VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

INTEGRACIJA PODATKOV O GROŽNJAH V KIBERNETSKI DOMENI S SIEM REŠITVIJO

Kandidat: Žan Rotar

Vrsta študija: študent izrednega študija

Študijski program: informatika

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: mag. Nataša Klenovšek Arh

Lektor: mag. Ana Žagar, prof. slov. jez. in knj.

Maribor, 2023

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Žan Rotar, sem avtor diplomskega dela z naslovom »Integracija podatkov o grožnjah v kibernetski domeni s SIEM rešitvijo«, ki sem ga napisal pod mentorstvom mag. Ervina Schaffa.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Kompolje, september 2023 Podpis študenta:

ZAHVALA

Rad bi se zahvalil mentorici v podjetju Nataši Klenovšek Arh za strokovno svetovanje, potrpežljivost in spodbudo pri nastajanju diplomskega dela.

Zahvalil bi se tudi punci Jani Božič, ki mi je vedno stala ob strani in me spodbujala, ter staršem za vso podporo in finančno pomoč pri študiju.

POVZETEK

Kibernetske grožnje postajajo vse bolj kompleksne, kar me je spodbudilo, da se poglobim v razvoj celovite rešitve za njihovo odkrivanje in obvladovanje. V tem smislu sem samostojno prepoznal, da bi platforma Open Cyber Threat Intelligence (OpenCTI) lahko odigrala ključno vlogo pri združevanju, obogatitvi in izmenjavi obveščevalnih podatkov o grožnjah za izboljšanje varnosti naše organizacije. Izziv, ki sem ga sprejel, je bil precej zahteven, saj sem se srečeval z integracijo OpenCTI, s sistemom za upravljanje varnostnih informacij in dogodkov (SIEM), kar je lahko oviralo nemoteno izmenjavo obveščevalnih podatkov.

V okviru diplomske naloge sem se zato odločil, da se bolj poglobim v to problematiko in izvedem raziskavo ter razvoj priključka za OpenCTI, ki temelji na programskem jeziku Python. Moj cilj je bil izboljšati izmenjavo obveščevalnih podatkov o grožnjah in povečati njihovo uporabo znotraj naše obstoječe varnostne infrastrukture. Skozi ta proces sem se odločil slediti sistematičnemu pristopu, ki je vključeval naslednje ključne korake: Analiza zahtev; V sodelovanju s strokovnjaki za kibernetsko varnost in skrbniki SIEM so bile opredeljene potrebe in pričakovanja za integracijo OpenCTI-SIEM, Oblikovanje in arhitektura; Na podlagi opredeljenih zahtev je bila zasnovana dobro strukturirana arhitektura za priključek na osnovi Python-a, Izvedba; Povezovalnik je bil skrbno kodiran v jeziku Python, pri čemer je bila uporabljena moč ustreznih knjižnic in upoštevane najboljše prakse pri razvoju programske opreme, Testiranje in potrjevanje; Uporabljene so bile stroge metodologije testiranja, da bi preverili natančnost, zmogljivost in skladnost priključka z opredeljenimi zahtevami, Dokumentacija in uporabniški priročnik; Za lažjo uvedbo in uporabo priključka v našem podjetju v prihodnosti je bil izdelan obsežen uporabniški priročnik.

V okviru diplomske naloge je bil uspešno razvit po meri narejen priključek OpenCTI, ki temelji na Python-u, za nemoteno izmenjavo podatkov med OpenCTI in sistemom SIEM naše organizacije. Ta integracija omogoča uporabo obogatenih podatkov o grožnjah za boljše odkrivanje v realnem času, proaktivne ukrepe kibernetske varnosti in povečanje odpornosti organizacije proti grožnjam. Poleg tega smo z implementacijo tega priključka omogočili tudi avtomatizacijo procesov obveščanja in odzivanja na kibernetske incidente, kar izboljšuje učinkovitost obrambe in zaščito občutljivih informacij ter infrastrukture.

Ključne besede: *kibernetska varnost, OpenCTI, raziskovanje in razvoj, programiranje, Python*

ABSTRACT

Integrating cyber threat data with a SIEM solution

Cyber threats are becoming increasingly complex, which has motivated me to delve deeper into developing a comprehensive solution for their detection and management. In this regard, I independently recognized that the Open Cyber Threat Intelligence (OpenCTI) platform could play a crucial role in aggregating, enriching, and sharing threat intelligence data to enhance the security of our organization. The challenge I undertook was quite demanding, as I faced the task of integrating OpenCTI with our Security Information and Event Management (SIEM) system, which could potentially hinder the seamless exchange of threat intelligence data.

Within the scope of my thesis, I therefore decided to delve further into this issue and conduct research and development of a Python-based connector for OpenCTI. My goal was to improve the exchange of threat intelligence data and enhance their utilization within our existing security infrastructure. Throughout this process, I chose to follow a systematic approach, which included the following key steps: Requirements Analysis; Collaborating with cybersecurity experts and SIEM administrators to identify the needs and expectations for the OpenCTI-SIEM integration, Design and Architecture; Based on the identified requirements, a well-structured architecture for the Python-based connector was designed, Implementation; The connector was carefully coded in Python, leveraging the power of appropriate libraries and adhering to best practices in software development, Testing and Validation; Stringent testing methodologies were employed to verify the accuracy, performance, and compliance of the connector with defined requirements, Documentation and User Manual; An extensive user manual was created to facilitate the introduction and use of the connector within our organization in the future.

Within the scope of the thesis, a custom-made Python-based OpenCTI connector was successfully developed to facilitate seamless data exchange between OpenCTI and our organization's SIEM system. This integration enables the use of enriched threat data for real-time threat detection, proactive cybersecurity measures, and enhancing our organization's resilience against evolving threats. Additionally, the implementation of this connector has enabled the automation of processes for alerting and responding to cyber incidents, improving defense efficiency and the protection of sensitive information and infrastructure.

Keywords: *cybersecurity, OpenCTI, research and development, programming, Python*

KAZALO VSEBINE

1 UVOD 6

1.1 Opis področja in opredelitev problema 6

1.2 Namen, cilji in osnovne trditve 7

1.3 Predpostavke in omejitve 7

1.4 Uporabljene raziskovalne metode 7

2 PREDSTAVITEV PLATFORME SIEM 9

2.1 Premerjava SIEM rešitev iz vidika majhnega podjetja 9

3 KAJ SO PODATKI O GROŽNJAH V KIBERNETSKI DOMENI (KROVNA TEMA) 11

3.1 Big data 12

3.2 Big Data in OpenCTI 12

3.3 Namestitev OpenCTI in predstavitev delovanja 14

3.4 Testiranje zmogljivosti OpenCTI 16

4 PODROBNEJŠA PREDSTAVITEV PROBLEMA 18

5 INTEGRACIJA PODATKOV O GROŽNJAH S SIEM REŠITVIJO 20

6 RAZVOJ MODULA ZA INTEGRACIJO 21

6.1 Uporabljena orodja 21

7 OPIS DELOVANJA MODULA 24

7.1 Uporabljene knjižnice 24

7.2 Konfiguracijska datoteka 24

7.3 Branje podatkov z OpenCTI virov 25

7.3.1 Razred OpenCTIFeed 27

7.3.2 Predstavitev funkcij razreda OpenCTIFeed 27

7.4 Komunikacije s SIEM 30

7.4.1 Razred SIEMAPICommunication 30

7.4.2 Funkcija za izvajanje iskalnih ukazov 31

7.4.3 Razčlenjevanje podatkov in ustvarjanje varnostne kopije 32

7.4.4 Primerjava podatkov in zapis v SIEM 34

7.4.5 Funkcija za zapis podatkov v iskalno tabelo 35

7.5 Pomožna razreda 35

7.5.1 Pomožni razred za beleženje delovanja modula 36

7.5.2 Pomožni razred za izključitev vrednosti ... 37
7.6 ZDRUŽITEV KODE V CELOVITO DELOVANJE ... 38
7.7 PREMIK V PRODUKCIJSKO OKOLJE ... 40
7.8 IZBOLJŠAVE OBSTOJEČE REŠITVE ... 41
8 SKLEP ... 43
9 VIRI ... 45
10 SEZNAM UPORABLJENIH KRATIC ... 46

KAZALO SLIK

SLIKA 1 : KONFIGURACIJA SPLETNEGA STREŽNIKA NGNINX ... 16
SLIKA 2: AVTOMATIČNO GENERIRAN KOMENTAR FUNKCIJE ... 22
SLIKA 3: DATOTEKA ".ENV.EXAMPLE" ... 25
SLIKA 4: IZPIS PODATKOV OPENCTI NA VIR ... 26
SLIKA 5: OPENCTIFEED KONSTRUKTOR ... 27
SLIKA 6: GLAVNA FUNKCIJA RAZREDA OPENCTI, SKRAJŠANA ... 28
SLIKA 7: FUNKCIJA ZA IZBRIS PODVOJENIH PODATKOV – REMOVEDUPES ... 29
SLIKA 8: FUNCKIJA ZA PRAVILNO NASTAVITEV DATUMOV VELJAVNOSTI – CHECKDATES ... 29
SLIKA 9: FUNKCIJA ZA VZPOSTAVITEV POVEZAVE S SIEM ... 31
SLIKA 10: FUNKCIJA ZA IZVAJANJE ISKALNIH UKAZOV ... 32
SLIKA 11: DEL FUNKCIJE ZA RAZČLENJEVANJE PODATKOV ... 33
SLIKA 12: FUNKCIJA ZA USTVARJANJE VARNOSTNIH KOPIJ ... 34
SLIKA 13: IZPIS ŠTEVILA VPISOV ISKALNE TABELE ... 35
SLIKA 14: KLIC STATIČNE FUNKCIJE ... 36
SLIKA 15: FUNKCIJA ZA ISKANJE DATOTEK NA DOLOČENI POTI ... 36
SLIKA 16: FUNKCIJA ZA ZAPISOVANJE BESEDILA V DNEVNIŠKO DATOTEKO ... 37
SLIKA 17: RAZRED ZA BRANJE IZKLJUČENIH VREDNOSTI ... 37
SLIKA 18: POVEZAVA S SIEM V GLAVNI SKRIPTI ... 38
SLIKA 19: POVEZAVA Z OPENCTI IN PRIMERJAVA PODATKOV ... 39
SLIKA 20: ZAPIS PODATKOV IN ZAKLJUČEK GLAVNE SKRIPTE ... 40

KAZALO TABEL

TABELA 1: PRIMERJAVA MOŽNOSTI MALEGA PODJETJA ... 10
TABELA 2 : PORABA SREDSTEV OPENCTI GLEDE NA KOLIČINO INDIKATORJEV ... 17

KAZALO GRAFOV

GRAF 1 : PORABA SREDSTEV OPENCTI GLEDE NA KOLIČINO PODATKOV ... 17

1 UVOD

V hitro razvijajočem se okolju današnjega tehnološkega napredka so vse pogostejše razprave o digitalizaciji, digitalni preobrazbi in prenosu običajnih storitev na informacijske in komunikacijske platforme. Ob tem premiku je treba upoštevati hkratno pojavljanje novih groženj na digitalnem področju. Medtem ko lahko fizični in tehnični varnostni ukrepi varujejo otipljive strukture, je zaščita neoprijemljivih entitet, kot so podatkovne zbirke, omrežja in informacijski sistemi, poseben izziv. V fazi namenskega raziskovanja sem se najprej srečal s konceptom sistema za upravljanje varnostnih informacij in dogodkov (angl. Security Information and Event Management - SIEM) - prefinjenim okvirom, namenjenim zbiranju in analiziranju podatkov, povezanih z grožnjami na kibernetskem področju, ki zapolnjuje vrzel med običajnimi varnostnimi praksami in zmanjševanjem digitalnih ranljivosti. Ta prizadevanja za utrditev postanejo še toliko bolj ključna, če upoštevamo večplastne tokove podatkov, ki jih obdelujejo ti medsebojno povezani sistemi, kar poudarja potrebo po zanesljivih obrambnih strategijah v vse bolj povezanem digitalnem svetu.

1.1 Opis področja in opredelitev problema

Osrednji cilj moje diplomske naloge je komajda prikrit, saj ga je mogoče razbrati že iz njenega naslova. Poglabljanje v obsežno področje kibernetske varnosti, za katero so značilne velike razsežnosti in zapleteni odtenki, predstavlja velik podvig. Vendar sem si med širino in zapletenostjo tega področja zadal nalogo, da se osredotočim na točno določeno težavo v našem sistemu. Ta namerna zožitev fokusa ni posledica pomanjkanja spoštovanja do širine predmeta, temveč priznanje, da je mogoče razčleniti in rešiti poseben izziv, ki je pomemben v kontekstu naše tehnološke pokrajine. V bistvu moje prizadevanje ne želi zajeti celotne kibernetske varnosti, temveč prispevati pomemben prispevek z razvozlavanjem posebne uganke, katere posledice segajo daleč prek njenih neposrednih meja.

Opis problema: Kibernetska varnost temelji na zaznavi in uspešni zavrnitvi potencialnih zlonamernih aktivnosti, bodisi na naši spletni aplikaciji, omrežju ali v naših elektronskih sporočilih. Temu se reče pasivna kibernetska varnost, saj samo nastavimo svojo zaščito in jo spremljamo za potencialne sumljive aktivnosti oziroma vdore. Na drugi strani kovanca imamo proaktivno kibernetsko varnost, ki poleg same zaščite vsebuje tudi zbiranje podatkov o zlonamernih dejavnostih in deljenje le-teh s skupnostjo. Problem leži v integraciji podatkov o potencialnih grožnjah s spremljanjem dogajanja v našem informacijskem sistemu.

1.2 Namen, cilji in osnovne trditve

Namen diplomske naloge je raziskava obstoječih rešitev predstavljenega problema oziroma razvoj samostojnega modula, ki bi zadostoval našim potrebam v podjetju.

Glavni cilj diplomske naloge je avtomatski prenos podatkov iz sistema OpenCTI v naš SIEM, kjer bi se le-ti lahko korelirali z dnevniškimi zapisi prometa našega omrežja. Sekundarni cilji so boljše spoznavanje z zaledjem SIEM in rešitvijo OpenCTI, kar bi omogočilo boljše opravljanje dela na mojem delovnem mestu in morebiten razvoj modula, saj sem zelo entuziastičen kar se tiče razvoja novih rešitev.

Osnovne trditve diplomske naloge:

- H1: Ali lahko rečemo, da je Security Onion najboljša SIEM rešitev?
- H2: OpenCTI lahko prenese velike količine podatkov o grožnjah in jih smiselno korelira med seboj.
- H3: Povezava SIEM rešitve z OpenCTI že obstaja.
- H4: Obstoječo oziroma razvito rešitev lahko predelamo tako, da bo podpirala neskončen tok podatkov (krovna tema).

1.3 Predpostavke in omejitve

Iskanje rešitve bom omejil na literaturo in spletne vire, in sicer pregled že razvitih tako imenovanih priključkov (angl. connector), ki se nahajajo v Github repozitoriju in raziskavo, ali se morebitna rešitev nahaja kje drugje na uradnih spletnih straneh OpenCTI. Izognil se bom spraševanju skupnosti za tovrstno rešitev, saj v kolikor ne obstaja, želim le-to razviti sam. Za to diplomsko nalogo je pomembno le, da skripta deluje in ni pomembno, da je razvita rešitev optimizirana.

1.4 Uporabljene raziskovalne metode

Diplomska naloga je sestavljena iz preučevanja obstoječih rešitev oziroma iskanja le-teh, branja dokumentacije in samega razvoja modula oziroma rešitve problema. Pri razvoju sem velikokrat prebiral dokumentacijo in med možnimi rešitvami dotičnega problema s procesom eliminacije in testiranja izločal rešitve, ki so bile za moj problem neuporabne.

Uporabljene raziskovalne metode:

- pregled literature (branje dokumentacije),
- opazovanje (iskanje rešitve za problem, primerjanje),
- prototipiranje (preizkušanje različnih možnih rešitev),
- eksperimentiranje (preizkušanje različnih možnih rešitev).

2 PREDSTAVITEV PLATFORME SIEM

SIEM zbira podatke z različnih naprav v informacijskem sistemu, ustvarja povezave med podatki, ki jih dobi s teh naprav, in na podlagi dobljenih rezultatov obvešča o varnostnih incidentih. Prejeti podatki so v sistemu indeksirani, kar omogoča hitro iskanje in odzivnost iskalne vrstice. Sistem nam lahko služi za zaznavo groženj, vendar za to potrebujemo podatke o le-teh. Samo detekcijo groženj lahko s pomočjo vgrajenih orodij avtomatiziramo in prilagodimo osebnim potrebam. (Microsoft, 2023)

Varnostni operativni center (angl. Securtiy Operations Center - SOC) je temelj digitalne varnosti, saj upravlja prefinjen sklop tehnologij in metodologij za zaščito celovitosti sodobnih informacijskih sistemov. Njegov temeljni cilj je budno spremljanje, odkrivanje in odzivanje na raznovrstne potencialne varnostne grožnje in incidente.

Jedro SOC sestavlja večplastna infrastruktura, opremljena s SIEM sistemom, ki služi kot analitični živčni center. Ta sistem SIEM je zasnovan tako, da združuje, povezuje in analizira obsežno paleto podatkov, ki izhajajo iz različnih virov v ekosistemu IT organizacije. Ti viri vključujejo aplikacije, omrežne naprave, kot so požarni zidovi in usmerjevalniki, strežnike in pomembne podatkovne shrambe.

Proaktivna vloga SOC presega upravljanje incidentov v realnem času. Vključuje zgodovinsko perspektivo, ki omogoča sledenje in analizo preteklih varnostnih dogodkov in trendov. To časovno razumevanje je nujno za prepoznavanje ponavljajočih se vzorcev in razvijajočih se področij groženj, kar na koncu vodi k oblikovanju strateških varnostnih ukrepov.

2.1 Premerjava SIEM rešitev iz vidika majhnega podjetja

Predpostavimo, da smo manjše podjetje, ki ima spletno aplikacijo, prek katere podjetje ponuja svoje storitve širšemu svetu. Ker opažajo oziroma so doživeli kibernetski napad, začenjajo razumevati, kako pomembna je kibernetska varnost in krepitev le-te. Za krepitev varnosti so si iz finančnih sredstev rezervirali določeno količino denarja, ki bi ga radi čim bolj efektivno uporabili. Zavedajo se, da je spremljanje omrežnega prometa ključno za kibernetsko varnost, zato želijo postaviti SIEM, ki bo prejemal dnevniške zapise iz različnih strežnikov, na katerih imajo postavljene spletne storitve.

Po iskanju po spletu podjetje ugotovi, da imajo nekaj možnih rešitev tovrstnega problema. Lahko se odločijo:

A) da nalogo SIEM-a podajo zunanjim izvajalcem, kar pomeni, da podjetje ne potrebuje novega kadra, ki bi se ukvarjalo s tem.

B) da postavijo SIEM na lokaciji, kjer se znajdejo v dilemi, saj je večina tovrstnih rešitev plačljivih in niso kompatibilni z njihovim proračunom. Zastonjske rešitve, kot na primer Security Onion so vse zgrajene iz podobnega ogrodja, in sicer ELK Stack-a. (Security Onion Solutions, 2023) To je odprtokodna rešitev za shranjevanje in upravljanje z dnevniškimi zapisi, ki jo večina različnih SIEM rešitev uporabi za zaledje in zgradi samo vizualizacijo podatkov. Ker je tovrstna rešitev odprtokodna, le-to za seboj po navadi prinese časovni vložek, da bo delovala in prinesla doprinos podjetju.

Za potrebe majhnega podjetja, kjer je prioriteta zagotavljanje splošne oziroma minimalne kibernetske varnosti za njihove storitve, bi bilo smiselno izbrati možnost zunanjih izvajalcev kibernetske varnosti. S to odločitvijo moramo sprejeti tudi slabosti le-te, saj to pomeni, da moramo vse podatke oziroma dnevniške zapise pošiljati izven našega omrežja, kar bi lahko bilo sporno v primeru, da je v le-teh kaj kar želi podjetje prikriti, pa naj bodo to poslovne skrivnosti ali pa kdo je pri njih zaposlen.

Tabela 1: Primerjava možnosti malega podjetja

Rešitev	**Prednosti**	**Slabosti**	**Stroški**
Postavitev plačljive SIEM rešitve (IBM QRadar)	Zelo prepoznaven in dolgoživ SIEM sistem, kar pomeni veliko dokumentacije, in sicer olajša delo vsem, ki bi se z njim ukvarjali.	Poleg cene, potrebujemo tudi usposobljene zaposlene, ki bodo skrbeli za sistem in spremljali dogajanje na omrežju.	Zelo odvisna od zakupljene licence, ki se meri glede na število naprav v omrežju (vsaj 297 €/mesec). (IBM QRadar, 2023)
Postavitev odprtokodne SIEM rešitve (Security Onion)	Odprtokodna SIEM rešitev	Usposobljeni zaposleni, ki bodo skrbeli za sistem in spremljali dogajanje na omrežju.	Brezplačno
Prepustitev kibernetske varnosti zunanjih izvajalcev	Minimalen vpliv na podjetje (razen vzpostavitev posredovanja dnevniških zapisov).	Pošiljanje podatkov o dogajanju na omrežju izven le-tega je lahko sporno.	1.22 € na mesec (za navadne naročnike, za podjetje cenika nisem našel). (Telekom Slovenija, 2023)

Vir: (Lasten vir)

3 KAJ SO PODATKI O GROŽNJAH V KIBERNETSKI DOMENI (KROVNA TEMA)?

Podatki o grožnjah v kibernetski domeni (angl. Cyber Threat Intelligence - CTI) so znanje, veščine in informacije, ki so bazirane na izkušnjah, o grožnjah v kibernetski ali fizični domeni in zlonamernih akterjih. Zbiramo jih z namenom, da nam pomagajo olajšati oziroma izničiti potencialne napade in škodljive dogodke v kibernetski domeni.

Viri podatkov niso omejeni le na interne, vendar lahko le-te najdemo tudi na spletu in iz njih črpamo ter agregiramo podatke, na primer pridobivanje informacij iz javnih virov (angl. Open-Source Intelligence - OSINT), kot so mediji, javni uradni podatki (javni proračuni, telefonski imeniki, spletne strani), akademske publikacije in javno neobjavljeni dokumenti. Našteli smo primere zunanjih virov, primeri notranjih so tehnična znanja ali veščine, dnevniški zapisi, forenzično odkriti podatki, obveščevalne informacije o prometu na omrežju (primer takih podatkov so zaznave na našem SIEM sistemu) in podatki, izvlečeni iz globokega ali temnega spleta.

Podatke o grožnjah delimo na štiri tipe:

- Strateški: informacije o tveganjih napadov v kibernetski domeni, ki jih lahko uporabimo za sprejemanje odločitev glede organizacijske strategije.
- Taktični: informacije o vektorjih napada in šibkosti našega sistema, ki bi jih lahko napadalci izkoristili.
- Tehnični: fokusirano na namige in dokaze napada, pod kar spadajo IP naslovi, besedila sporočil spletnega ribarjenja, primeri zlonamerne kode in URL naslovi.
- Operativni: podrobnosti o ciljih in sposobnostih zlonamernih akterjev, kar vključuje orodja, tehnike in procedure. (CrowdStrike, 2023)

Zbiranje podatkov je zelo koristno, oziroma nujno za vse organizacije, ki same skrbijo za svojo kibernetsko varnost in ne posegajo po zunanjih virih, torej ne najemajo IT podjetij, da prevzamejo breme kibernetske varnosti. To je značilno za srednje velika in velika podjetja oziroma organizacije, ki imajo dovolj virov in kadra, da obvladujejo svoje potrebe po kibernetski varnosti. Zbrani podatki nam pomagajo pri razumevanju napadalcev, njihovih namenov, ciljev in procedur. Analiza zbranih podatkov o zlonamernih akterjih nam omogoča predvidevanja o njihovem vedenju in stopnji znanja ter tehnični opremljenosti.

3.1 Big data

Big Data podatki se nanašajo na izjemno velike in zapletene slope podatkov, ki presegajo zmogljivosti obdelave tradicionalnih orodij za upravljanje s podatki in podatkovnih zbirk. Kategoriziramo jih lahko s pomočjo treh tipičnih karakteristik:

1. Volumen; Big Data vključujejo ogromne količine podatkov, običajno od terabajta naprej. Podatki lahko izvirajo iz različnih virov, vključno z družbenimi mediji, senzorji, fizičnimi transakcijami ali drugimi mehanizmi za ustvarjanje podatkov.
2. Hitrost; Podatki se ustvarjajo, zbirajo in obdelujejo z neprimerljivo hitrostjo. S pojavom interneta in podatkovnih tokov v realnem času se je povečala potreba po hitri obdelavi za pridobivanje dragocenih vpogledov in takojšnje odzivanje na dogodke.
3. Raznolikost; Big Data se pojavljajo v številnih oblikah in vrstah. Vključujejo strukturirane podatke (npr. podatkovne zbirke), polstrukturirane podatke (npr. XML, JSON) in nestrukturirane podatke (npr. besedilo, slike, videoposnetki). Učinkovito upravljanje in analiziranje tako raznolikih virov podatkov predstavlja velik izziv.

Vendar z veliko količino podatkov pridejo tudi nekateri pomisleki, ki jih je potrebno upoštevati. Preverjanje zanesljivosti in točnosti Big Data, je zaradi njihovih različnih virov in formatov ključnega pomena. Zagotavljanje kakovosti le-teh je bistvenega pomena za sprejemanje utemeljenih odločitev na podlagi pridobljenih vpogledov. Drugi pomislek je vrednost tovrstnih podatkov, saj je glavni cilj analize Big Data pridobiti dragocene vpoglede in znanje, ki lahko spodbudijo premišljene poslovne odločitve, optimizacijo procesov, izboljšave izdelkov in storitev ter splošno izboljšanje učinkovitosti.

Za učinkovito obdelavo velikih količin podatkov se tradicionalne metode obdelave podatkov izkažejo za nezadostne. Zato so bile razvite specializirane tehnologije, kot so Apache Hadoop, Apache Spark in podatkovne zbirke NoSQL. Le-ta ogrodja porazdeljenega računalništva organizacijam omogočajo učinkovito shranjevanje, obdelavo in analizo obsežnih podatkov.

3.2 Big Data in OpenCTI

OpenCTI je odprtokodna platforma, namenjena zbiranju podatkov o kibernetskih grožnjah in koreliranju le-teh. Podatke hrani v podatkovni bazi Elasticsearch-a zgrajeni s pomočjo Apache Lucene, ki je namenjena za hitro upravljanje z velikimi količinami podatkov. Apache Lucene je visoko zmogljiva, polno funkcionalna knjižnica za iskanje besedil, napisana v jeziku Java.

Pogosto se uporablja za vgradnjo iskalnih zmogljivosti v aplikacije, spletna mesta in druge programske sisteme. OpenCTI jo uporablja predvsem za:

- Iskanje in poizvedovanje: OpenCTI uporablja Apache Lucene za iskanje in poizvedovanje znotraj platforme. Zmožnosti indeksiranja in iskanja Lucene bi lahko uporabnikom omogočilo hitro in natančno iskanje po podatkih o grožnjah, shranjenih v OpenCTI. To vključuje iskanje kazalnikov, akterjev groženj, kampanj in drugih povezanih informacij.
- Indeksiranje celotnega besedila: Apache Lucene je znan po svojih učinkovitih zmogljivostih indeksiranja in iskanja celotnega besedila. OpenCTI uporablja Lucene za ustvarjanje indeksa besedilne vsebine poročil o grožnjah, opomb, komentarjev in drugih pomembnih informacij. Le-to uporabnikom omogoča iskanje določenih ključnih besed ali besednih zvez v teh dokumentih.
- Funkcije naprednega iskanja: Lucene podpira napredne funkcije iskanja, kot so poizvedbe z nadomestnimi znaki, nejasno iskanje, iskanje po bližnjici in drugo. OpenCTI uporablja te funkcije, da bi uporabnikom ponudil zmogljive in prilagodljive možnosti iskanja pri iskanju podatkov o grožnjah.
- Indeksiranje v realnem času: Lucene podpira indeksiranje v realnem času, kar pomeni, da se lahko novi podatki o grožnjah, dodani v OpenCTI, indeksirajo in ponudijo za iskanje skoraj takoj. To je ključnega pomena za ohranjanje posodobljenih rezultatov iskanja in pravočasno zagotavljanje informacij uporabnikom.
- Skalabilnost in zmogljivost: OpenCTI uporablja porazdeljene zmožnosti Lucene za doseganje razširljivosti in izboljšane zmogljivosti. Lucene je mogoče konfigurirati za porazdeljeno delovanje, kar omogoča učinkovito indeksiranje in iskanje velikih količin podatkov o grožnjah.
- Prilagajanje: Apache Lucene je zelo prilagodljiv in ga je mogoče prilagoditi posebnim zahtevam. OpenCTI prilagodi indeksiranje in iskanje tako, da ustreza edinstvenim potrebam področja obveščanja o grožnjah.

OpenCTI ima zanj razvite tudi odprtokodne priključke, ki so javno dostopni in tako na razpolago vsem, ki jih želijo uporabljati. Priključki so razširitve obstoječega sistema in nudijo nadgradnjo iz vidika možnosti uvažanja ali izvažanja podatkov. Podatki so zato lahko drugačni od slik, dokumentov, datotek, imena domen ali IP naslovov.

Ali lahko OpenCTI označimo kot Big Data, je odvisno od načina uporabe in obsega podatkov, ki jih obdeluje. Big Data se običajno nanaša na nabore podatkov, ki so veliki, kompleksni in jih je težko obdelati s tradicionalnimi tehnikami upravljanja in obdelave podatkov. Ker se OpenCTI uporablja za upravljanje in analizo obsežnih podatkov o kibernetski varnosti, ga je mogoče obravnavati kot del infrastrukture za Big Data.

3.3 Namestitev OpenCTI in predstavitev delovanja

OpenCTI ima dve možnosti namestitve, ki sta ločeni po načinu namestitve, in sicer ga lahko namestimo s pomočjo okolja za ustvarjanje kontejnerjev docker ali ga postavimo ročno. Razlika je v tem, da pri docker namestitvi programske opreme ne namestimo direktno na strežnik oziroma okolje, kjer ga nameščamo, temveč se vsa programska oprema in njegove zahteve zložijo v virtualizirana izvajalna okolja, ki jim rečemo kontejnerji. Pri ročni namestitvi vso programsko opremo nameščamo direktno na strežnik, kar nam morebiti izboljša hitrost in odzivnost OpenCTI, vendar je težje za konfiguracijo in pusti veliko nereda oziroma programske opreme na strežniku. Na svojih namestitvah sem uporabljal docker način namestitve, saj sem se želel naučiti, kako se dela z docker-jem. (Filigran, 2023)

Namestitev z docker-jem poteka tako, da kopiramo centralo konfiguracijsko datoteko, z imenom *"docker-compose.yml"* na strežnik oziroma napravo, kjer bomo namestili OpenCTI. Konfiguracijsko datoteko moramo pred namestitvijo še izpolniti oziroma vpisati poverilnice za različne servise, ki se namestijo v kontejnerje. Ti servisi so:

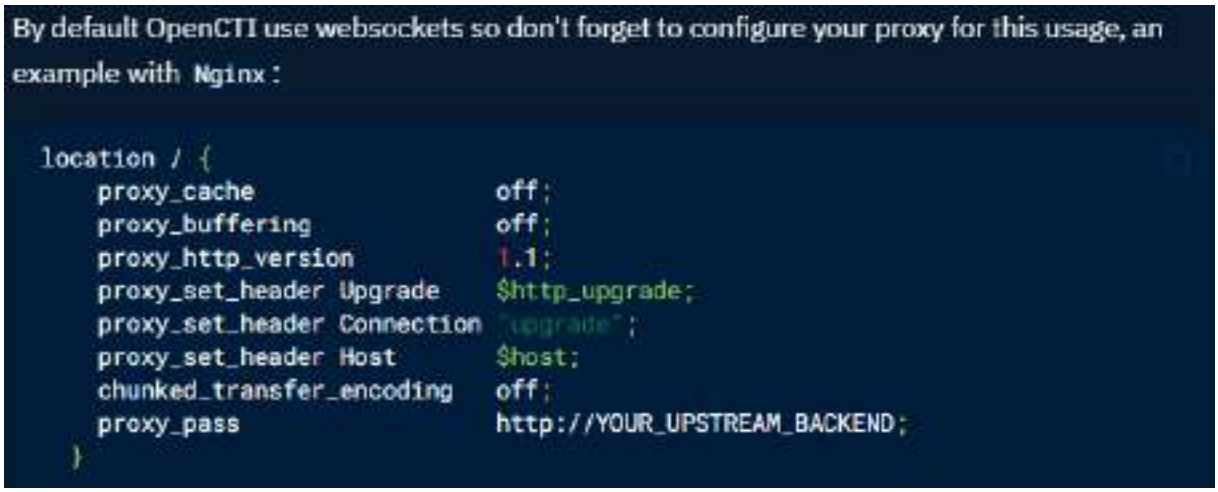
- E*lasticsearch*: je zmogljiv in skalabilen iskalnik in analitični pogon. V okviru OpenCTI se Elasticsearch uporablja za hitro indeksiranje in iskanje velikih količin podatkov o grožnjah,
- *MinIO*: se uporablja za shranjevanje in upravljanje velikih količin nestrukturiranih podatkov, kot so datoteke, slike in drugi binarni podatki. V OpenCTI se uporablja za shranjevanje artefaktov obveščanja o grožnjah, kot so vzorci zlonamerne programske opreme, poročila o grožnjah in druge povezane datoteke,
- *RabbitMQ*: je posrednik sporočil, ki omogoča komunikacijo med različnimi deli porazdeljenega sistema. Običajno se uporablja za upravljanje asinhrone komunikacije med različnimi komponentami aplikacije. V sistemu OpenCTI se RabbitMQ uporablja za upravljanje nalog, kot so sprejemanje, obdelava in distribucija podatkov o grožnjah med različnimi moduli ali komponentami,

- *Redis*: je shramba podatkovnih struktur v pomnilniku, ki se lahko uporablja kot predpomnilnik, posrednik sporočil in drugo. Znana je po hitrem branju in pisanju, zato je primerna za naloge, ki zahtevajo dostop do podatkov z majhno zakasnitvijo. V OpenCTI se Redis uporablja za predpomnilnik za pogosto dostopne podatke in upravljanje posodobitev v realnem času,
- Platforma OpenCTI: je odgovorna za postavitev spletnega vmesnika za dostop do celotnega sistema oziroma aplikacije.

Predpogoj izvajanja je še paket, ki nadgradi docker z imenom *docker-compose*, kar lahko storimo v Linux ukazni vrstici s pomočjo ukaza *"sudo apt install docker-compose"*. Po spremembi privzetih poverilnic in izpolnitve predpogojev, lahko testiramo ali vse deluje s pomočjo ukaza *"docker-compose up –d"*. Delovanje posameznih servisov in platforme lahko spremljamo iz ukazne vrstice s pomočjo ukaza *"docker ps"*, ki nam izpiše vse kontejnerje, ki se trenutno izvajajo in poleg njih izpiše tudi stanje izvajanja. Če nam ta ukaz izpiše vse servise in platformo OpenCTI-ja kot *"Running"*, nam le-to pove, da smo vse pravilno konfigurirali in je stran dostopna na lokalnem internetnem naslovu, ki smo ga podali v konfiguracijski datoteki.

Če želimo dostopati do OpenCTI iz drugih naprav v našem omrežju, moramo nanj namestiti tudi aplikacijo, ki bo servirala našo spletno stran. V mojem primeru namestitve sem uporabil Nginx, ki je uporabljen kot proxy. To pomeni, da služi kot vmesnik med uporabnikom in storitvijo. Po inštalaciji Nginx se nam, privzeto na našem IP naslovu računalnika, pokaže internetna stran, ki nam pove, da spletni strežnik deluje.(Nginx, 2023)

Da bi namesto privzete spletne strani Nginx prikazali našo spletno stran, je potrebno popraviti konfiguracijo le-tega. To lahko naredimo s spremembo konfiguracijske datoteke Nginx, ki se na Linux izvajalnem okolju nahaja na absolutni poti *»/etc/nginx/nginx.conf«*. OpenCTI dokumentacija nam pove tudi, kako naj le-tega konfiguriramo - spodnja slika prikazuje, kaj moramo dodati v konfiguracijsko datoteko.



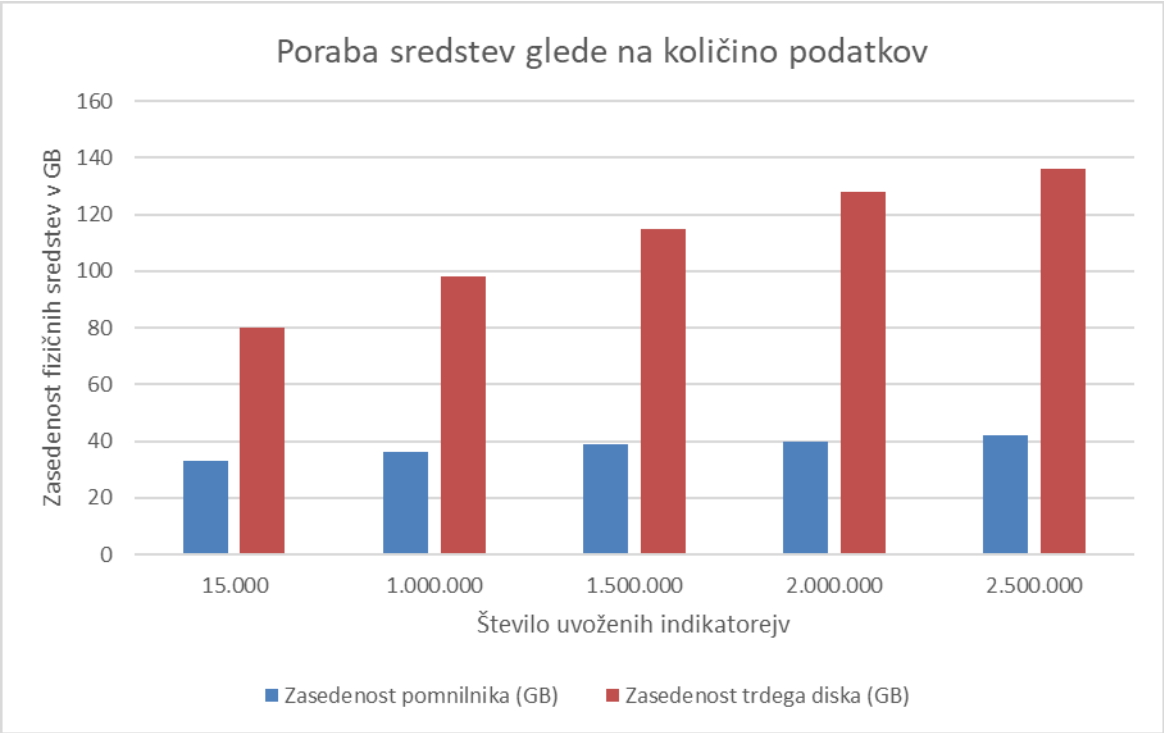
Slika 1: Konfiguracija spletnega strežnika Ngninx

Vir: (Filigran, 2023)

3.4 Testiranje zmogljivosti OpenCTI

Za namene raziskave sem si lokalno postavil instanco OpenCTI in testiral različne priključke za uvoz podatkov. Sklopu povezanih podatkov o kibernetskih grožnjah oziroma uvoženih podatkov se na področju kibernetske varnosti reče indikator kompromisa (angl. indicator of compromise - IOC). V fazi testiranja sem opazil, da za svoje delovanje porabi velik delež pomnilnika, ki ga ima na voljo, vendar le-to ne vpliva na delovanje strežnika. Problem, na katerega sem naletel, je bil pri uvozu podatkov.

Ob začetni inštalaciji sem strežniku podal na voljo 32 GB pomnilnika, kar je bilo za uvoz manjše količine podatkov (100.000 indikatorjev) dovolj, vendar ko sem želel prenesti večje količine podatkov (2 milijona indikatorjev) le-to ni več zadostovalo in je vplivalo na delovanje le-tega. Strežnik sem formatiral in mu dodal še 32 GB pomnilnika, vendar kljub temu je med uvozom deloval s težavami, kar se je kazalo v odzivnih časih iskanja in času uvoza (trajalo je mesec dni).



Graf 1: Poraba sredstev OpenCTI glede na količino podatkov

Vir: (Lasten vir)

Tabela 2: Poraba sredstev OpenCTI glede na količino indikatorjev

Število indikatorjev	**Zasedenost pomnilnika (GB)**	**Zasedenost trdega diska (GB)**
15.000	33	80
1.000.000	36	98
1.500.000	39	115
2.000.000	40	128
2.500.000	42	136

Vir : (Lasten vir)

4 PODROBNEJŠA PREDSTAVITEV PROBLEMA

V naši organizaciji uporabljamo SIEM rešitev, ki sama po sebi ne vsebuje dodatnih podatkov o grožnjah. Za zbiranje le-teh uporabljamo odprtokodno rešitev OpenCTI, ki podpira različne vire, tako javno dostope, kot tudi naše interne, ki jih v OpenCTI vnašamo preko poročil ali navadnega besedila. Koristimo tudi številne javno dostopne vire, s katerih črpamo podatke s pomočjo priključkov. Nekateri viri, s katerih črpamo podatke, so sledeči:

- *AlienVault*: je podjetje za kibernetsko varnost, ki ponuja vrsto varnostnih rešitev, vključno s platformo Unified Security Management (USM). Platforma USM združuje različne varnostne zmogljivosti, kot so odkrivanje vdorov, ocenjevanje ranljivosti, odkrivanje sredstev, spremljanje vedenja in obveščanje o grožnjah. Organizacijam omogoča, da s korelacijo in analizo podatkov iz več virov v realnem času odkrivajo grožnje in se odzivajo nanje.
- *AbuseIPDB*: je brezplačna spletna storitev, ki uporabnikom omogoča prijavo in iskanje naslovov IP, povezanih z zlonamernimi dejavnostmi. Ta platforma zbira in deli informacije o naslovih IP, ki so bili prijavljeni zaradi pošiljanja neželene pošte, poskusov vdora in drugih zlorab. Pogosto jo uporabljajo sistemski administratorji in strokovnjaki za kibernetsko varnost, da prepoznajo potencialno zlonamerne naslove IP in ustrezno ukrepajo.
- *AbuseSSL*: je podobno kot AbuseIP, brezplačna spletna storitev, katere cilj je odkrivanje zlonamernih povezav SSL z identifikacijo in uvrstitvijo na črno listo certifikatov SSL, ki jih uporabljajo ukazni strežniki (angl. command and control – C2C) botnetov. Poleg tega identificira prstne odtise JA3, ki pomagajo pri odkrivanju in blokiranju zlonamernih komunikacij botnetov C2C na plasti TCP.
- *MITRE ATT&CK*: (Adversarial Tactics, Techniques, and Common Knowledge) je celovit okvir, ki kategorizira taktike in tehnike, ki jih nasprotniki uporabljajo v različnih fazah kibernetskega napada. Zagotavlja standardiziran način opisovanja in razumevanja kibernetskih groženj. Okvir je organiziran v matrike, ki opisujejo taktike (cilji na visoki ravni) in tehnike (posebne metode), ki jih uporabljajo napadalci.

MITRE ATT&CK ni zgolj podatkovna zbirka, temveč služi kot zbirka tehnik, ki jih uporabljajo napadalci. S pomočjo tega okvirja se lahko podatki v OpenCTI obogatijo s strani napadalnih vektorjev in tehnik za doseganje ciljev zlonamernih akterjev.

Izziv je izvoz podatkov iz platforme OpenCTI v SIEM, ***ker povezava med tema platformama še ni podprta, oziroma rešitev, ki bi zadostovala našim potrebam, še ni bila razvita***. (Filigran, 2023) Podatke iz OpenCTI želimo v SIEM zapisati kot iskalno tabelo (angl. lookup table), saj bo to najbolj koristilo analitikom in bo omogočalo najlažjo implementacijo iskalnega ukaza za zaznavo potencialnih aktivnosti zlonamernih akterjev.

5 INTEGRACIJA PODATKOV O GROŽNJAH S SIEM REŠITVIJO

Problema integracije sem se lotil s temeljito raziskavo možnih rešitev in posledično našel knjižnico za programski jezik Python, ki je bila razvita z namenom komunikacije z zaledjem SIEM-a. Knjižnico sem testiral in naredil preizkus koncepta, vzpostavil prvotno povezavo modula in SIEM platforme. Ugotovil sem, da naš SIEM ne podpira dodajanja vrstic v iskalne tabele, ampak se lahko le-te zgolj prepisuje, kar bom moral upoštevati pri razvoju modula. Seznanil sem se tudi s podatkovno strukturo, ki jo SIEM definira kot iskalno tabelo. To so datoteke, v katerih so podatki ločeni z vejicami (angl. Comma Separated Values - CSV). Format je enostaven za generacijo, saj pomeni, da moramo samo zapisati vrednosti zaglavja oziroma prvo vrstico, s katero povemo imena stolpcev, nato pa vanjo zapišemo podatke, med katere ustavimo vejice, ki ločujejo stolpce med seboj. Vrstice so ločene z znakom za nove vrstice *"\n"* oziroma premikom v novo vrstico.

Preveril sem uvoz podatkov, nato sem se lotil še raziskave možnosti izvoza podatkov iz OpenCTI-ja. Tudi za OpenCTI sem našel Python knjižnico (Filigran, 2023), ki se je, po pregledu dokumentacije, izkazala za neuporabno. Namenjena je predvsem za razvoj vtičnikov (angl. connector) za OpenCTI, ki so običajno uporabljeni za uvažanje ali bogatenje (angl. enrichment) že obstoječega izvoza. Tudi periodičnega izvoza podatkov knjižnica ne podpira, zato sem se iskanja rešitve lotil v platformi, kjer sem odkril vire (angl. feed). Viri so spletni naslovi, na katere lahko pripnemo nastavljive vire izvoza podatkov, ter na le-teh ponastavimo podatke, vrstice in časovno obdobje za izvoz. Izbrani čas se definira v minutah in na našem viru se bodo izpisali vpisi, od trenutnega časa do prej izbranega časovnega intervala. (Filigran, 2023)

6 RAZVOJ MODULA ZA INTEGRACIJO

Modul sem razvijal v programskem jeziku Python, s katerim se lahko v zelo kratkem času razvije funkcionalno dovršen modul. Razvoj sem razdelil na različne podmodule. Začetek je vključeval vzpostavitev trdne povezave s sistemom SIEM, kar je postavilo trdne temelje za nadaljnje delovanje. Sledila je integracija podatkov OpenCTI, ki je izkoristila prilagodljivost Python-a za učinkovito pridobivanje in strukturiranje ustreznih informacij.

V ključni fazi je bilo treba analizirati podatke, pri čemer so se uporabile Python-ove zmožnosti za razčlenjevanje, s čimer so se podatki izboljšali za naslednji korak. To je pripeljalo do ključne faze prenosa razčlenjenih podatkov v pregledovalno tabelo SIEM, kjer so se izkazale Python-ove zmožnosti manipulacije s podatki.

Z združitvijo teh podmodulov se je oblikoval celovit in funkcionalen modul, ki je dokaz moči modularnega programiranja. Učinkovitost in vsestranskost Pythona sta bili gonilni sili te uspešne razvojne poti, ki se je končala z učinkovitim modulom, pripravljenim za izboljšanje varnostnih prizadevanj.

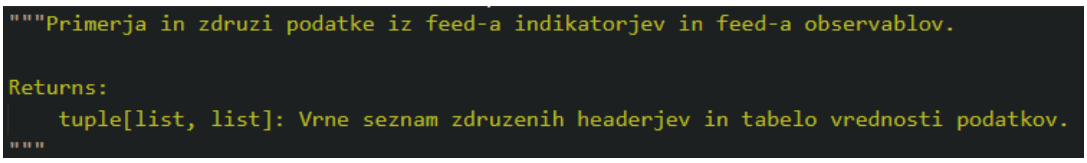
6.1 Uporabljena orodja

Držal sem se dobre prakse programiranja, ki je zbirka neformalnih pravil, ki pripomorejo k temu, da je koda bolj berljiva in modularna. Kot sem že omenil, so ta pravila neformalna, vendar ko pri razvoju sodeluje več programerjev, si sami določijo pravila, ki se jih posledično držijo, da je koda napisana bolj enotno. Ker teh pravil v našem oddelku še nismo definirali, sem se držal splošnih, kot so modularnost kode, intuitivna poimenovanja funkcij, izogibanje vdelani kodiranih vrednosti (angl. hardcoded) in uporabo orodja za kontrolo verzij.

Za kontrolo verzij (angl. version control) sem uporabljal Git, ki je eden izmed najbolj priljubljenih orodij za razvijalce. S pomočjo tega orodja sem imel popoln pregled nad kodo, ki sem jo dodajal. Poleg tega je delovalo kot mesto, kjer se v primeru porušenja ali izbrisa modula nahaja varnostna kopija. Uporaba Git-a je zagotovila tudi zelo lahko namestitev modula v produkcijsko okolje, saj sem lahko celoten modul iz repozitorija direktno kloniral na virtualno okolje.

Za pisanje kode oziroma modula sem uporabljal odprtokodni program Visual Studio Code. Za to urejevalno okolje (angl. Integrated Development Environment - IDE) sem se odločil, ker je brezplačno in ponuja veliko razširitev, ki so mi zelo olajšale razvoj. Uporabljal sem razširitve,

ki pametno dopolnjujejo še nenapisano kodo in predlagajo možno kodo, ki jo lahko napišemo, ter razširitev, ki izboljša oziroma doda avtomatsko generiranje komentarjev za funkcije.



Slika 2: Avtomatično generiran komentar funkcije

Vir: (Lasten vir)

Uporaba IDE mi je pri razvoju omogočila predvsem hitrejši razvoj s pomočjo vgrajenega razhroščevalnika (angl. debugger). Ta funkcija nam omogoči, da ob določenih točkah delovanje programa ustavimo. To dosežemo z implementacijo tako imenovanih točk prekinitve (angl. breakpoint), ki jih nastavimo kjer koli v kodi. V nadaljevanju sem opisal delovanje prekinitvenih točk v IDE:

1. Nastavitev prekinitve: V IDE lahko točko prekinitve nastavite tako, da kliknete na območje za prekinitev ob določeni vrstici kode. To je običajno označeno z rdečo piko ali podobnim indikatorjem. Ko je točka prekinitve nastavljena, pove razhroščevalniku, da ustavi izvajanje programa, ko je dosežena ta vrstica kode.
2. Začetek razhroščevanja: Za začetek načina razhroščevanja zaženete program z omogočenim razhroščevalnikom. Ko izvajanje doseže vrstico kode s točko prekinitve, se program ustavi in razhroščevalnik prevzame nadzor.
3. Pregledovanje spremenljivk: Medtem ko je program ustavljen na točki prekinitve, lahko pregledujete vrednosti spremenljivk, predmetov in drugih podatkovnih struktur. To vam pomaga razumeti, kaj se dogaja v programu na določeni točki, in je lahko ključnega pomena pri odkrivanju težav.
4. Korak skozi kodo: Ko je razhroščevalnik aktiven, lahko korak za korakom pregledate kodo po vrsticah. To pomeni, da lahko program izvajate vrstico za vrstico ter opazujete spremembe spremenljivk in obnašanja. To je izredno koristno pri odkrivanju logičnih napak in razumevanju poteka izvajanja.
5. Nadaljevanje izvajanja: Ko preučite stanje programa in morebiti spremenite spremenljivke, lahko nadaljujete izvajanje programa s trenutne prekinitvene točke. Tako si lahko ogledate, kako vaše spremembe vplivajo na nadaljnje obnašanje programa.

6. Odstranjevanje točk prekinitve: Točke prekinitve lahko po potrebi odstranite. To običajno storite tako, da kliknete na indikator točke prekinitve.

Prekinitvene točke so bistveno orodje za odpravljanje napak in diagnosticiranje težav v vaši kodi. V kritičnih trenutkih vam omogočajo vpogled v notranje delovanje programa, pomagajo vam pri odkrivanju napak in razumevanju zapletenega obnašanja. Še posebej so uporabne, kadar vaša koda ne daje pričakovanih rezultatov ali kadar poskušate ugotoviti vzrok za sesutje ali nepravilno delovanje.

7 OPIS DELOVANJA MODULA

V naslednjih poglavjih bom opisal delovanje in potek modula ter predstavil ključne dele kode. Ker je koda razdeljena na podmodule oziroma razrede in posledično bolj berljiva, jo bom razložil po vrstnem redu. Začel bom s konfiguracijo modula in datoteke, zadolžene za to, nato bom opisal branje podatkov z OpenCTI, nadaljeval s komunikacijo s SIEM in predstavil celoten potek modula. Čisto na koncu bom predstavil še dva dodatna razreda, ki služita zgolj kot izboljšave kvalitete (angl. Quality of life improvements). Delčki predstavljene kode bodo morda ob branju nejasni, vendar bodo ob razlagi funkcij, razloženi tudi takšni delčki kode.

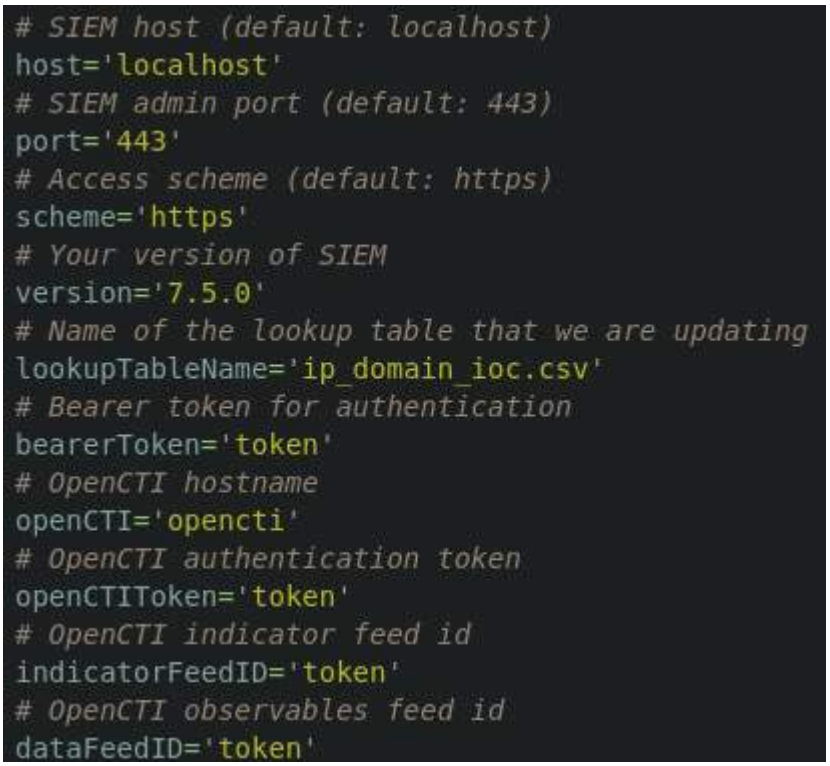
7.1 Uporabljene knjižnice

Programski jezik Python je znan po tem, da ima razvitih veliko knjižnic, ki olajšajo delo z različnimi podatkovnimi strukturami in realizacijo naših rešitev. Uporabljene knjižnice v posameznih razredih bom opisal pri vsakem razredu posebej. Nekaj le-teh uporabljamo po celotnem modulu:

- *datetime*: vsebuje razred datetime, ki nam pomaga pri generaciji in branju datumov,
- *dotenv*: modul za lažje branje konfiguracijske datoteke, ki le-to pretvori v slovar,
- *os*: večinoma uporabljen modul za branje trenutnega delovnega direktorija (angl. Current Working Directory - CWD) za to, da program najde pravilne poti za pisanje oziroma branje pomožnih datotek.

7.2 Konfiguracijska datoteka

Ker so vdelani podatki v kodi slaba praksa, sem v modul vgradil namensko datoteko "*.env*". V datoteko uporabnik oziroma izvajalec vpiše vse podatke, ki jih modul zahteva za delovanje. Ker občutljivih podatkov ne smemo objavljati v repozitorij, sem v datoteko "*.gitignore*" zapisal ime namenske datoteke, kar pomeni, da se ta datoteka pri objavljanju kode na repozitorij ignorira. Za boljšo uporabniško izkušnjo oziroma da bo oseba, ki bo ta modul uporabljala vedela, da program pričakuje vhodno konfiguracijsko datoteko, sem ustvaril še datoteko *".env.example*", v katero sem zapisal primer datoteke in komentarje, kako naj izgleda. To uporabniku omogoča, da podatke samo zamenja s svojimi in shrani kot "*.env*" ter tako konfigurira modul.



Slika 3: Datoteka ".env.example"

Vir: (Lasten vir)

7.3 Branje podatkov z OpenCTI virov

Vire (angl. feed) na OpenCTI lahko najdemo pod menijem na levi strani pod izmenjavo podatkov (angl. data sharing). Platforma nam ponuja tri različne vrste konfiguracije virov, ki se po sebi razlikujejo zgolj v formatu podatkov. Možnosti za izvoz podatkov so sledeče:

- podatke lahko izvozimo preko prenosa v živo (angl. livestream),
- izpis podatkov na namensko spletno stran v obliki CSV, kjer so podatki zapisani v telo spletne strani (HTML - Hyper Text Markup Language),
- zaupanja vredni avtomatizirani izmenjavi obveščevalnih informacij (angl. Trusted Automated eXchange of Intelligence Information - TAXII) strežnika, ki posreduje podatke v obliki objektnega zapisa JavaScript.

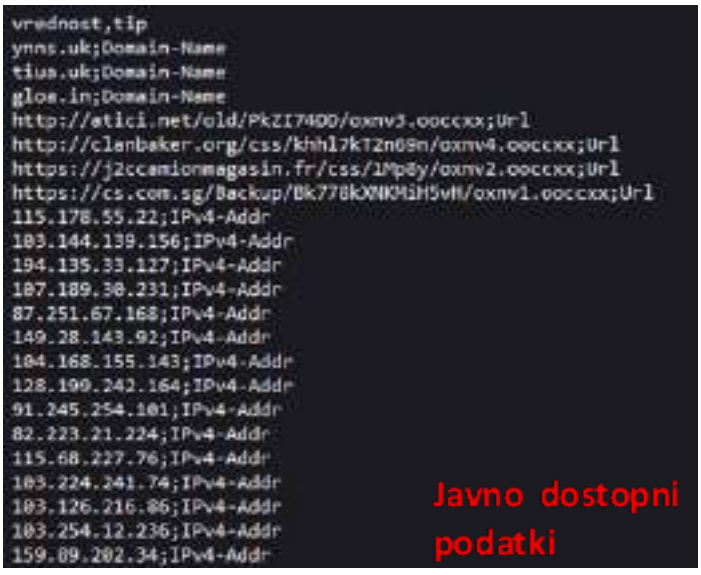
Ni nujno, da je izvažanje podatkov popolnoma ažurno in zato nam prenos v živo ne koristi. Po primerjavi ostalih možnosti sem ugotovil, da izvoz JSON objektov za prikaz podatkov potrebuje veliko več časa, poleg tega se izpiše tudi veliko nepotrebnih podatkov. Po izločitvi ostalih možnosti sem se odločil za izvoz preko spletne strani, na kateri se podatki nahajajo v

obliki CSV. Tovrstni izvoz lahko prilagodimo glede na naše potrebe, vendar tudi ta rešitev ni brez slabosti.

Iskalno tabelo želimo ustvariti kot tabelo, ki bo vsebovala podatke, kot so IP naslov, domena, datum začetka in konca veljavnosti indikatorja. Viri OpenCTI-ja delujejo na principu izbire objektov, ki jih želimo izvažati. Objekti, ki jih izvažamo, so v formatu za izražanje strukturiranih informacij o grožnjah (angl. Structured Threat Information Expression - STIX). (OASIS, 2023)

Izziv predstavlja datum veljavnosti, ki ga objekt s samo vrednostjo, na primer domeno, ne vsebuje. Podatek o veljavnosti indikatorjev je za našo iskalno tabelo pomemben, saj želimo na SIEM-u zaznavati le aktivnosti nedavno zaznanih indikatorjev. Če tega ne bi upoštevali, lahko pri zaznavi možnih aktivnosti odkrijemo tudi zastarele indikatorje, ki mogoče niso več aktivni.

Ta izziv je rešen v modulu, vendar moramo dodati še dodaten vir, kateri izpiše še drugi objekt, ki vsebuje tudi datume. Glavni razlog za dva vira je tudi to, da z izvozom objekta, ki je brez datumov, pridobimo dostop še do vrednosti, ki nam pove tip te vrednosti. To nam pomaga pri razčlenjevanju podatkov v vrednosti, ki jih pričakujemo, na primer domene brez "*www.*". Podatki, ki jih potrebujemo v iskalni tabeli, morajo biti sestavljeni iz IP-jev in domen, zato bo konfiguracija vira taka, da bo vir izpisoval objekte "*Domain-name*", "*Hostname*", "*Url*" in "*IPv4-Addr*".



Slika 4: Izpis podatkov OpenCTI na vir

Vir: (Lasten vir)

7.3.1 Razred OpenCTIFeed

Razred je zadolžen za pobiranje podatkov iz virov OpenCTI-ja in razčlenjevanje le-teh v tabele, da jih lahko lažje uporabljamo v ostalih delih programa. Pomaga tudi z urejanjem vrednosti, na primer brisanjem "www." ali "https://" in dodatnih številk vrat (angl. port), ki se lahko pojavijo v zapisu vrednosti. Razred uporablja tudi knjižnice za lažje razčlenjevanje in pridobitev podatkov. Uporabljene knjižnice so naslednje:

- *requests*: uporabljen za pridobitev spletne strani v obliki običajnega besedila (angl. plain text),
- *bs4* oziroma BeautifulSoup: razčlenjevalnik (angl. parser) uporabljen za razčlenitev prej pridobljenega besedila spletne strani,
- *urllib.parse* (iz njega urlparse): le-to uporabimo za izluščevanje domen iz URL naslovov,
- *urllib3.exceptions* (iz njega InsecureRequestWarning): s pomočjo tega modula onemogočimo izjeme, ki jih sproža modul za zahtevke, ker imamo OpenCTI postavljen lokalno in je posledično brez certifikata.

7.3.2 Predstavitev funkcij razreda OpenCTIFeed

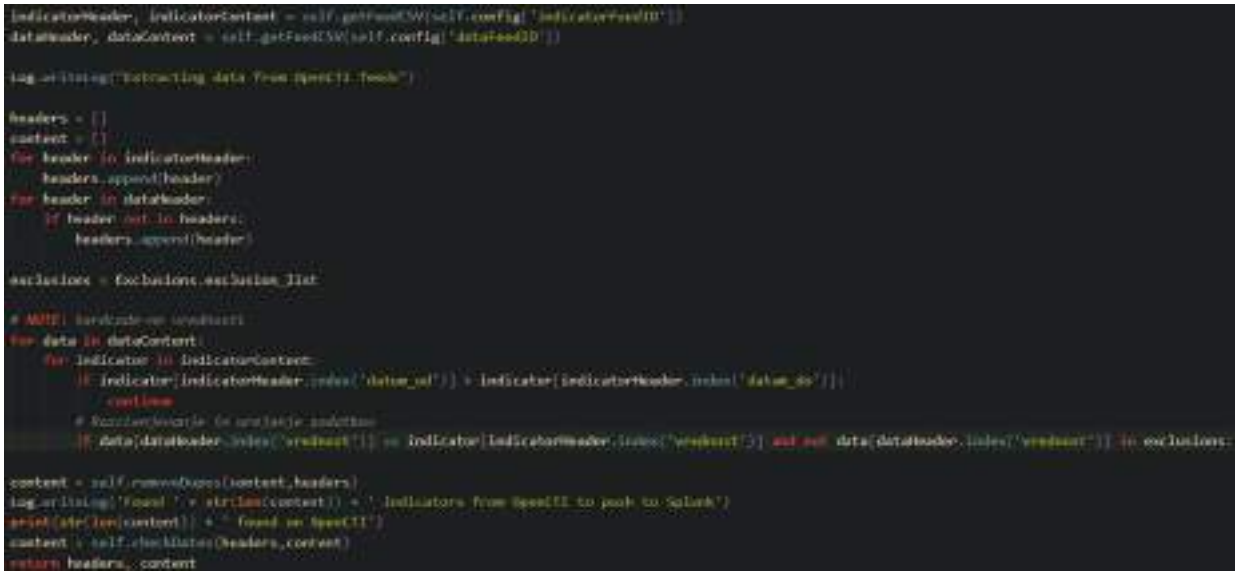
Ob kreaciji instance razreda se ob tem kliče tudi konstruktor. To je del kode oziroma funkcija, ki se kliče ob ustvarjeni instanci razreda. Opcijsko lahko privzeti konstruktor prepišemo in dodamo kodo, ki bi nam koristila. V večini primerov bo ta koda vsebovala nastavljanje spremenljivk v razredu na vrednosti, ki jih prejme preko argumentov ali drugih virov. V modulu prepišemo privzeti konstruktor in v njem naredimo dve stvari, onemogočimo že omenjene izjeme (angl. exception), ki se pojavijo s pošiljanjem zahtev za strani, ki nimajo certifikata in so zato HTTP ne pa HTTPS, ter s pomočjo modula dotenv preberemo konfiguracijsko datoteko in vrednosti le-te naložimo v lokalni seznam.

Slika 5: OpenCTIFeed konstruktor

Vir: (Lasten vir)

Kot je bilo že prej omenjeno, vir podatkov za polnjenje tabele so viri OpenCTI-ja, ki so v obliki spletne strani. Iz spletne strani modul pobere HTML kodo, v kateri so zapisani podatki, ki jih potrebujemo. Za zahtevek spletne strani moramo dodati tudi avtentikacijo, ker jo OpenCTI zahteva za dostop do virov. To storimo z dodanim žetonom za avtentikacijo, ki ga prebere iz konfiguracijske datoteke in doda v zaglavje zahtevka. Podatki so zapisani v formatu CSV in med seboj ločeni s podpičjem. S pomočjo tovrstnih ločil razčlenimo podatke v dvodimenzionalni seznam, kar pomeni, da imamo seznam vrstic in posamezna vrstica je sestavljena iz seznama stolpcev. Navedena logika je vgrajena v funkcijo, v katero kot parameter pošljemo identifikacijsko številko (ID) samega vira, le-ta pa vrne dva seznama, in sicer seznam stolpcev zaglavja in podatke, ki so urejeni po enakem vrstnem redu kot zaglavje.

Glavna funkcija programa uporabi prej opisano funkcijo, da pridobi podatke z virov in jih združi v seznam. Podatke združi po vrednostih, saj ima STIX objekt enake vrednosti, zapisane v indikatorju okužbe, kot tudi v objektu tipa spremenljivke. Vrednosti, ki jih zapiše v združen seznam tudi razčleni, kar pomeni, da domenam in IP naslovom odstrani številke vrat itd.



Slika 6: Glavna funkcija razreda OpenCTI, skrajšana

Vir: (Lasten vir)

Razred vsebuje tudi dve funkciji, *removeDupes* in *checkDates*, ki jih uporabi v glavni funkciji za pomoč pri razčlenjevanju in urejanju podatkov. Prva izmed funkcij je namenjena preprečevanju podvojenih podatkov v združenem seznamu. To reši tako, da najprej iz seznama izloči identične vpise, saj se lahko s tem, ko podatke razčlenimo in preuredimo, le-ti podvojijo. Tako smo izbrisali identične duplikate, zgodi se lahko, da so vrednosti v seznamu enake, le datumi veljavnosti so različni. To rešimo v nadaljevanju funkcije s ponovno iteracijo čez

seznam in brisanjem podvojenih vrednosti, ki imajo manjši datum veljavnosti. V končni seznam zato dobimo le vrednosti z najdaljšim datumom veljavnosti.

```
results = []
[results.append(x) for x in content if x not in results]

dateFormat = "%Y-%m-%dT%H:%M:%S.%fZ"
for result in results:
    datum_do = datetime.strptime(result[headers.index("datum_do")],dateFormat)
    for deeperResult in results:
        if result[headers.index('vrednost')] == deeperResult[headers.index('vrednost')]:
            datum_doX = datetime.strptime(deeperResult[headers.index("datum_do")],dateFormat)
            if datum_do < datum_doX and datum_do != datum_doX:
                results.pop(results.index(result))
            elif datum_do != datum_doX:
                results.pop(results.index(deeperResult))

return results
```

Slika 7: Funkcija za izbris podvojenih podatkov – removeDupes

Vir: (Lasten vir)

Druga funkcija *checkDates* je zadolžena za popravljanje datumov, ki jih prejme od OpenCTI. Problem je, ker sami podatki pogosto nimajo zapisanega datuma zapadlosti, ki se po privzetem zapiše enako kot začetni datum oziroma datum, ko so bili podatki vneseni v OpenCTI. Take podatke uredi tako, da jim samo nastavi datum kreacije oziroma začetek veljavnosti na 7 dni pred datumom zapadlosti. Ker se podatki glede na nastavljen časovni interval na viru lahko spremenijo, moramo v tem primeru posodobiti podatke, kar storimo v nadaljevanju.

```
dateFormat = "%Y-%m-%dT%H:%M:%S.%fZ"
for i in range(len(content)-1):
    datum_od = datetime.strptime(content[i][headers.index("datum_od")],dateFormat)
    datum_do = datetime.strptime(content[i][headers.index("datum_do")],dateFormat)

    if datum_od == datum_do:
        datum_od = datum_od - timedelta(days=7)
        dateString = datetime.strftime(datum_od,dateFormat)
        dateString = dateString.split(".")
        dateString[1] = dateString[1][:-4] + "Z"
        dateString = ".".join(dateString)
        content[i][headers.index("datum_od")] = dateString

return content
```

Slika 8: Funckija za pravilno nastavitev datumov veljavnosti – checkDates

Vir: (Lasten vir)

7.4 Komunikacije s SIEM

Komunikacija preko SIEM poteka zgolj preko ukazne vrstice in zaledja (angl. backend), ki ga za delovanje našega modula ne uporabljamo oziroma potrebujemo. Komunikacija poteka tako, da lahko po vzpostavljeni povezavi na SIEM-u izvajamo iskalne ukaze, kot da bi uporabljali SIEM. S pomočjo ukazne vrstice lahko izvedemo vrsto različnih ukazov, kot na primer izvajamo iskanje nad že vnesenimi podatki v naš SIEM, definiramo vizualizacijo le-teh, izpišemo podatke iskalne tabele ali podatke iskalne tabele prepišemo.

Ostali ukazi, ki jih lahko izvedemo preko zaledja storijo naslednje:

- naložijo nove aplikacije v SIEM,
- spreminjajo konfiguracijske datoteke in nastavitve,
- upravljajo s strežniki SIEM in odjemalci,
- upravljajo vnose podatkov,
- upravljajo licence,
- upravljajo s sistemskimi viri in iskalnimi delovnimi obremenitvami,
- izvajajo dejanske iskalne ukaze.

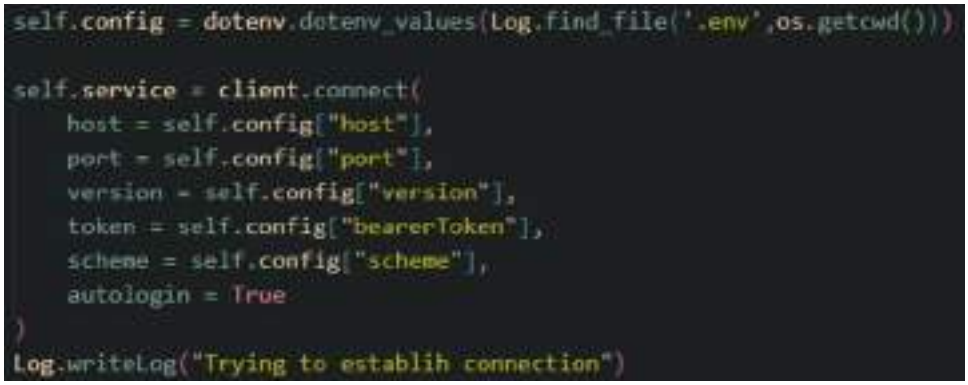
Glede na to, da bomo uvažanje podatkov in posodabljanje iskalne tabele delali preko iskalne vrstice, moramo upoštevati potreben format za pravilno kreiranje iskalne tabele. Za pravilno generacijo le-te potrebujemo kar nekaj zaporedno izvršenih ukazov, ki so sestavljeni iz definicije zaglavja, regularnega izraza (angl. Regular expression – *regex*) zaglavja in podatkov samih.

7.4.1 Razred SIEMAPICommunication

V tem razredu je realizirana vsa funkcionalnost, ki jo potrebujemo za našo komunikacijo s SIEM. Komunikacija se vzpostavi s pomočjo modula za API komunikacijo v konstruktorju razreda, kjer se komunikacija ustvari in shrani kot storitev (angl. service). Podatke za vzpostavitev povezave, enako kot za prejšnji razred, prebere iz lokalne datoteke. Parametri, ki jih potrebujemo za kreacijo storitve, so:

- *host* - naslov, na katerem se nahaja strežnik,

- *port* - vrata zaledja strežnika, preko katerega lahko komuniciramo z njim,
- *version* - verzija naše SIEM platforme,
- *token* - žeton SIEM-a, ki ga uporabimo za avtentikacijo,
- *scheme* - protokol, na katerem se nahaja spletna stran strežnika,
- *autologin* - preklopna spremenljivka, ki klicu funkcije za ustvarjanje storitve pove ali naj se takoj prijavi s podatki, ki jih prejme.



Slika 9: Funkcija za vzpostavitev povezave s SIEM

Vir: (Lasten vir)

Preko te shranjene storitve lahko s pomočjo vgrajenega objekta v storitvi, imenovanega zaposlitve (angl. jobs), izvajamo iskalne ukaze. Ukazi se izvajajo asinhrono, kar pomeni, da moramo počakati na rezultate ukaza. To dosežemo s funkcijo spanje (angl. sleep). V našem primeru počakamo pol sekunde, kar je po navadi dovolj, da dobimo rezultate iskalnega ukaza.

7.4.2 Funkcija za izvajanje iskalnih ukazov

Funkcija kot parametre prejme iskalni ukaz, ki ga želimo izvesti in opcijski parameter, s katerim lahko določimo način izvajanja (angl. execution mode) samega ukaza. Po tem se s pomočjo prej ustvarjene storitve ustvari zaposlitev, ki jo shranimo v lokalno spremenljivko, preko katere lahko spremljamo izvajanje le-te in vrnjene rezultate iz SIEM-a. Ker moramo počakati, da se zaposlitev pripravi v ozadju SIEM-a, le-to zagotovimo tako, da s pomočjo skoraj neskončne zanke konstantno preverjamo, ali je storitev že pripravljena na obdelavo. Nato sledi neskončna zanka, v kateri se na standardni izhod beleži stanje izvajanja ukaza, kar je zelo pomagalo pri razhroščevanju (angl. debugging) funkcije. Ko se v objektu seznama nastavi zastavica (oziroma

polje, ki lahko ima samo vrednost 1 ali 0, angl. true or false), zadolžena za sporočanje končanja iskalnega ukaza na true, lahko čez dobljene podatke iteriramo.

Dobljeni rezultati so zapisani v obliki pretočnega bralnika XML (angl. Extensible Markup Language), ki ga lahko pretvorimo v manjše kose XML, JSON ali CSV. Podatke lahko preberemo z iteracijo čez njih, vendar ta iteracija naredi ponovno iteracijo podatkov nedostopno, kar pomeni, da v primeru, da potrebujemo podatke večkrat, moramo ponoviti iskalne ukaze. V našem modulu smo izbrali izpis v obliki JSON formata, saj bomo tako podatke najlažje prebrali. Izhod funkcije je objekt, ki že vsebuje prebran in razkosan tok (angl. stream) rezultatov in naše SIEM platforme.

Slika 10: Funkcija za izvajanje iskalnih ukazov

Vir: (Lasten vir)

7.4.3 Razčlenjevanje podatkov in ustvarjanje varnostne kopije

Za razčlenjevanje podatkov sem napisal namensko funkcijo. Kot parametre prejme iterativen objekt podatkov, spremenljivko, ki jo lahko nastavimo na znak, ki loči vrstice med seboj in

zastavico, ki funkciji pove ali zapiše tudi zaglavje in *regex* zaglavja v spremenljivke razreda. S pomočjo te zastavice se funkcija razveje na dva dela.

Prvi del se zgodi, v kolikor je zastavica nastavljena na *true*. Ustvari se CSV besedilo (angl. string) brez vrstice zaglavja. Zaglavje zapiše v namenske spremenljivke razreda in tudi *regex* le-tega (*regex* je zaporedje znakov, ki določa vzorec iskanja v besedilu). To pomeni, da se vsako polje zaglavja zapiše z dodatnimi znaki, ki jih lahko vidimo na spodnjem odseku kode. Vrstice podatkov so ločene z znakom, zapisanim v parametru funkcije.

```
text = ""
if includeHeaders:
    for element in dataJSON:
        if hasattr(element, 'type'):
            continue
        for key, value in element.items():

            text += value
            if value != list(element.values())[len(element.values())-1]:
                text += ","
            else:
                text += delimiter

            if self.headerTable == "":
                self.headerTable = key
                self.headerRegex = "(?<"+key+">.*)"
            else:
                self.headerTable = self.headerTable + "," + key
                self.headerRegex = self.headerRegex + ",(?<"+key+">.*)"
        if self.headerTable != "":
            break
```

Slika 11: Del funkcije za razčlenjevanje podatkov

Vir: (Lasten vir)

Drugi del funkcije je podoben, samo da besedilo vsebuje tudi zaglavje pred podatki. Pred iteracijo čez podatke je uporabljen ukaz, ki preveri ali ima element seznama podatkov atribut tip (angl. type), ki je namenjen temu, da se izpusti vrstica, kjer nam SIEM pove metapodatke izvedenega ukaza. Po končani iteraciji čez podatke CSV besedilo vrne kot rezultat funkcije, besedilo pa lahko direktno uporabi v funkciji za ustvarjanje varnostne kopije.

Varnostne kopije iskalne tabele na SIEM-u se ustvarijo tam, kjer modul izvedemo oziroma poženemo, ob pisanju naloge je mesto izvajanja skripte na OpenCTI strežniku. Kot parameter prejme še neobdelane podatke iskalnega ukaza in ime datoteke, v katero se bo zapisala varnostna kopija, ki je po privzetem imenu *"backup.csv"*. V kolikor datoteka s tem imenom že

obstaja, jo izbriše, da ustvari novo s posodobljenimi podatki. Za prepis ne uporablja posebne logike in samo zapiše dobljene rezultate zgoraj opisane funkcije.

```
text = self.parseJSONIntoCSVString(dataJSON)

if os.path.isfile(os.getcwd() + '\\' + filename):
    os.remove(os.getcwd() + '\\' + filename)

Log.writeLog("Backup created")
with open(filename, 'w') as f:
    f.write(text)
    f.close()
```

Slika 12: Funkcija za ustvarjanje varnostnih kopij

Vir: (Lasten vir)

7.4.4 Primerjava podatkov in zapis v SIEM

Funkcija je namenjena primerjanju podatkov že zapisanih v SIEM in podatkov, ki jih dobimo od OpenCTI platforme. Te podatke prejme preko parametrov, ki so že pretvorjeni in urejeni podatki s SIEM iskalne tabele, zaglavja podatkov OpenCTI in vrednosti podatkov ter znak, ki loči vrstice med seboj, ki bi moral biti enak kot v klicu funkcije za generiranje besedila CSV. Funkcija ne služi samo za primerjavo podatkov iskalne tabele in OpenCTI podatkov, vendar ima v sebi implementiranih še nekaj drugih funkcionalnosti, kot so:

- Uvrščanje vrednosti zapisov v prave stolpce (na primer domene zapiše v tako imenovano polje, enako stori z IP naslovi).
- V prazne stolpce umesti besedilo "UNKNOWN" (ker kot smo že prej omenili, so namensko ustvarjeni stolpci za IP in domene in za vsak opis bo zapolnjen le eden s pravo vrednostjo).
- Preveri, ali je vrednost že zapisana v iskalni tabeli in v takem primeru odstrani ter jo posodobi z novim zapisom.

Funkcija je sama po sebi zelo neberljiva, saj se samo zaglavje in posledično vrednosti stolpcev določijo mehko kodirano (angl. soft coding), kar pomeni, da se podatki pridobijo ob izvajanju kode in niso vnaprej definirani. Le-to pomaga predvsem pri preprečevanju morebitnih problemov z vrstnim redom stolpcev, kar bi otežilo delo z modulom.

7.4.5 Funkcija za zapis podatkov v iskalno tabelo

Ta zelo preprosta funkcija je napisana za pomoč zapisovanja podatkov v iskalno tabelo, kar doseže s klicem prej opisane funkcije za izvajanje iskalnih ukazov. Iskalni ukaz definira vrstice zaglavja s pomočjo prej definiranega regex-a zaglavja in podatkov zaglavja ter vrednosti iskalne tabele, ločenih z vejicami, vrstice pa so med seboj ločene s podpičjem oziroma znakom, ki ga definiramo znotraj iskalnega ukaza. Na koncu sledi še ukaz, ki zapiše vse prej definirane podatke v iskalno tabelo. Po izvedenem ukazu se, kot smo že prej opisali, vrnejo tudi rezultati izvedenega ukaza, ki so sestavljeni iz enako kot zgoraj, vrstice, ki nam pove status storitve in število vseh vrstic v iskalni tabeli. Rezultate v tem primeru samo preštejemo, da preverimo, ali so se ustvarili novi vpisi ali ne, število vpisov pa tudi za namene razhroščevanja izpiše na standardni izhod.

```
count = 0
for i in reuslts:
    if hasattr(i, 'type'):
        print(i)
        continue
    count += 1
Log.writeLog("SIEM lookup table now has " + str(count) + " entries")
print("SIEM lookup table now has ", count, " entries")
```

Slika 13: Izpis števila vpisov iskalne tabele

Vir: (Lasten vir)

7.5 Pomožna razreda

V okviru delovanja modula sem napisal kodo za dva pomožna razreda. Razred Exclusions je uporabljen za branje lokalne datoteke, v kateri so zapisani podatki, ki naj jih modul izključi iz vpisa v iskalno tabelo. Razred Log se uporablja za beleženje dnevniških zapisov (angl. log) o izvajanju modula v lokalni datoteki. Oba razreda in njune funkcije sem opisal v nadaljevanju.

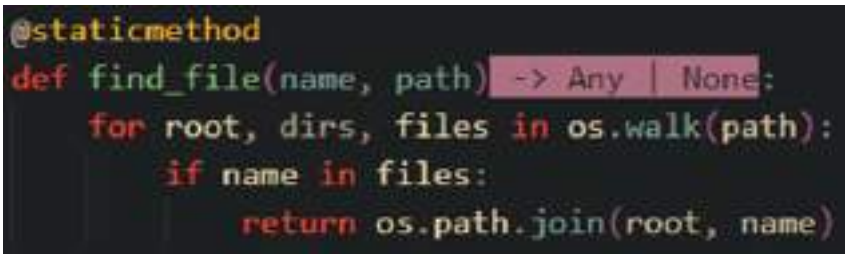
Funkcije razredov imajo pred seboj napisano besedilo "@staticmethod". To pomeni, da so funkcije statične in dostopne brez instance razreda, v katerem so definirane. Statične funkcije lahko uporabimo v kodi tako, da uvozimo razred, v katerem je funkcija, ki jo potrebujemo. Pokličemo jo s tem, da napišemo ime razreda, dodamo piko in ime želene funkcije.

Slika 14: Klic statične funkcije

Vir: (Lasten vir)

7.5.1 Pomožni razred za beleženje delovanja modula

Razred v sebi vsebuje dve statični funkciji, ki pomagati najti datoteke in dodati želeno besedilo datoteki za beleženje delovanja modula. Iskanje datotek je realizirano v za to namenjeni funkciji, ki kot parametra prejme ime datoteke, ki jo iščemo in po kateri poti datotečnega sistema želimo določeno datoteko najti. Nato se funkcija s pomočjo modula "os" sprehodi po vseh datotekah, ki se nahajajo v nadaljevanju poti, ki smo jo vnesli v parametrih, in le-te primerja z imenom datoteke v parametrih. V kolikor najde datoteko, funkcija vrne absolutno pot do najdene datoteke. Funkcija ne podpira iskanja več kot ene datoteke, saj le-ta funkcionalnost ni bila potrebna za delovanje modula. Primer uporabe te funkcije bomo lahko videli v opisu naslednje funkcije, izven tega razreda pa pri ustvarjanju varnostne kopije in pri iskanju konfiguracijske datoteke.



Slika 15: Funkcija za iskanje datotek na določeni poti

Vir: (Lasten vir)

Druga funkcija tega razreda služi za zapis v parametre podanega besedila v datoteko, kjer beležimo dogajanje modula. Iz kode lahko takoj vidimo klic zgoraj opisane funkcije s pomočjo katere najde pot do datoteke "log.txt". Sledi vejitev funkcije v dva možna scenarija, in sicer v primeru, da datoteka še ne obstaja, kar pomeni, da tudi funkcija za iskanje datoteke ni vrnila ničesar, se datoteka odpre v načinu pisanja, kar posledično ustvari tudi datoteko. Druga možnost je, da datoteka že obstaja, in sicer se takrat datoteka odpre v načinu dodajanja, kar v bistvu samo doda poljubno besedilo na koncu datoteke. Besedilu, ki se zapiše v dnevniško datoteko, dodamo spredaj še datum in čas, kdaj se je ta zapis ustvaril ter na koncu besedila še znak za novo vrstico.

```
@staticmethod
def writeLog(text) -> None:
    filePath = Log.find_file('log.txt',os.getcwd())
    if filePath == None:
        with open("log.txt", "w") as f:
            f.write("[" + datetime.now().strftime("%d/%m/%Y %H:%M:%S") + "] " + text + "\n")
            f.close()
    else:
        with open(filePath, "a") as f:
            f.write("[" + datetime.now().strftime("%d/%m/%Y %H:%M:%S") + "] " + text + "\n")
            f.close()
```

Slika 16: Funkcija za zapisovanje besedila v dnevniško datoteko

Vir: (Lasten vir)

7.5.2 Pomožni razred za izključitev vrednosti

Razred za svoje delovanje uporabi pomožno funkcijo za iskanje datotek, s pomočjo katere najde datoteko "exclusions.csv", v kateri so zapisane vrednosti IoC-jev, ki jih želimo ignorirati oziroma katere nočemo zapisati v iskalno tabelo. To funkcionalnost dosežemo z razredno spremenljivko, ki je seznam vrednosti, zapisanih v datoteki. Po preskoku zaglavja oziroma imen stolpcev, funkcija zapiše vse vrednosti v ta seznam. Ko je seznam enkrat napolnjen oziroma ko je bila enkrat ustvarjena instanca tega razreda, lahko seznam beremo kjer koli v modulu, vendar mora biti uvožen v naš pomožni razred.

```
class Exclusions:
    exclusion_list = []

    def __init__(self) -> None:
        self.fill_exclusion_list()

    def fill_exclusion_list(self) -> None:
        with open(Log.find_file('exclusions.csv',os.getcwd()),"r") as f:
            csv_reader = csv.DictReader(f,["exclusion"])
            # Preskoči imena stolpcev
            next(csv_reader)
            for line in csv_reader:
                self.exclusion_list.append(line["exclusion"])
            f.close()
```

Slika 17: Razred za branje izključenih vrednosti

Vir: (Lasten vir)

7.6 Združitev kode v celovito delovanje

Da bi vsi razredi in funkcije, ki jih vsebujejo, delovali kot celota, smo pripravili glavni modul, kateri implementira vse prej opisane razrede in povezuje funkcionalnosti posameznih razredov. To lahko vidimo na začetku modula, saj se tam po navadi nahajajo vsi uvozi modulov oziroma v našem primeru razredov. Sam opis bo strukturiran podobno kot vsak posamezen razred, le da bom tukaj priložil slike celotne skripte in jih opisal oziroma razložil. Ob kodi so napisani tudi komentarji, kar bi moralo pomagati pri razumevanju. Med vso kodo so raztrošeni tudi klici pomožnega razreda Log za zapisovanje dnevniških zapisov, ki zapisujejo, kaj se je dogajalo v skripti.

Modul po uvozu razredov ustvari instanco razreda SIEMAPICommunication, s katerim poizkuša vzpostaviti komunikacijo s SIEM in izvesti prvi iskalni ukaz za pridobitev podatkov, ki so že vpisani v iskalno tabeli. Rezultate shrani v lokalno spremenljivko, ki jo kasneje preveri, če vsebuje podatke. V kolikor v spremenljivki ni podatkov, modul le-to obravnava, kot da nima povezave s SIEM-om in zato preneha z nadaljnjim izvajanjem. Če dobi rezultate iskalnega ukaza, s pomočjo funkcije za varnostne kopije, ustvari lokalno datoteko z vsebino iskalne tabele pred zapisom novih vrednosti vanjo.

```
from REST.log import Log
from REST.exclusions import Exclusions
from REST.OpenCTIFeed import OpenCTIFeed
from REST.SIEMAPI import SIEMAPICommunication

# Ustvari povezavo s SIEM serverjem
Log.writeLog("Script initiated")
SIEMAPI = SIEMAPICommunication()
SIEMAPI.createConnection()

# Search zahtevek za vsebino lookup tabele
lookupTableContents = SIEMAPI.searchQuery("| getlookupdata " + SIEMAPI.config['lookupTableName'])

# Če komunikacija ni uspela se program samo zaključi
if lookupTableContents == []:
    Log.writeLog('Query didn\'t get any results')
    exit()
# Ustvari lokalni backup v cwd (current working direktory) z imenom backup.csv
SIEMAPI.writeCSV(lookupTableContents)
```

Slika 18: Povezava s SIEM v glavni skripti

Vir: (Lasten vir)

Po iteraciji čez podatke moramo iskalni ukaz izvesti še enkrat, saj podatkov ne moremo več prebrati oziroma iterirati čez njih. Nato naredi instanco pomožnega razreda za izključitev vrednosti, kar napolni seznam, ki se nahaja v razredu. Potem ustvari še instanco razreda za OpenCTI komunikacijo in pokliče funkcijo za primerjanje in združevanje virov, v kateri se

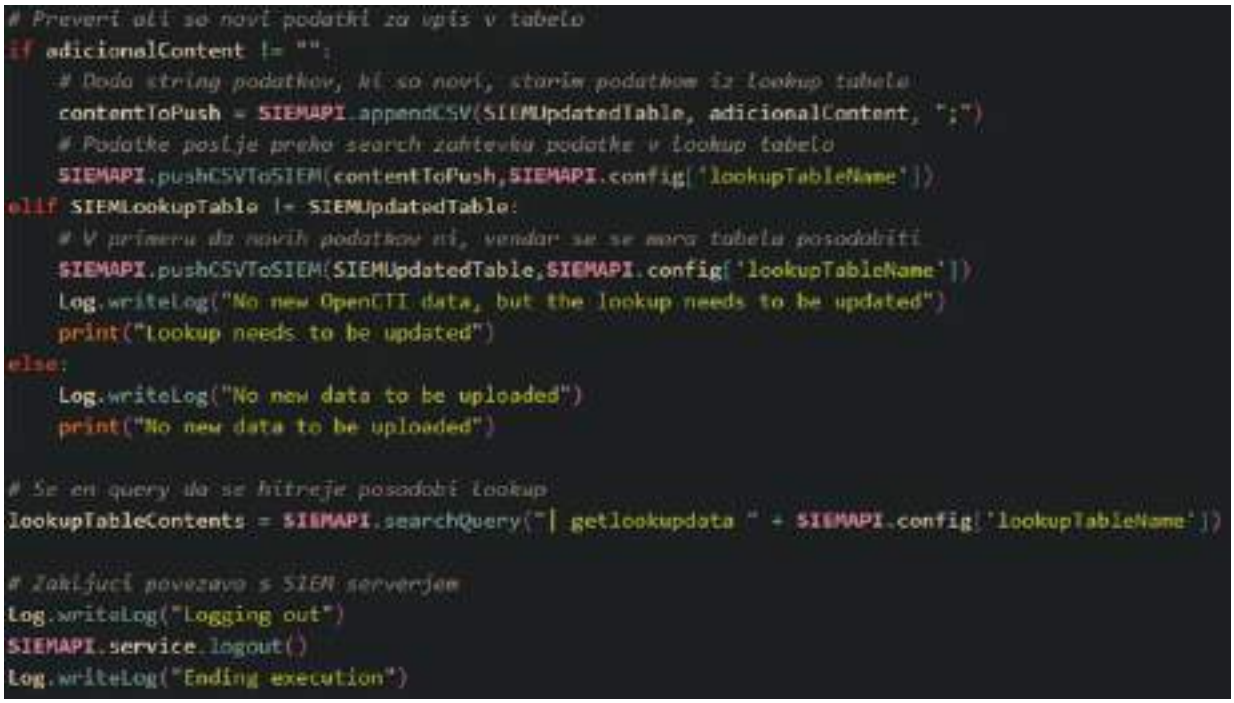
dvakrat kliče funkcija za pridobivanje podatkov vira, enkrat za vsak vir. Podatki, ki jih funkcija vrne, so seznam zaglavja podatkov in po njih urejen seznam podatkov, ki jih modul shrani v lokalne spremenljivke za kasnejšo uporabo. Nad prej izvedenimi rezultati izvede funkcijo za razčlenjevanje podatkov iz SIEM JSON podatkov z zastavico za generacijo zaglavja na true, saj želimo, da zgenerira tudi *regex*, ki ga bomo potrebovali za pošiljanje podatkov. Zadnji ukaz v tem odseku kode je ukaz za primerjavo OpenCTI podatkov z že zapisanimi podatki v iskalni tabeli SIEM-a. Funkcija vrne dve vrednosti, ki ju shranimo, saj nam funkcija vrne nove podatke, potrebne za zapis ali spremenjene podatke iskalne tabele.

```
# Še enkrat zahteva podatke iz lookup tabele, ker po enem iteriranju cez rezultate se jih ne more dostopati vec
lookupTableContents = SIEMAPI.searchQuery("| getlookupdata " + SIEMAPI.config['lookupTableName'])
if lookupTableContents == []:
    Log.writeLog('Query didn\'t get any results')
    exit()

# Igenerira seznam IoC-jev, ki jih zelimo zavreci
exclusion = Exclusions()

# Vzpostavi povezavo do OpenCTI feed-ov
Log.writeLog("Establishing OpenCTI connection")
openCTIdata = OpenCTIFeed()
# Podatke dobi iz feed-ov in jih sparsea
header, content = openCTIdata.compareCSV()

# Parsea podatke iz SIEM lookup tabele v uporabne podatke za skripto
SIEMLookupTable = SIEMAPI.parseJSONIntoCSVString(dataJSON=lookupTableContents, delimiter=";", includeHeaders=True)

# Primerja podatke dobljene iz SIEMa in OpenCTI-ja in dobi string novih podatkov
Log.writeLog("Comparing new data versus old data")
adicionalContent, SIEMUpdatedTable = SIEMAPI.compareCSV(SIEMLookupTable, header, content, ";")
```

Slika 19: Povezava z OpenCTI in primerjava podatkov

Vir: (Lasten vir)

Pred zapisom v iskalno tabelo se modul, glede na dobljene podatke zadnje klicane funkcije odloči, kaj bo storil. Glede na podatke preveri, ali je kaj novih podatkov iz OpenCTI-ja za vpis v iskalno tabelo. V kolikor so, združi CSV besedili iskalne tabele in podatkov za vpis iz OpenCTI, ter pokliče funkcijo za vpis z združenimi podatki. Če novih podatkov ni, vendar so se spremenili podatki, ki so že vpisani, modul izvede ukaz zapisa v iskalno tabelo z novimi podatki. Če novih podatkov ni in je iskalna tabela ostala enaka, se ne zgodi nič, le ustvari se dnevniški zapis, da ni bilo novih podatkov. Po končani vejitvi skripte se še enkrat izvrši iskalni ukaz za pridobitev podatkov s SIEM iskalne tabele, da sesami podatki iskalne tabele posodobijo za uporabo na SIEM-u oziroma, da se posodobi predpomnilnik (angl. cache). Na koncu se še odjavi oziroma prekine povezavo s SIEM in zapiše zadnji dnevniški zapis, ki pove, da se je skripta prenehala izvajati.



Slika 20: Zapis podatkov in zaključek glavne skripte

Vir: (Lasten vir)

7.7 Premik v produkcijsko okolje

Modul je opravil obsežno testiranje v nadzorovanem testnem okolju, ki je potrdilo njegovo splošno funkcionalnost in nemoteno delovanje. Po uspešnem testiranju je modul brez težav prešel v robustno produkcijsko okolje. V tem okolju modul trenutno deluje v virtualnem okolju Linux, ki harmonično sobiva s platformo OpenCTI.

Za zagotavljanje doslednega izvajanja skripte modula je bil vzpostavljen zanesljiv mehanizem. S pomočjo zmogljivosti periodičnih opravil (angl. cronjob) v operacijskem sistemu Linux je izvajanje modula zagotovljeno tako, da se izvaja v vnaprej določenih časovnih presledkih, kar zagotavlja njegovo neprekinjeno in pravočasno izvajanje. Ti cronjob-i služijo kot temelj načrtovanih opravil, ki modulu omogočajo, da sistematično preverja, ali so na viru novi razpoložljivi podatki, ki lahko obogatijo lookup tabelo.

Trenutno modul postopek preverjanja podatkov izvaja vsako uro. To stalno delovanje je brezhibno usklajeno s splošnimi cilji sistema in prispeva k njegovi splošni učinkovitosti ter zanesljivosti. Z reševanjem tega problema sem se srečal z zaledjem SIEM-a in OpenCTI-ja, od česar sem se posledično veliko naučil in dobil dodatno razumevanje uporabe obeh platform. Poleg razvoja modula sem pisal tehnično dokumentacijo in navodila za uporabo, kar je od mene

zahtevalo tudi druge veščine, kot na primer razlaganje programske kode na način, ki je razumljiv drugim, ki mogoče ne znajo programirati oziroma ne vedo, kako modul deluje v ozadju. Spoznal sem se tudi s celotnim življenjskim ciklom programiranja rešitve, ki vključuje raziskovanje možnosti, branje dokumentacije, razhroščevanje in iterativno razvijanje rešitve.

7.8 Izboljšave obstoječe rešitve

Razvita rešitev je bila ekstenzivno testirana v produkcijskem okolju, kjer se periodično izvaja že od začetnega testiranja. Med izvajanjem sem redno spremljal dogajanje s pomočjo v modul vgrajenega sistema zapisovanja poteka dogajanja, s katerim sem ugotovil nekatere pomanjkljivosti modula. Največ problemov se je pojavilo s podatki, ki jih modul sprejme kot vhodne parametre. Ti podatki so vrednosti, zapisane v OpenCTI, ki kot smo obravnavali na začetku, vase prejmejo zelo raznolike podatke. To je sicer odlično za sistem, saj pomeni, da je skupno stičišče raznoraznih virov, vendar iz vidika našega priključka to pomeni zgolj težave. Med izvajanjem sem opazil, da se modul ustavi pri obdelovanju podatkov, ki jih normalizira v obliko, ki jo naš SIEM zna interpretirati oziroma je za njegovo delovanje uporabna. Dogajanje bom še naprej spremljal, vendar se zavedam, da bi bilo potrebno funkcijo, ki podatke razčlenjuje ponovno napisati oziroma vanjo dodati mehanizem, ki bi zaznaval napake in se varno ognil potencialnim sproženim izjemam, ki ustavijo delovanje modula. V kolikor pride do take izjeme, se delovanje modula prekine, vendar to ne pomeni, da se podatki ne prenesejo, saj takoj ko podatki postanejo dovolj stari, da se ne pokažejo več na OpenCTI viru, lahko modul normalno prenese podatke. Kljub temu to ni dobra rešitev in bi jo lahko izboljšal.

Pri velikem prenosu podatkov v OpenCTI se tudi vir tovrstnih podatkov zelo podaljša, kar pomeni veliko več vhodnih podatkov za modul. Ko sem testno vnašal ogromne podatke, sem ugotovil, da je modul potreboval veliko več časa za izvajanje. Po ročnem zagonu sem ugotovil, da modul ne preneha z delovanjem, vendar da potrebuje le zelo veliko časa za razčlenjevanje podatkov, saj ima v sebi zelo veliko zank, ki povečajo časovno zahtevnost in s tem tudi čas delovanja modula. Ob ročnem zaganjanju sem se lotil razhroščevanja s pomočjo za to namenjenega orodja v izbranem IDE-ju. Postavil sem si točke prekinitve na ključne dele izvajanja modula. To mi je močno olajšalo iskanje problema, saj sem se s pomočjo le-teh lahko premikal po izvajanju modula. Odkril sem, da je problem ležal v isti funkciji, kot je omenjena v zgornjem problemu.

Funkcijo sem ob nastalih odkritih slabostih že večkrat predelal in poskusil izboljšati, vendar domnevam, da problem ne leži zgolj v funkciji, temveč tudi v programskem jeziku, ki sem ga izbral za razvoj modula. Python ni znan po hitrosti, ampak bolj po svoji prilagodljivosti, ki omogoča hiter razvoj rešitev. Kot morebitno izboljšavo bi lahko vsaj ta del modula prepisal v kakšen drug programski jezik, ter shranil v CSV datoteko, kjer bi ga lahko prebral z že obstoječim modulom. To je ena izmed možnih rešitev problema, ki ne bi zahtevala prevelikega časovnega vložka v raziskavo oziroma spoznavanje drugega programskega jezika. Druga rešitev bi lahko bila tudi prepis celotnega modula v drug programski jezik, vendar lahko ta rešitev za seboj potegne tudi druge probleme, saj bi bilo potrebno nekatere dele kode prilagoditi in upoštevati sintakso drugih programskih jezikov.

Poleg tega sem ugotovil, da je bil sistem zapisovanja delovanja modula zelo dobra ideja, vendar sem poleg tega ugotovil tudi nekaj pomanjkljivosti, ki so prišle z načinom implementacije. V konfiguracijski datoteki bi lahko dodal dodatne spremenljivke, ki bi konfigurirale mesto, kjer se nahaja datoteka z zapisom dogajanja in dodatno spremenljivko, ki bi modulu povedala, koliko stare zapise delovanja lahko briše za seboj, saj se v trenutnem sistemu le-ta ne pobriše in se samo veča. Poleg tega je problem tudi v redkosti sporočil, saj nenatančno obveščajo o dogajanju in bi lahko bili bolj pogosto napisani v kodi. Tudi zapisovanje sporočil bi lahko bilo pogojeno s tem, da je šlo nekaj narobe v modulu ali pa bi le-ta obveščal bolj o tem, koliko podatkov se ni preneslo zaradi problemov z razčlenjevanjem.

8 SKLEP

Diplomska naloga je zahtevala poglobljeno razumevanje različnih vidikov na področju kibernetske varnosti, ki ne zajema le zapletenih odtenkov programiranja, temveč tudi natančne zapletenosti raziskovanja dokumentacije. Z natančnim poglabljanjem v vrstice kode in vestnim prebiranjem ustaljenih virov sem se lotil ključne naloge natančnega preverjanja in posledično potrjevanja ali zavračanja določenih hipotez. Ta celoviti pristop ni le izpopolnil mojih analitičnih spretnosti, temveč je tudi utrdil moje strokovno znanje in izkušnje pri krmarjenju po večplastnem področju kibernetske varnosti.

Prvo hipotezo H1: Ali lahko rečemo, da je Security Onion najboljša SIEM rešitev? sem zavrgel na podlagi dejstev iz poglavja 2 in 2.1, ker sem po raziskavi odkril, da vsaj z vidika malega podjetja obstaja boljša rešitev, in sicer predaja kibernetske varnosti zunanjemu izvajalcu.

Drugo hipotezo H2: OpenCTI lahko prenese velike količine podatkov o grožnjah in jih smiselno korelira med seboj, sem potrdil v poglavju 3.3, saj količina podatkov, ki sem jih vnašal v OpenCTI ni bila majhna. Kljub temu je bila aplikacija po končanem uvozu zelo odziva in je smiselno kolorirala podatke med seboj.

Tretjo hipotezo H3: Povezava SIEM rešitve z OpenCTI že obstaja, ki predvideva, da je bila rešitev za zapolnitev vrzeli med OpenCTI in SIEM že razvita, sem zavrgel v 4. poglavju na podlagi raziskave obstoječih rešitev, ki niso bile kompatibilne z našo SIEM rešitvijo. Že razviti priključki so bili za druge SIEM rešitve, ki niso kompatibilni s to, ki jo uporabljamo v našem podjetju, zato sem se razvoja priključka lotil sam in ga uspešno razvil.

Četrto hipotezo H4: Obstoječo oziroma razvito rešitev lahko predelamo tako, da bo podpirala neskončen tok podatkov (krovna tema), ki predvideva, da lahko priključek predelam tako, da bo podpiral neskončen tok podatkov, sem potrdil v poglavju 6., saj je med razvojem priključek deloval na tem principu. V neskončni zanki so se podatki primerjali s podatki na SIEM in se sproti pošiljali, vendar so bile specifikacije delovanja modula drugačne, zato sem ga predelal v obliko, ki je bila predstavljena v poglavju 7. in njegovih podpoglavjih.

Uspešen razvoj in uvedba priključka OpenCTI za naš sistem za upravljanje varnostnih informacij in dogodkov (SIEM) pomenita pomemben mejnik pri izboljšanju kibernetske varnosti naše organizacije. S skrbnimi raziskavami, natančnim kodiranjem in testiranjem ta projekt ni odpravil le obstoječih omejitev, temveč je tudi precedens za prihodnji napredek pri vključevanju obveščevalnih podatkov o grožnjah.

Vzpostavitev priključka OpenCTI je dokaz moči inovacij, sodelovanja in prilagodljivosti v naši organizaciji. S premostitvijo vrzeli med našim sistemom SIEM in platformo OpenCTI, smo naši ekipi za kibernetsko varnost omogočili dinamičen nabor orodij, ki lahko združuje, povezuje in analizira podatke o grožnjah iz številnih virov. To bo omogočilo hitrejše odkrivanje, bolj premišljeno sprejemanje odločitev in na koncu bolj odporno obrambo pred vedno novimi kibernetskimi grožnjami.

Pri napredku se je treba zavedati, da se tehnologija nenehno spreminja. Področje groženj se bo še naprej razvijalo, zato se morajo razvijati tudi ukrepi za kibernetsko varnost. Uspeh tega projekta poudarja, kako pomembno je, da ostanemo proaktivni in agilni pri svojem pristopu k varovanju občutljivih podatkov in kritičnih sistemov. Redne posodobitve in izboljšave priključka OpenCTI bodo zagotovile, da bomo ostali v ospredju na področju vključevanja obveščevalnih podatkov o grožnjah in se dosledno prilagajali nastajajočim grožnjam ter najboljšim industrijskim praksam.

Ta diplomski projekt ne prispeva le k zbirki znanja na področju kibernetske varnosti, temveč prikazuje tudi zavezanost naše organizacije k odličnosti in inovativnosti. Prepričan sem, da bo imel priključek OpenCTI ključno vlogo pri krepitvi naše kibernetske obrambe in spodbujanju kulture nenehnih izboljšav v naši organizaciji.

Na koncu bi se rad zahvalil vsem, ki so prispevali k uspehu tega projekta. Od mentorjev in sodelavcev, ki so zagotovili smernice in spoznanja, do celotne ekipe za kibernetsko varnost, ki bo to orodje uporabljala za zaščito naših digitalnih sredstev in je ta dosežek dokaz naše skupne predanosti in strokovnega znanja. Naj ta projekt služi kot odskočna deska za nadaljnji napredek na področju kibernetske varnosti in navdihuje prihodnja prizadevanja, ki premikajo meje tehnoloških inovacij.

9 VIRI

CrowdStrike. (27. Julj 2023). *What is Threat Intelligence?* Pridobljeno iz CyberSecurity 101 - Threat Intelligence: https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/

Filigran. (27. Julij 2023). *OpenCTI connectors GitHub repository*. Pridobljeno iz OpenCTI connectors GitHub repository: https://github.com/OpenCTI-Platform/connectors

Filigran. (7. Avgust 2023). *OpenCTI documentation*. Pridobljeno iz OpenCTI installation: https://docs.opencti.io/latest/deployment/installation/

Filigran. (27. Julij 2023). *OpenCTI Ecosystem*. Pridobljeno iz OpenCTI Ecosystem: https://filigran.notion.site/OpenCTI-Ecosystem-868329e9fb734fca89692b2ed6087e76

Filigran. (27. Julij 2023). *OpenCTI Python Client*. Pridobljeno iz OpenCTI Python Client GitHub repository: https://github.com/OpenCTI-Platform/client-python

IBM QRadar. (7. Avgust 2023). *IBM QRadar SIEM*. Pridobljeno iz IBM QRadar SIEM: https://www.ibm.com/products/qradar-siem/pricing

Matthes, E. (2016). *Python Crash Course.* Alaska: No Starch Press.

Microsoft. (27. Julij 2023). *What is SIEM?* Pridobljeno iz SIEM Defined: https://www.microsoft.com/en-us/security/business/security-101/what-is-siem

Nginx. (20. Avgust 2023). *Nginx documentation*. Pridobljeno iz Nginx documentation: https://nginx.org/en/docs/

OASIS. (27. Julij 2023). *Introduction to STIX*. Pridobljeno iz Introduction to STIX: https://oasis-open.github.io/cti-documentation/stix/intro.html

Security Onion Solutions. (19. Julij 2023). *Security Onion Solutions*. Pridobljeno iz Security Onion Solutions: https://securityonionsolutions.com

Telekom Slovenija. (7. Avgust 2023). *Varen splet*. Pridobljeno iz Varen splet - Ponudba storitve: https://www.telekom.si/poslovni-uporabniki/ponudba/internet/storitve/varen-splet

10 SEZNAM UPORABLJENIH KRATIC

angl. – angleško

API – (po angl. Application Programming Interface); programski vmesnik aplikacije

ATT&CK – (angl. Adversarial Tactics, Techniques, and Common Knowledge); nasprotnikove taktike, tehnike in splošno znanje

C2C – (angl. command and control); ukazovalni in nadzorni strežnik, tipično se pojavi v škodljivi programski kodi, ki je namenjena za oddaljen dostop do okužene naprave

CSV – (angl. Comma Separated Values); podatki ločeni z vejico

CTI – (angl. Cyber Threat Intelligence); podatki o grožnjah v kibernetski domeni

CWD – (angl. Current Working Directory); trenutni delaven direktorij

HTML – (angl. Hyper Text Markup Language); programski jezik, v katerem so napisane spletne strani

HTTP – (angl. Hypertext Transfer Protocol); protokol za prenos informacij na spletu

HTTPS – (angl. Hypertext Transfer Protocol Secure); nadgrajena različica HTTP protokola, s poudarkom na varnost

IDE – (angl. Integrated Development Environment); integrirano urejevalno okolje

IOC – (angl. Idicator of compromise); indikator kompromisa

JSON – (angl. JavaScript Object Notation); oblika objektnega zapisa v programskem jeziku JavaScript

OSINT – (angl. Open-source intelligence); informacije iz javnih virov

regex – (angl. Regular expression); regularni izraz

SIEM – (angl. Security Information and Event Management); sistem za upravljanje varnostnih informacij in dogodkov

SOC – (angl. Security Operations Center); Varnostno-Operativni Center - VOC

STIX – (angl. Structured Threat Information Expression); format za izražanje strukturiranih informacij o grožnjah

TAXII – (angl. Trusted Automated eXchange of Intelligence Information); zaupanja vredna avtomatizirana izmenjava obveščevalnih informacij

XML – (angl. Extensible Markup Language); razširljiv označevalni jezik