitellyistä toiminnallisuuksista tärkein on hakea automaattisesti Sentineliin muodostuneet uudet tietoturvapoikkeamat ja luoda niistä hälytykset SOARiin. Tällöin analyytikko saa tiedon uudesta tapahtumasta SOARin hälytysjonoon ja voi reagoida siihen. Tähän ominaisuuteen liittyvät testit oli saatava ehdottomasti suoritettua ilman virheitä hyväksytyillä kriteereillä, koska tämä on ohjelmiston kriittisin toiminnallisuus. Testit suoritettiin automatisoidusti luomalla Sentineliin haluttu määrä tietynlaisia uusia tietoturvapoikkeamia muokkaamalla liitteen 7 koodia tarpeen mukaan ja seuraamalla samalla Sentinelin ja SOARin tapahtumia.

Ominaisuuteen liittyvät ei-toiminnalliset vaatimukset testattiin ajastamalla pelikirjan ajoa eri tiheyksille ja luomalla testissä tarvittava määrä testipoikkeamia Sentineliin. Ohjelmiston suorituskykyyn liittyvä vaatimus PERF_REQ_001 testattiin tuotannonomaisesti ajastamalla pelikirja suoriutumaan kahden minuutin välein vuorokauden ajaksi. Vuorokauden aikana pelikirjan ajaminen

epäonnistui vain kerran, jolloin syynä oli SOARin sisäisen APIn palauttama virhe. Näin ollen toiminnallisuuden toimintavarmuus ylitti helposti vaaditun 99%. Vaatimuksen PERF_REQ_002 testaamiseksi kahden minuutin sisällä luotiin sata tietoturvapoikkeamaa Sentineliin. (ks. liite 3.) Suorituskykyvaatimus täyttyi, sillä jokainen luotu poikkeama muodostui SOARiin hälytykseksi pelikirjan seuraavan ajastetun ajon aikana kuvion 12 mukaisesti.

Azure Sentinel: TEST Incident: Medium	Investigating	Teemu Pätsi	03/19/2021 11:14 AM
Azure Sentinel: TEST Incident: High	Investigating	Teemu Pätsi	03/19/2021 11:14 AM
Azure Sentinel: TEST Incident: Informational	Investigating	Teemu Pätsi	03/19/2021 11:14 AM
Azure Sentinel: TEST Incident: Low	Investigating	Teemu Pätsi	03/19/2021 11:14 AM

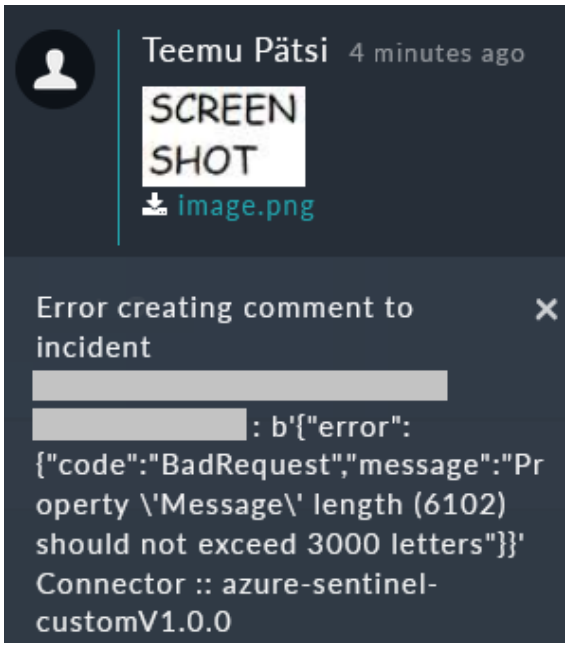
Kuvio 12. Automaattisesti luodut Sentinelin poikkeamat muodostuivat SOARiin hälytyksiksi

5.3 SOARin hälytysten tilan muutosten synkronoituminen Sentineliin

SOARin hälytyksen tilan muuttaminen pitäisi synkronoitua automaattisesti Sentinelin vastaavaan tietoturvapoikkeamaan. Toiminnallisuuden testaamisessa käytettiin kappaleessa 5.2 luotuja hälytyksiä. Ensin kaikki mahdolliset SOARin hälytyksen tilat käytiin läpi järjestyksessä samalla varmistaen, että vastaavan Sentinelin tietoturvapoikkeaman tila vaihtuu halutulla tavalla. Tämän jälkeen toisella hälytyksellä käytiin läpi mahdolliset tilat eri järjestyksessä samalla varmistaen, että muutokset synkronoituvat Sentineliin oikein. Lopulta testattiin liitteen 3 vaatimus PERF_REQ_003 muuttamalla kerralla sadan kappaleessa 5.2 luodun hälytyksen tilaa samanaikaisesti. Kuviossa 1 esitelty SOARin pelikirja ajautui tällöin sata kertaa ja muutti kaikkien liittyvien Sentinelin tietoturvapoikkeamien tilan muutaman sekunnin aikana.

5.4 Tietoturvapoikkeaman kommentointi SOARin kautta

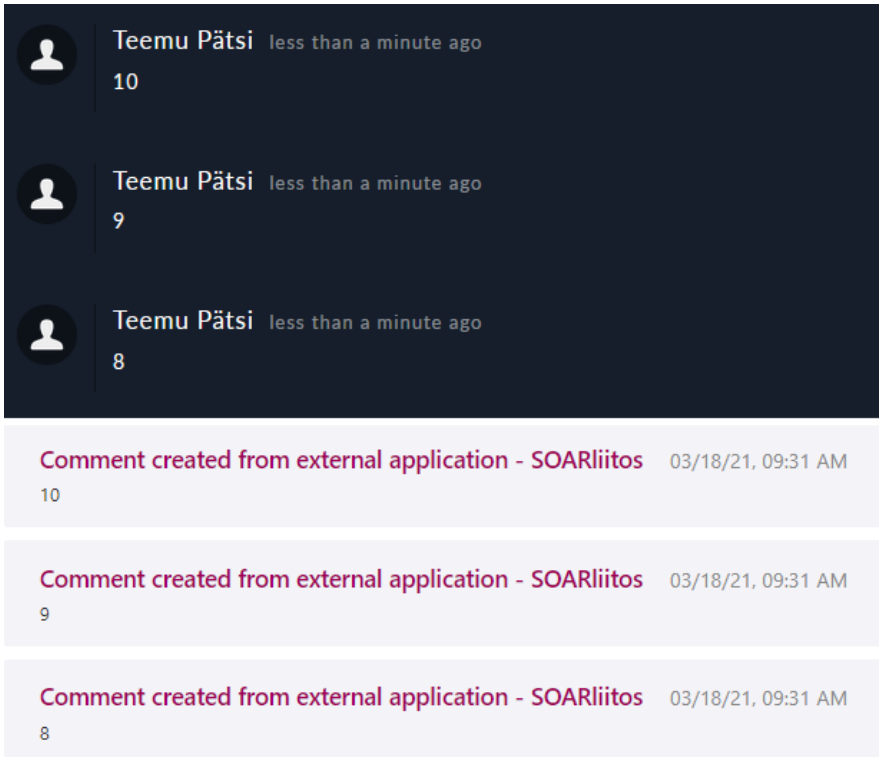
SOARin pelikirjan pitäisi lähettää uusin kommentti vastaavaan Sentinelin tietoturvapoikkeamaan. Kommentti muodostuu Sentinelin tietoturvapoikkeamaan kuviossa 1 esitellyllä tapahtumaketjulla. Ensiksi todettiin, että toiminnallisuus toimii normaalilla kommentilla. Tämän jälkeen käytettävyyttä testattiin laittamalla SOARin tietoturvapoikkeamaan erilaisia syötteitä sisältäviä kommentteja. Syötteinä testattiin näppäimistön erikoismerkkejä, emojeita, kuvankaappauksia, HTML-tageja sekä skandimerkkejä *å*, *ä* ja *ö*. Näistä ainoastaan kuvankaappaukset eivät toimineet syötteenä, koska API ei suostu ottamaan vastaan yli 3000 merkkiä pitkiä kommentteja. Ohjelmointirajapinta palautti kuvion 13 mukaisesti HTTP-tilakoodin *400 Bad Request* kun yritti lähettää kuvaa kommenttina. Sen sijaan emojit ja erikoismerkit näkyivät Sentinelissä oikein, joten liitteen 3 käytettävyysvaatimus USAB_REQ_001 täyttyy.



Kuvio 13. Azure Sentinel REST API ei hyväksy yli 3000 merkkiä pitkiä kommentteja

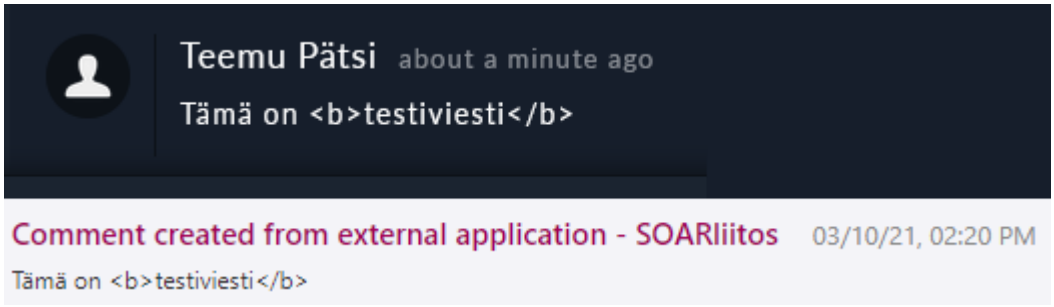
Toimintavarmuutta testattiin luomalla käsin mahdollisimman nopeasti kymmenen uutta kommenttia SOARin tietoturvapoikkeamaan samalla varmistaen, että kaikki kommentit saapuvat Sentinelin vastaavaan poikkeamaan. Kaikki kommentit synkronoituivat Sentineliin parin sekunnin sisällä kommentin lisäämisestä SOARiin kuvion 14 mukaisesti. Kommenttien sykronointiominaisuus kuitenkin koettiin liian automaattiseksi, sillä analyytikoilla voi olla tarve tehdä kommentteja ainoastaan SOARin sisällä. Tästä syystä automaattisynkronoinnista luovuttiin ja Sentinelin

poikkeamista luotuihin SOARin tietoturvapoikkeamiin tehtiin manuaalinen mahdollisuus lisätä uusin kommentti Sentineliin.



Kuvio 14. SOARiin tehdyt kommentit synkronoituivat Sentinelin poikkeamaan

Ohjelmointirajapinnan tietoturvallisuutta sai testattua laittamalla kommentteihin syötteeksi HTML-tageja. Kuten kuviosta 15 näkee, Sentinel validoi käyttäjän syötteen ainakin HTML-tagien varalle ja käsittelee niitä HTML-tagien sijasta pelkkänä tekstinä. Näin ollen toiminnallisuus ei mahdollista käyttäjän syötteen aiheuttamia hyökkäyksiä kuten cross-site scripting -hyökkäystä (XSS) ja liitteessä 3 esitelty ei-toiminnallinen vaatimus SEC_REQ_003 täyttyy.



Kuvio 15. SOARin pelikirja lisäsi kommentin Sentinelin tietoturvapoikkeamaan

6 Tulokset

Kehitystyön tuloksena oli tuotantoon käyttöönotettu Azure Sentinelin liitosohjelmisto SOARissa. Integraation ansiosta Sentineliin muodostuvat tietoturvapoikkeamat saadaan automaattisesti SOARin hälytysjonoon. SOCin analyytikoiden ei tämän ansiosta tarvitse erikseen seurata Sentinelin poikkeamia, vaan he saavat tiedon uusista poikkeamista keskitettyyn hälytysjonoon. Sentinelistä haetuista poikkeamista muodostuvista SOAR-hälytyksistä löytyy linkki, josta analyytikko pääsee suoraan Sentineliin tutkimaan poikkeamaa sekä siihen liittyviä tapahtumia ja hälytyksiä.

Hälytyksien muodostamisen lisäksi integraatio mahdollistaa Sentinelin tietoturvapoikkeamien tilan vaihdoksen ja kommentoimisen SOARin kautta. Kommenttien siirtäminen SOARista Sentineliin yksinkertaistaa käsittelyprosessia, sillä analyytikon ei tarvitse kirjata työvaiheita ja toimenpiteitä kahteen eri paikkaan. Hälytystä ei myöskään tarvitse erikseen merkitä aktiiviseksi tai suljetuksi Sentinelin puolella.

Integraation kehitystyö jatkuu käyttöönottamisen jälkeen. Liitosohjelmiston ensimmäisestä versiosta jäi puuttumaan muutamia vaatimusmäärittelyssä esiteltyjä hyödyllisiä tai toivottuja toiminnallisuuksia. Liitoksen seuraavan version avulla SOARin hälytykseen pitäisi saada rikastettua mahdollisimman paljon oleellisia tietoja poikkeamaan liittyvistä hälytyksistä ja tapahtumista, jotta analyytikot voisivat tutkia tapahtumaa suoraan SOARin kautta mahdollisimman tehokkaasti. Taulukossa 2 määriteltyjen toiminnallisuuksien toteutuminen on esitelty liitteessä 4.

Vaatimusmäärittelyssä liitosohjelmistolle määriteltiin toiminnallisia ja ei-toiminnallisia vaatimuksia. Ei-toiminnalliset vaatimukset asettivat ohjelmiston toiminnallisuuksille reunaehtoja ja rajoituksia, joista kaikki otettiin huomioon ohjelman kehityksessä. Liitteessä 2 määriteltyjen toiminnallisten vaatimusten toteutuminen ja niiden testaamiseen käytetyt testit on esitelty liitteessä 5.

7 Pohdinta

Kehitystyön tavoitteena oli kehittää integraatio SOAR-järjestelmän ja Sentinel-palvelun välille. Tavoite saavutettiin siltä osin, että Sentinelin hälytykset saadaan automaattisesti muodostumaan SOARiin hälytyksinä ja hälytyksen käsittelyprosessin mukaiset toimenpiteet hälytykselle synkronoituvat automaattisesti vastaavaan Sentinelin poikkeamaan. Valitettavasti käytettäväksi valituilla tekniikoilla ei saada poikkeamiin liittyvistä hälytyksistä riittävän rikastettua dataa, jotta tapausten käsittely SOARin kautta olisi realistista. Onneksi SOARiin muodostuvat hälytykset sisältävät linkin suoraan Sentinelin tietoturvapoikkeamaan, joten käsittelyyn käytetyn ajan ei pitäisi merkittävästi erota, vaikka analyytikko joutuu käyttämään kahta järjestelmää.

Opinnäytetyön tuloksena oli SOARissa toimiva liitosohjelmisto, jonka toiminnallisuuksia saa ajettua SOARin pelikirjojen kautta. Liitosohjelmiston kehittäminen oli luontainen ratkaisu Sentinelin tietoturvapoikkeamien hakemiseksi SOARiin, koska Microsoftilla oli kaksi ohjelmointirajapintaa, jota voisi käyttää hyväksi tiedon hakemiseen Azuresta. Liitoksen tärkein toiminnallisuus, tietoturvapoikkeamien hakeminen Sentinelistä, oli helppo toteuttaa yhdellä API-kyselyllä ja hälytyksien hakemisen Sentinelistä SOARiin saatiin testattua toimivaksi hyvin aikaisessa vaiheessa kehitystyötä. Kehitystyön nopeus ja ohjelmiston helppo arkkitehtuuri tekivät ratkaisusta järkevän, vaikka Microsoftin ohjelmointirajapinnat antoivat omat rajoitteensa ohjelmiston toiminnallisuuksille.

Valmis liitosohjelmisto hakee Sentinelistä uusia poikkeamia kahden minuutin välein ja sen toimintavarmuus täyttää vaatimukset. Ohjelmisto hakee jokaisella ajolla takautuvasti tunnin ajalta uudet poikkeamat, joten on erittäin epätodennäköistä, että poikkeamia jäisi normaalitilanteessa huomaamatta. Kuitenkin on mahdollista, että SOARilla on ongelmia esimerkiksi tietoliikenneyhteyksien tai pelikirjojen ajamisen kanssa. Tässäkin tapauksessa SOARin pitäisi muodostaa hälytys järjestelmän ongelmasta. Mikäli jostain syystä näin ei ole, SOAR-järjestelmässä on todennäköisesti jotain isompaakin ongelmaa, joka näkyisi käyttäjille. Näissä tapauksissa analyytikon täytyy tarkastaa manuaalisesti, toimiiko integraatio normaalisti. Sentinel-integraation tapauksessa tämän saa tehtyä ajamalla manuaalisesti pelikirjan, joka hakee Sentinelistä kaikki uudet tietoturvapoikkeamat.

Kehitystyön aikana meni huomattavasti aikaa sen selvittämiseen, kuinka Sentinelin poikkeaman hälytyksien tiedot saisi rikastettua SOARiin käyttäen Microsoftin ohjelmointirajapintoja. Lopulta päädyttiin siihen tulokseen, että puuttuvat toiminnot pitäisi tehdä Azuren puolella, sillä ohjelmointirajapinnoilla ei ollut tarvittavia toiminnallisuuksia. Azuren kautta voisi lähettää tietoa tietoturvapoikkeamiin liittyvistä hälytyksistä esimerkiksi Syslog-muodossa, jonka avulla SOAR saisi rikastettua Sentinelin hälytyksiä. Seuraamme myös aktiivisesti Azure Sentinel REST APIn kehitystä, sillä sen seuraavat versiot voisivat sisältää uusia toiminnallisuuksia.

Käyttöönotetusta integraatiosta jäi Microsoftin ohjelmointirajapintojen puutteiden vuoksi puuttumaan useita vaatimusmäärittelyssä esiteltyjä toiminnallisuuksia. Azure Sentinel REST APIsta puuttuu kokonaan mahdollisuus hakea tietoa poikkeamaan liittyvistä hälytyksistä. Tietoturvahälytysten hakeminen mahdollistuisi käyttämällä Microsoft Graph APIa. Kehittämisprosessin aikana ei kuitenkaan selvinnyt kuinka Graph APIlta haettujen hälytysten ja Sentinelin tietoturvapoikkeamien yhteys saataisiin varmistettua, sillä ne käyttävät eri ID:tä samasta hälytyksestä. Tuotanto-ohjelmistoon ei haluttu tehdä esimerkiksi hälytysten nimiin tai aikaleimoihin perustuvia relaatioita, koska ainoastaan ID:n perusteella saadaan varmistettua mitkä hälytykset liittyvät tiettyyn poikkeamaan. Poikkeamien käsittelyn automatisointia se ei kuitenkaan haittaa, ettei SOARin hälytyksiin rikastettava data sisällä riittävästi tietoa, sillä Sentinel on itsessään SOAR-palvelu. Työn suunnittelun aikana todettiin, että tietoturvapoikkeamien tutkimisen automatisointi on järkevämpää toteuttaa SOARin sijaan Sentinelin pelikirjoilla.

Azure Sentinel REST APIn toiminnallisuuksien puutteet vaikuttavat johtuvan siitä, että se on varsin uusi ohjelmointirajapinta vanhemman Graph APIn rinnalla. Havaitut puutteet voisivat myös johtua siitä, ettei Microsoft halua Sentinelin tapahtumia tutkittavan Azuren ulkopuolella. Järjestelmän tavoite on kuitenkin olla Microsoftin pilvessä toimiva SIEM- ja SOAR-palvelu. Näin ollen olisi Microsoftin edun mukaista, että mahdollisimman usean järjestelmän tietoturvatapahtumia tuotaisiin Sentineliin analysoitavaksi sen sijasta, että niitä haettaisiin sieltä tutkittavaksi toiseen järjestelmään.

Tuotannossa olevasta liitoksesta ei saatu kerättyä palautetta käyttäjiltä, sillä Sentinelin kautta ei tule vielä yhdenkään asiakkaan tietoturvahälytyksiä. Keskusteluiden perusteella yleinen mielipide kuitenkin on, että liitoksen ensimmäinen versio sisältää riittävästi toiminnallisuuksia ollakseen hyödyllinen. SOARiin luodun hälytyksen tärkeimmäksi rikastustiedoksi koettiin linkki suoraan Sentinelin tietoturvapoikkeamaan. Liitosohjelmiston tärkeimmäksi toiminnallisuudeksi koettiin tilatietojen automaattinen muutos kahden järjestelmän välillä. Liitoksen suurimmaksi heikkoudeksi koettiin Sentinelin hälytystietojen rikastamisen puutteet SOARin hälytyksessä, jonka vuoksi tapauksia on mahdotonta tutkia SOARin kautta.

Tämä kehitystyö oli CSOC-tiimin ensimmäinen itse tehty SOAR-liitosohjelmisto. Työn merkitys korostunee jatkossa, kun tarvitaan uusia palveluliitoksia. Sen ansiosta tiimissä on kokemusta liitosohjelmiston kehitystyön vaiheista ja kerättyä tietoa saadaan koulutettua eteenpäin. Työn aikana toimeksiantajalle dokumentoitiin kehitystyön aikana huomioitavia asioita sekä hyväksi koettuja käytänteitä. Liitosohjelmiston arkkitehtuurin ja käyttöönottoprosessin ymmärtäminen sekä kerätty dokumentaatio tekee tulevien liitoksien kehitystyöstä nopeampaa ja vähemmän virhealtista.

Lähteet

azsec. 2020. Get Alert Relation from an Incident using Azure Sentinel Incident Relation API. Blogikirjoitus Azure Sentinel Incident Relation APIsta. Viitattu 26.2.2020. https://azsec.azurewebsites.net/2020/07/15/get-alert-relation-from-an-incident-using-azure-sentinel-incident-relation-api/

Azure Sentinel. 2020. Microsoftin Azure Sentinel REST API -dokumentaatio. Viitattu 17.3.2021. https://docs.microsoft.com/en-us/rest/api/securityinsights/

Cohn, M. 2004. User Stories Applied: for Agile Software Development. Viitattu 25.2.2021. http://athena.ecs.csus.edu/~buckley/CSc191/User-Stories-Applied-Mike-Cohn.pdf

Dierks, T & Rescorla, E. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. IETF:n julkaisema Request for Comments (RFC) dokumentaatio. Viitattu 31.3.2021. https://tools.ietf.org/html/rfc5246

ETag. 2021. Mozillan dokumentaatio ETag HTTP-otsikosta. Viitattu 17.3.2021. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag

Fielding, R. 2000. Architectural Styles and the Design of Network-based Software Architectures. Väitöskirja, filosofian tohtori. Californian yliopisto, Irvine, Information and Computer Science. Viitattu 23.2.2020. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Gallant, J. 2017. Azure REST APIs with Postman (March 2017). Microsoftin Azure REST APIn dokumentaation suosittelema video APIin autentikoitumiseen Postman-ohjelmalla. Viitattu 25.2.2021. https://www.youtube.com/watch?v=ujzrq8Fg9Gc

Imam, F. 2019. Security Orchestration, Automation and Response (SOAR). Artikkeli Infosec Instituten verkkosivustolla. Viitattu 24.3.2021. https://resources.infosecinstitute.com/topic/security-orchestration-automation-and-response-soar/#gref

Jelen, S. 2020. Security Automation: Definition, Benefits, Best Practices and Tools. Blogikirjoitus SecurityTrailsin verkkosivustolla. Viitattu 22.2.2021. https://securitytrails.com/blog/security-automation

JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely. 2018. Julkisen tietohallinnon neuvottelukunta JUHTAn suositus vaatimusmäärittelylle. Versio 1.2. Viitattu 11.3.2021. https://www.suomidigi.fi/sites/default/files/2020-06/JHS173.doc

Jones, M & Hardt, D. 2012. The OAuth 2.0 Authorization Framework: Bearer Token Usage. IETF:n julkaisema Request for Comments (RFC) dokumentaatio. Viitattu 25.2.2021. https://tools.ietf.org/html/rfc6750

Lucassen, G, Dalpiaz, F, Van der Werf, J & Brinkkemper, S. 2016. The Use and Effectiveness of User Stories in Practice. DOI: 10.1007/978-3-319-30282-9_14. Viitattu 25.2.2021 https://www.researchgate.net/profile/Fabiano-Dalpiaz/publication/312702795_The_Use_and_Effectiveness_of_User_Stories_in_Practice/links/5b61642c0f7e9bc79a72dc21/The-Use-and-Effectiveness-of-User-Stories-in-Practice.pdf

Microsoft identity platform and the OAuth 2.0 client credentials flow. 2020. Microsoftin OAuth2 autentikaation dokumentaatio. Viitattu 29.3.2021. https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-client-creds-grant-flow

Niesen, J. 2020. Getting Azure Sentinel Entities by Rest API (an undocumented feature). Artikkeli ohjelmointirajapinnan dokumentoimattomasta toiminnallisuudesta. Viitattu 30.3.2021. https://medium.com/wortell/getting-azure-sentinel-entities-by-rest-api-f5ff56ca2edd

Postman Tutorial: How to use Postman Tool for API Testing. 2020. Postman-ohjelman ohjeistus aloittelijoille. Viitattu 25.2.2021. https://www.guru99.com/postman-tutorial.html

Ryan, M & Galindo, D. 2017. Message authentication codes (MACs). Birminghamin yliopiston luentomateriaali. Viitattu 31.3.2021 https://www.cs.bham.ac.uk/~mdr/teaching/crypto19/crypto6.pdf

SOAR-järjestelmän dokumentaatio. 2020. Toimeksiantajan julkaisematon dokumentaatio.

Townsend, K. 2018. Fighting Alert Fatigue With Security Orchestration, Automation and Response. Kolumni Securityweekin verkkosivuilla. Viitattu 22.2.2021. https://www.securityweek.com/fighting-alert-fatigue-security-orchestration-automation-and-response

Use the Microsoft Graph Security API. 2021. Microsoftin Graph Security APIn dokumentaatio. Viitattu 29.3.2021. https://docs.microsoft.com/en-us/graph/api/resources/security-api-overview?view=graph-rest-1.0

Viria Oyj Tilinpäätöstiedote. 2021. Virian tilinpäätöstiedote. Viitattu 22.3.2021. https://www.viria.fi/wp-content/uploads/2021/03/Viria-Oyj-Tilinpaatostiedote-2020.pdf

What is Azure Sentinel? 2020. Kuvaus Azure Sentinel -palvelusta Microsoftin dokumentaatiossa. Viitattu 22.2.2021. https://docs.microsoft.com/en-us/azure/sentinel/overview

Liitteet

Liite 1. CSOC-tiimin analyytikoiden vastaukset kyselyyn integraation vaatimuksista

Mitä toimintoja SOAR:n Azure Sentinel liitosohjelman (connector) pitäisi tukea?

- Incidentin kommentointi ja tilanvaihto yhdestä paikasta.
- Soarista pystyis käsittelemään hälyn, esim sulkemalla se soarista sulkee sen myös sentinelissä tai merkkaa FP yms.
- Alerttien hakeminen Azuresta, alerttien statusten muutto suoraan SOAR:sta (simppelien taskien automatisointi)
- Incidentien haku. Valittavina parametreina incidentin status (open,closed jne), luontiaika (esim. haetaan viimeisen 10min aikana tulleet tai vapaavalintainen staattinen aikaväli)
- jos edellä mainitun incident-listan mukana ei tule kaikkien yksittäisen incidentien kaikki data, niin toinen funktio, jolla haetaan yksittäisen incidentin kaikki data. Parametrinä ID.
- Incidentin päivitys (sulkeminen, jne). Connectorin valmiiksi ehdottamina fieldeinä ainakin status ja severity. Ehkä myös assigned henkilö?
- tärkeimpien connectoriin tulevien toimintojen API-kyselyiden parametrien vapaamuotoinen määritys.
- Mahdollisuuksien mukaan myös Sentinelin omien playbookien ajo.
- Vaihtoehto suorittaa haku eventeistä/lokeista? SOARista käyttäjän vapaasti määrittämä ja kirjoittama query.
- Mahdollisen kommentin/noten lisääminen (+ sen päivittäminen) incidentiin.

Mitä tietoja Azure Sentinel hälytyksestä pitää saada tuotua SOAR hälytykseen?

- Samanlaiset source target details tiedot kuin SIEMistä + Raaka loki tapahtuma jotta aina ei tarvitse avata Azurea.
- Sanoisin että vähän samaan tyyliin kun siemistä. Kohde IP, lähde IP, raakalokia, käyttäjä, lähdelaitteen host name, linkki suoraan siihen hälytykseen, jne.
- Suurinpiirtein Vastaavat datat mitä SIEM:stä saa ja mahdolliset muut datat jotka voivat hyödyttää analysointia // automaatiota SOAR:n päässä.
- Sentinelistä halutaan Incident-tietue, sekä siihen liittyvät alertit ja eventit. Kaikki palautettavat arvot voi mielestäni hakea APIsta, SOARin playbookeilla voidaan sitten valkata mitä tietoja käytetään ja mapataan SOAR-alertiin jne. Incidenteistä käytetään ainakin ne tavalliset: IDt, timestamp, hälyn title/description, severity, status, hälyyn liittyvät käyttäjät/laitteet/assetit ja itse alertin lähde (esim onko kyseessä jokin Sentinelin connectori?), yksittäisten eventien raakalokit, suorat linkit recordeihin Azuressa yms. Lisäksi voisi olla hyvä saada sieltä hälytyssäännön tiedot, joka kunkin hälyn on triggeröinyt.
- Jos tuon Azure Security Centerin ja Sentinelin väliltä löytyy APIsta haetusta datasta jotain korrelaatiota (jokin ID tms.) niin Security Centerin omia alerteja hakemalla saadaan ilmeisesti tarkempia tietoja yksittäisistä Sentinelin alerteista.

Liite 2. Liitosohjelmiston toiminnalliset vaatimukset

ID	Prioriteetti	Toiminnallisuus
FUNC_REQ_001	P1	Käyttäjän on voitava ajamaan olemassa olevia pelikirjoja, jotka käyttävät liitosohjelmistoa
FUNC_REQ_002	P1	SOARin on automaattisesti luotava hälytys jokaisesta uudesta Sentinelin tietoturvapoikkeamasta
FUNC_REQ_003	P1	Ylläpitäjän on voitava luomaan ja muokkaamaan pelikirjoja, jotka käyttävät liitosohjelmistoa
FUNC_REQ_004	P1	Ylläpitäjän on voitava lisätä liitosohjelmistolle konfiguraatioita eri asiakkaiden Sentinel ympäristöille
FUNC_REQ_005	P2	SOARin on automaattisesti haettava uuteen hälytykseen rikaste-tietoa Sentinelin tietoturvapoikkeamaan liittyen
FUNC_REQ_006	P2	Sentinelin tietoturvapoikkeamasta muodostuneen SOAR-hälytyksen tilan muutokset on päivityttävä Sentineliin
FUNC_REQ_007	P2	Käyttäjän lisäämät kommentit SOARin hälytykseen tulee saada päivittymään vastaavaan Sentinelin tietoturvapoikkeamaan
FUNC_REQ_008	P3	Käyttäjän on voitava käyttää SOARin pelikirjoja hyväkseen hälytyksen rikastetietojen hankinnan ja tutkimisen orkestroimiseen

Liite 3. Liitosohjelmiston ei-toiminnalliset vaatimukset

ID	Liittyvä toiminnallisuus	Vaatimus
PERF_REQ_001	FT001	Sentinelistä pitää saada haettua uudet tietoturvapoikkeamat 2 minuutin välein 99% toimintavarmuudella
PERF_REQ_002	FT001	Liitosohjelmiston tulee kyetä hakemaan vähintään sata uutta tietoturvapoikkeamaa Sentinelistä 2 minuutin välein
PERF_REQ_003	FT003	Liitosohjelmiston tulee kyetä kerralla muuttamaan vähintään sadan Sentinelin tietoturvapoikkeaman tilaa
PERF_REQ_004	FT002, FT003, FT005, FT007	Liitosohjelmiston manuaalisesti ajettavia pelikirjatoimintoja pitää pystyä ajamaan sekunnin välein 95% toimintavarmuudella
SEC_REQ_001	Kaikki	Jokainen liitosohjelmiston käyttökerta on tarkasteltavissa jälkikäteen
SEC_REQ_002	Kaikki	Liitosohjelmistoa käyttävien pelikirjojen on lokitettava ainakin toimenpiteen aiheuttaneen käyttäjän tiedot, kellonajan ja annetut parametrit
SEC_REQ_003	FT002, FT005	Kumpikaan järjestelmä ei saa käsitellä käyttäjän syötettä koodina tai komentoina
SEC_REQ_004	Kaikki	SOARin ja Azure Sentinel REST APIn välinen liikenne tulee olla salattua
SEC_REQ_005	Kaikki	SOARin ja Azure Sentinel REST APIn välisen liikenteen eheydestä tulee olla varmuus
USAB_REQ_001	FT002	SOARin tietoturvapoikkeaman kautta lisättävät kommentit näkyvät oikein Sentinelissä, vaikka ne sisältäisivät erikoismerkkejä

Liite 4. Vaatimusmäärittelyssä esitettyjen toiminnallisuuksien toteutuminen

ID	Toteutettu?	Kommentti
FT001	Kyllä	Uudet Sentinelin tietoturvapoikkeamat muodostuvat automaattisesti SOARiin hälytyksiksi kahden minuutin välein
FT002	Kyllä	SOARin hälytykseen lisätyn kommentin saa lisättyä vastaavaan Sentinelin tietoturvapoikkeamaan SOARin painikkeen avulla
FT003	Kyllä	SOARin hälytyksen tilan muutokset synkronoituvat vastaavaan Sentinelin tietoturvapoikkeamaan
FT004	Ei	Microsoftin Azuren ohjelmointirajapintojen avulla ei saatu haettua tiettyyn tietoturvapoikkeamaan liittyviä hälytyksiä
FT005	Kyllä	Liitosohjelmistoon tehtiin pelikirjatoiminnallisuus, joka mahdollistaa mielivaltaisten kyselyiden tekemisen Azure Sentinel REST APIin
FT006	Ei	Sentinelin tietoturvapoikkeamien automatisointi päätettiin tehdä Sentinelin pelikirjoilla, joten toiminnallisuutta ei tarvita liitosohjelmistoon
FT007	Ei	Microsoftin Azuren ohjelmointirajapintojen avulla ei saada ajettua olemassa olevia Sentinelin pelikirjoja

Liite 5. Liitosohjelmiston toiminnallisten vaatimusten toteutuminen ja toteutetut testit

ID	Toteutettu?	Kommentti
FUNC_REQ_001	Kyllä	Integraation pelikirjojen testaaminen toteutettiin tavallisena käyttäjänä SOARissa
FUNC_REQ_002	Kyllä	Sentineliin luotiin kappaleessa 5.2 erilaisia tietoturvapoikkeamia, jotka muodostuivat automaattisesti SOARiin
FUNC_REQ_003	Kyllä	Integraatiossa käytettävät pelikirjat on luotu käyttäjällä, jolla on ylläpitäjän oikeudet
FUNC_REQ_004	Kyllä	Liitosohjelman testauksessa käytettyjen Sentinel-ympäristöjen konfiguraatiot lisättiin ylläpitäjän käyttöoikeuksilla.
FUNC_REQ_005	Ei	Microsoftin Azuren ohjelmointirajapintojen avulla ei saatu haettua tiettyyn tietoturvapoikkeamaan liittyviä hälytyksiä
FUNC_REQ_006	Kyllä	Kappaleessa 5.3 sadan hälytyksen tila muutettiin yhtäaikaisesti SOARissa ja kaikki muutokset synkronoituivat Sentineliin
FUNC_REQ_007	Kyllä	Kappaleessa 0 SOARin hälytykseen luotiin erilaisia kommentteja, jotka päivittyivät Sentinelin tietoturvapoikkeamaan
FUNC_REQ_008	Ei	Microsoftin Azuren ohjelmointirajapintojen avulla ei saatu haettua tiettyyn tietoturvapoikkeamaan liittyviä hälytyksiä

Liite 6. API-kysely – Tietoturvapoikkeamien tietojen hakeminen

GET https://management.azure.com/subscriptions/:subscriptionId/resourceGroups/:resource-Group/providers/Microsoft.OperationalInsights/workspaces/:workspaceName/providers/Microsoft.SecurityInsights/incidents?api-version=2020-01-01

JSON Rivi	**Rivin arvon sisältö**
id	APIn palauttaman tietueen ID
name	Tietoturvapoikkeaman ID
type	Microsoft.SecurityInsights/Incidents
etag	Tietoturvapoikkeaman version yksilöivä tieto
properties/title	Otsikko
properties/description	Kuvaus
properties/severity	Vakavuus
properties/status	Tila
properties/classification	Oikea-, väärä- vai haitaton hälytys
properties/classificationReason	Peruste *classification* rivin arvolle
properties/classificationComment	Kommentti miksi tietoturvapoikkeama suljettiin
properties/owner	Tietoturvapoikkeaman omistajan tiedot
properties/owner/objectId	Omistajan ID
properties/owner/email	Tietoturvapoikkeaman omistajan sähköpostiosoite
properties/owner/assignedTo	Tietoturvapoikkeaman omistajan nimi
properties/owner/userPrincipalName	Tietoturvapoikkeaman omistajan käyttäjänimi
properties/labels	Tags-objektien taulukko
properties/labels/labelName	Tag-objektin nimi
properties/labels/labelType	Tag-objektin tyyppi, esimerkiksi User
properties/firstActivityTimeUtc	Ensimmäisen liittyvän hälytyksen aikaleima
properties/lastActivityTimeUtc	Viimeisen liittyvän hälytyksen aikaleima
properties/lastModifiedTimeUtc	Viimeisimän muokkauksen aikaleima
properties/createdTimeUtc	Tietoturvapoikkeaman luomisen aikaleima
properties/incidentNumber	Tietoturvapoikkeaman numero
properties/additionalData	Taulukko tietoturvapoikkeaman lisätiedoista
properties/additionalData/alertsCount	Liittyvien hälytysten lukumäärä
properties/additionalData/bookmarksCount	Liittyvien kirjanmerkkien lukumäärä
properties/additionalData/commentsCount	Liittyvien kommenttien lukumäärä
properties/additionalData/alertProductNames	Taulukko liittyvistä tuotteista
properties/additionalData/tactics	Taulukko liittyvistä taktiikoista
properties/relatedAnalyticRuleIds	Taulukko liittyvistä analytiikkasäännöistä
properties/incidentUrl	Linkki Sentinelin tietoturvapoikkeamaan

Liite 7. Koodi – Eri kriittisyystason poikkeamien luominen testausta varten

```
# Returns bearer token for Azure REST API from Microsoft OAuth2
def get_bearer_token(tenant, client_id, client_secret):
    endpoint = 'https://login.microsoftonline.com/' + tenant + '/oauth2/token'
    post_data = {
            'grant_type'    : 'client_credentials',
            'client_id'     : client_id,
            'client_secret' : client_secret,
            'resource'      : 'https://management.azure.com/'
            }
    return requests.post(endpoint, data=post_data).json() ['access_token']

# Azure Sentinel REST APIs URL
azure_rest_api_endpoint = 'https://management.azure.com' \
    + '/subscriptions/' + subscription_id \
    + '/resourceGroups/' + resource_group \
    + '/providers/Microsoft.OperationalInsights' \
    + '/workspaces/' + workspace_name \
    + '/providers/Microsoft.SecurityInsights/Incidents/'

bearer_token = get_bearer_token(tenant, client_id, client_secret)
severities = ['Informational', 'Low', 'Medium', 'High']

# Create 4 incidents in total with different severities
for i in range(4):
    incident_id     = str(random.randint(1, 133333337))   # Create random ID for incident
    api_url         = azure_rest_api_endpoint + incident_id
    incident_name   = 'TEST Incident: ' + severities[i]
    headers = { 'Content-Type':     'application/json',
            'Authorization':    f'Bearer {bearer_token}' }
    payload = { 'api-version': '2020-01-01' }

    request_body = {
              'properties': {
                  'description': 'This is a demo incident',
                  'severity': severities[i],
                  'status': 'New',
                  'title': incident_name
                  }
              }
    # Create incident to Azure Sentinel
    response = requests.request(method='PUT', url=api_url, headers=headers, params=payload, json=request_body)
    if response.status_code == 200:
        print('Created incident', incident_name)
    else:
        print('Failed to create incident:', response.content)
        exit(1)
```