SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-8906

Detekcia správania programov

DIPLOMOVÁ PRÁCA

2017 **Bc. Peter Vaš**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-8906

Detekcia správania programov

DIPLOMOVÁ PRÁCA

Študijný program :	Aplikovaná informatika
Číslo študijného odboru:	2511
Názov študijného odboru:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	Ing. Štefan Balogh, PhD .
Konzultant ak bol určený:	

2017

Bc. Peter Vaš

Zadanie

Slovenská technická univerzita v Bratislave
Ústav informatiky a matematiky

Fakulta elektrotechniky a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Autor práce:	Bc. Peter Vaš
Študijný program:	aplikovaná informatika
Študijný odbor:	9.2.9. aplikovaná informatika
Evidenčné číslo:	FEI-5384-8906
ID študenta:	8906
Vedúci práce:	Ing. Štefan Balogh, PhD.
Miesto vypracovania:	Ústav informatiky a matematiky
Názov práce:	**Detekcia správania programov**
Jazyk, v ktorom sa práca vypracuje:	slovenský jazyk
Špecifikácia zadania:	Analýza programov sa v súčasnosti vykonáva rôznymi prístupmi. Využívajú sa rôzne metódy analýzy programov. K vyhodnocovaniu výstupov analýzy je potrebné pri automatizácii procesu analýzy nájsť vhodný systém rozhodovania. Jednou z možností je využitie ontológie. Vstupné dáta je v tomto prípade nutné previesť do ontologického modelu. Navrhnite a implementujte systém pre rozhodovanie s využitím ontológie pre vybraný systém na analýzu programov. Úlohy: 1. Naštudujte možné prístupy pre transformáciu získaných poznatkov do ontologického modelu. 2. Navrhnite vlastný postup transformácie a spracovania dát, pre potreby systému pre rozhodovanie. 3. Implementujte navrhnuté riešenie a vyhodnoťte jej využiteľnosť pre analýzu programov so zameraním na detekciu správania a chýb programov.

Literatúra:

- Parmelee, Mary C. "Toward an Ontology Architecture for Cyber-Security Standards." STIDS 713 (2010): 116-123.
- Syed, Zareen, et al. "UCO: A unified cybersecurity ontology." Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security. AAAI Press, 2016.

Dátum zadania:	19. 09. 2016
Dátum odovzdania:	19. 05. 2017

Bc. Peter Vaš
študent

prof. RNDr. Otokar Grošek, PhD.
vedúci pracoviska

prof. Dr. Ing. Miloš Oravec
garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program :	Aplikovaná informatika
Diplomová práca:	Detekcia správania programov
Autor:	Bc. Peter Vaš
Vedúci záverečnej práce:	Ing. Štefan Balogh, PhD.
Konzultant ak bol určený:	
Miesto a rok predloženia práce:	Bratislava 2017

V tejto diplomovej práci sa zaoberáme problematikou využitia ontológii v oblasti kybernetickej bezpečnosti. V úvode si povieme niečo o tom, čo sú to kyber bezpečnostné štandardy, na čo sa využívajú a akú rolu zohrávajú v našej práci. V ďalšej podkapitole si vysvetlíme čo je to ontológia a ako sa využíva na internete. Ďalej preskúmame potenciál ontológie v oblasti kyber bezpečnosti a povieme si niečo o tom, prečo by ontológia mohla spôsobiť revolúciu v oblasti kyber obrany. Hlavným cieľom tejto práce je zhrnutie dostupných informácii z týchto oblasti do jedného celku a následné vytvorenie programu, ktorý dokáže mapovať dáta zapísané vo formátoch kyber bezpečnostných štandardov do ontológie. Ciele sa nám podarilo splniť len čiastočne a výsledkom je programové riešenie schopné načítavať a upravovať vstupné XML súbory zapísané podľa štandardov. Toto riešenie je tiež schopné pracovať s ontológiou pomocou SPARQL dotazov avšak mapovanie medzi XML a ontológiou sa nám nepodarilo implementovať.

Kľúčové slová: kybernetická bezpečnosť, ontológia, bezpečnostný štandard

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Diploma Thesis:	Detection of program behaviour
Autor:	Bc. Peter Vaš
Supervisor:	Ing. Štefan Balogh, PhD.
Consultant:	
Place and year of submission:	Bratislava 2017

This master thesis is focused on the use of ontology in the area of cyber security. In the introduction we will make an overview of standards in cybersecurity and we will discuss their role in our thesis.

In the next chapter we will introduce the concept of ontology and its use in the world wide web. Next we will examine the potential of ontology in the area of cybersecurity and we will speak about its features that could revolutionize the domain of cyber defense. The main objective of this thesis is to summarize available information from these areas into one whole and to create a program, that could map the data written according to cyber security standards into an ontology.

The objectives of this thesis were partly met and the result of it is program that can load and parse XML files written according to cyber standards and it can also work with ontology using SPARQL queries, but it cannot map the data XML into ontology.

Key words: cybersecurity, ontology, cybersecurity standard

Vyhlásenie autora

Podpísaný Peter Vaš čestne vyhlasujem, že som Diplomovú prácu Detekcia správania programov vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Uvedenú prácu som vypracoval pod vedením Ing. Štefana Balogha, PhD.

V Bratislave dňa 26.05.2017

.....

podpis autora

Poďakovanie

Touto cestou by som sa chcel poďakovať pánovi Ing. Štefanovi Baloghovi, PhD. za pomoc, odborné vedenie a cenné rady pri vypracovaní mojej diplomovej práce.

Obsah

Zoznam obrázkov a tabuliek

Ak máte veľa obrázkov a tabuliek, rozdeľte tabuľku na dve samostatné.

Zoznam skratiek a značiek

API (Application programming interface) - množina jasne definovaných metód komunikácie medzi rôznymi softvérovými komponentmi.

DARPA(Defense Advanced Research Projects Agency) – agentúra ministerstva obrany US pre výskum pokročilých obranných projektov.

Slovník

Cryptolocker - malvér šifrujúci dáta na disku. Tvorca malvéru väčšinou poskytuje odšifrovanie týchto súborov za určité "výkupné".
Exploit – špeciálny program alebo sekvencia príkazov, slúžiaca na vniknutie do systému vďaka zraniteľnosti.
Exploitácia – činnosť pri ktorej útočník využíva exploity na prienik do systému.
Kernel (jadro) – centrálny komponent operačného systému, ktorý zabezpečuje programom prístup k hardvéru.
Kyber bezpečnosť (z angl. cyber security) - alebo tiež kybernetická bezpečnosť, počítačová bezpečnosť, je oblasť IT, ktorá sa zaoberá ochranou počítačových systémov pred krádežou, poškodením alebo neoprávneným prístupom k hardvéru, softvéru alebo informáciam.
Malvér (z angl. malware) - škodlivý softvér.
pip - balíčkový systém pre programovací jazyk python.
Plugin - softvérový komponent, ktorý pridáva špecifickú vlastnosť do už existujúceho softvérového systému.
Ransomvér (z angl. ransomware) - malvér, požadujúci výkupné. Väčšinou sa jedná o formu cryptolockeru.
Rootkit – malvér, ktorý je schopný sa maskovať pred detekciou.
Signatúra – podpis, alebo jednoznačný identifikátor.
Útočný vzor (attack pattern) – ide o abstrakciu spôsobov útoku, podobné ako návrhové vzory v programovaní.
Zero day zraniteľnosť – zraniteľnosť, o ktorej vývojári daného produktu nevedia, a preto zostáva neopravená. Takéto zraniteľnosti sú najnebezpečnejšie, pretože každý, kto využíva daný produkt je potenciálne ohrozený touto zraniteľnosťou.

Úvod

Od začiatku tohto storočia sa množstvo zariadení pripojených k internetu niekoľkonásobne zväčšilo. Dnes je na internet pripojené obrovské množstvo zariadení siahajúcich od realtime systémov cez klasické počítače a mobily až po priemyselné kamery a domáce spotrebiče. Internet im umožňuje navzájom komunikovať a umožňuje užívateľom ich monitorovanie a ovládanie. Nastáva tu však veľký problém. Internet nikdy nebol navrhnutý ako bezpečná sieť pre milióny zariadení. Pôvodne slúžil na vojenské účely, pričom sa nepredpokladalo jeho masívne rozšírenie na nezabezpečené zariadenia. Jednou z najväčších hrozieb pre tieto zariadenia je malvér. Našťastie však existujú na trhu špecializované nástroje slúžiace na detekciu a odstránenie malvéru ako napríklad rôzne antivírusové produkty, avšak bežný antivírus dokáže detegovať a odstrániť iba niečo okolo 40% nového malvéru. Toto je spôsobené zastaraným spôsobom klasifikácie malvéru do rôznych rodín podľa určitých špecifických vlastností pričom sa malvér delí na vírusy, ktoré napádajú iné súbory, červy, ktoré sa dokážu šíriť cez sieť, trójske kone, ktoré sú zamaskované v iných programoch atď. Malvér v dnešnej dobe však má vlastnosti hneď niekoľkých rodín a takmer vždy obsahuje kód, ktorý má zabrániť alebo aspoň skomplikovať antivírusovým produktom v jeho detekcii. Z tohto dôvodu sa začína uvažovať nad ontologickými systémami, ktoré by dokázali detegovať malvér podľa toho ako sa správa. V teoretickej časti tejto práce si povieme niečo o tom čo je to ontológia a aký je aktuálny stav ontológie v bezpečnostných riešeniach. V praktickej časti práce sa budeme zaoberať možnosťou využitia ontológie pri práci s dátami z agentových systémoch a implementáciou takéhoto systému. Nakoniec v zhrnutí zhodnotíme celú prácu a jej výsledky.

1 Analýza problému

Čím ďalej tým viac počítačová bezpečnosť znepokojuje firmy a štátne inštitúcie. Uvedomujú si totiž, že hrozby na internete rastú a vďaka jednoduchým nástrojom dokáže aj nie veľmi skúsený útočník narobiť veľké škody. Na internete je totiž možné nájsť množstvo nástrojov určených na penetračné testovanie veľmi jednoducho a zadarmo. Média ako YouTube poskytujú veľké množstvo návodov využitia týchto nástrojov.

Okrem hackerských útokov je však ďalšou hrozbou aj malvér. Slovo malvér označuje škodlivý softvér ako napríklad vírusy, trójske kone, červy atď. Len počas tretieho kvartálu roka 2016 bolo zachytených 18 miliónov nových vzoriek malvéru. Veľkú popularitu počas roku 2016 získal ransomvér, teda malvér ktorý zašifruje údaje a výmenou za ich odšifrovanie požaduje výkupné, väčšinou v kryptomene Bitcoin. Podľa štatistiky FBI každý deň roka 2016 došlo v priemere k 4000 útokom tohto typu malvéru. Oproti roku 2015 teda nastal 300% nárast. Najčastejším spôsobom rozšírenia tohto malvéru bola infikovaná emailová príloha, odkaz na útočnú stránku alebo ako inštalačný súbor "tváriaci" sa ako oficiálny inštalátor softvéru ako je Java, Flash Player alebo Adobe Reader (5).

Situácia je však ešte horšia, podľa (2) prieskumy v roku 2014 ukázali, že až 65% firiem sa už stalo obeťou kyber útoku alebo elektronickej krádeže údajov. Väčšina týchto firiem bola upozornená na tieto útoky externou firmou, pričom detekcia trvala v priemere 13 mesiacov. Počas tejto doby mohlo dochádzať k únikom informácii z týchto firiem a tým ich znevýhodneniu na trhu.

Vo väčšine prípadov kyber útokov, o ktorých sa dozvedáme každý deň sa jedná o útoky jednotlivcov alebo menších skupín, ktoré napádajú počítačové systémy za účelom obohatenia sa alebo upozornenia na ich politické ciele. Tieto útoky nazývané tiež kyber kriminalita využívajú väčšinou základné nedostatky v zabezpečení systémov, ako napríklad neaktuálny softvér. Okrem týchto druhov kyber hrozieb však existujú ešte aj iné, o dosť nebezpečnejšie a ťažšie identifikovateľné hrozby na internete. Ide o útoky veľkých celkov ako sú napríklad konkurenčné spoločnosti alebo dokonca štáty.

Tieto boje väčšinou vznikajú v dôsledku snahy získania prevahy nad opozičnými štátmi alebo korporáciami či už na ekonomickej rovine alebo vojenskej. Cieľmi takýchto útokov bývajú strategické celky ako napríklad elektrárne, vojenské zariadenia alebo finančné inštitúcie ako napríklad burzy (4). Môže sa nám zdať, že tieto útoky sa nás nijak

nedotýkajú a sú vedené len medzi štátmi a veľkými organizáciami, to však nie je pravda. Na Ukrajine v roku 2016 hackeri z istého štátu vyradili počas Vianoc na niekoľko hodín z prevádzky elektrárne tým, že napadli počítače elektrárne. O elektrinu tak na Vianoce prišlo 225 000 objektov vrátane domácností. Po niekoľkých hodinách sa podarilo zamestnancom elektrárne obnoviť rozvod elektriny manuálnym ovládaním.

Tento útok mal síce žiadaný efekt, avšak nebol ničivý. Štátmi sponzorovaných útokov však existujú tisíce, o väčšine z nich sa však nedozvedáme.

Známymi, štátom sponzorovanými útokmi boli útoky malvéru Stuxnet, Duqu a Flame. Jednalo sa o špecializované, vysoko sofistikované malvéry, ktoré svetu ukázali, že bežná antivírusová ochrana zďaleka nie je postačujúca. Malvér Stuxnet mal za úlohu vyhľadať a zničiť zariadenia na obohacovanie uránu v Iráne. Podľa nepotvrdených informácii však niekto zmenil kód malvéru, aby sa správal agresívnejšie, a tak sa podarilo rozšíriť tento malvér do celého sveta. Stuxnet zostal nedetegovaný viac ako rok a malvér Flame zostal nedetegovaný dokonca viac ako 2 roky. Všetky dostupné antivírusové riešenia tento malvér nepovažovali za nebezpečenstvo, aj vďaka tomu, že obsahovali pravé bezpečnostné certifikáty, ktoré boli odcudzené z dôveryhodných IT firiem.

Ďalším problémom bolo, že využívali takzvané zero-day exploity, čo sú zraniteľnosti o ktorých vývojári softvéru, pre ktorý sú určené nevedia a nedokážu ich teda opraviť.

Jedným z možných riešení danej situácie by mohlo byť využitie ontológie v oblasti kyber bezpečnosti. V spojení s umelou inteligenciou by mohlo vzniknúť funkčné riešenie na boj proti sofistikovanému malvéru a práve preto sme sa rozhodli pre vypracovanie tejto práce.

V nasledujúcich kapitolách si postupne popíšeme čo sú to kyber bezpečnostné štandardy a načo slúžia, čo je to ontológia a ako sa dá využiť v oblasti kyber bezpečnosti.

1.1 Kyber bezpečnostné štandardy

V tejto časti si povieme niečo o tom čo sú to kyber bezpečnostné štandardy, načo slúžia, kde sa využívajú a spomenieme si niektoré známe štandardy, pričom s niektorými z nich budeme ďalej pracovať.

Na dnešnom trhu je dostupné množstvo bezpečnostných nástrojov a systémov pre počítače a iné výpočtové zariadenia, pričom každý z týchto nástrojov generuje učité dáta a informácie ako napríklad logy. Problémom je, že každý výrobca využíva iné štandardy a formáty na zápis týchto informácii. Okrem bezpečnostných nástrojov sú pre detekciu útokov dôležité aj logy z aplikácii a operačných systémov. Tu však situácia, nie je o nič lepšia. Napríklad v operačnom systéme GNU/Linux je väčšina logov či už systémových alebo aplikačných uložená vo /var/logs, avšak každý program si ukladá vlastný log, vo vlastnom formáte. Ako som už spomínal tieto informácie sú dôležité pri detekcii hrozieb alebo pri ich dokumentácii. Ak však každý nástroj využíva vlastný zápis, veľmi sa tým komplikuje strojové spracovanie a vyhodnocovanie, prípadne sa strácajú niektoré informácie.

Cieľom kyber bezpečnostných štandardov preto je využívanie jednotnej štruktúry zápisu dát, ktorá by zjednodušila ich ďalšie spracovanie napríklad na účely bezpečnostných auditov. Pre tento účel vzniklo niekoľko štandardov, ktoré budeme využívať aj v našej práci. Jedná sa o CybOX, MAEC a STIX. Každý z týchto štandardov slúži na popis určitej oblasti počítačového sveta. CybOX je najvšeobecnejší a dajú sa ním popísať aj subjekty, ktoré nesúvisia priamo s počítačovou bezpečnosťou. STIX a MAEC sú určené výhradne na popis bezpečnostných záležitostí. Všetky tieto štandardy vznikli pred niekoľkými rokmi a sú dostupné pod open-source licenciami, čím autori chcú dosiahnuť ich rozšírenie. Momentálne sú pod patronátom spoločnosti MITRE, ktorá sa stará o ich vývoj.

Tieto štandardy sa pomaly rozširujú do bezpečnostných spoločností a preto je už dnes možné stiahnuť si dáta popisujúce malvér alebo bezpečnostné incidenty zo stránok niektorých spoločností zaoberajúcich sa počítačovou bezpečnosťou. Napríklad MAEC obsahuje plugin na získanie informácii zo stránky VirusTotal a ich konverziu na MAEC xml. Jedným zo známejších nástrojov podporujúcich MAEC štandard je aj softvér na izoláciu a dynamickú analýzu malvéru zvaný Cuckoo. Tento nástroj dokáže exportovať všetky získané informácie ako MAEC XML súbor.

Všetky tieto štandardy sa zapisujú vo forme XML súboru, ale je možné ich zapísať aj vo forme JSON. Na github stránkach týchto štandardov je možné nájsť aj prídavné pluginy, ktoré umožňujú zápis v jazykoch, ako je napríklad html alebo umožňujú konverziu na iné štandardy.

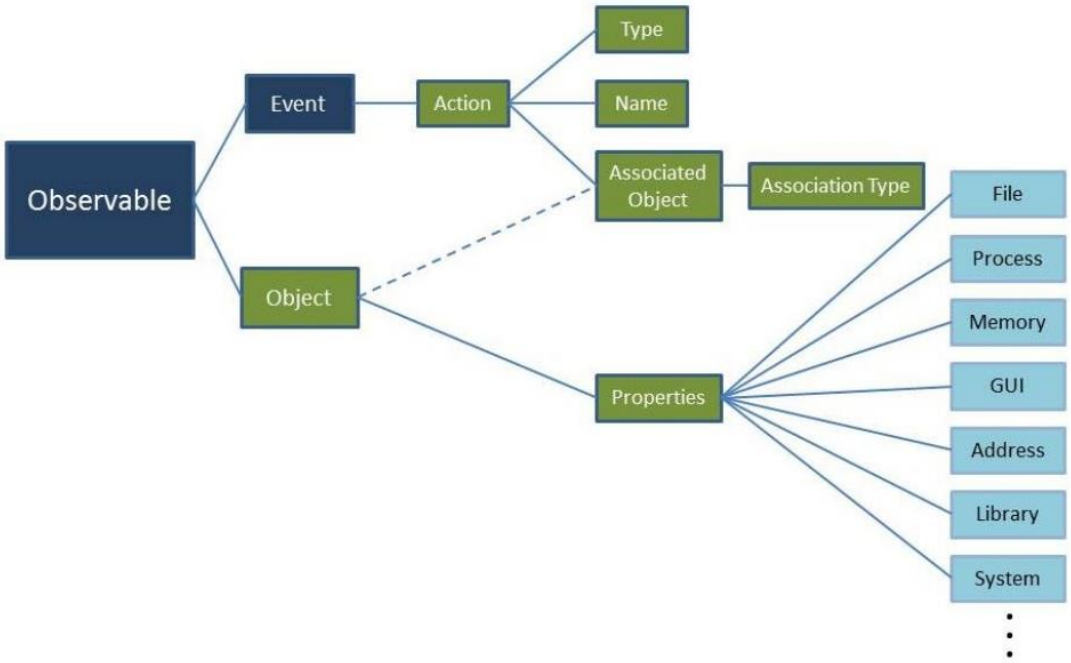
V nasledujúcich kapitolách si bližšie popíšeme jednotlivé štandardy.

1.1.1 CybOX (Cyber Observable eXpression)

CybOX je založený na koncepte takzvaných pozorovateľných udalostí (cyber observables), ktoré vlastne predstavujú udalosti alebo stavové vlastnosti v počítačových systémoch, ako napríklad zmena hodnoty registra, vymazanie súboru, http request, hash reťazec, atď. Ako je možné vidieť na týchto príkladoch, nie je zameraný na konkrétnu oblasť, ale je vytvorený tak, aby ním bolo možné popísať čo najširšiu oblasť. Vďaka tejto flexibilite, je pomocou neho možné popísať nielen konkrétne udalosti, ktoré sa dajú popísať alebo zmerať, ale zároveň sa ním popisujú aj abstraktné vzory správania.

Pred vznikom CybOX-u neexistoval žiaden štandardný mechanizmus slúžiaci na špecifikáciu, zachytávanie, popis alebo výmenu týchto udalostí, a preto každý tvorca softvéru využíval vlastný prístup. Implementáciou štandardizovaného jazyka CybOX do softvérových nástrojov by teda bolo možné zjednodušiť zdieľanie dát medzi rôznymi nástrojmi a organizáciami.

Hlavným účelom CybOX-u je jeho využitie pri popise malvéru, bezpečnostných operáciách, logovaní, reakcií na incidenty a forenznej analýze. Okrem CybOX nie je veľmi široko využívaný, avšak slúži ako základ pre jazyky STIX, MAEC, CAPEX a DFXML ktoré sú využívané častejšie a o ktorých si ešte niečo povieme.



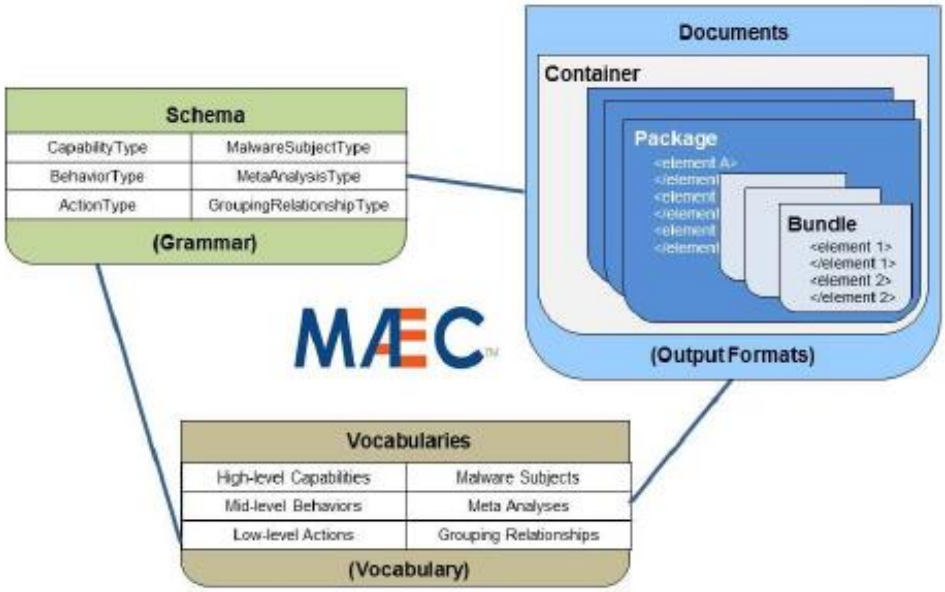
Obr. 01 CybOX schéma.

1.2 MAEC (Malware Attribute Enumeration and Characterization)

Malvér existuje už od roku 1971, kedy vznikol prvý počítačový vírus. Dnes je malvér zodpovedný za množstvo aktivít spojených s kyber kriminalitou siahajúcou od rozposielania spamu, cez botnety až po krádeže údajov a sociálne inžinierstvo (7). Či už sú útočníkmi tzv. script kiddies (útočníci využívajúci dostupné nástroje bez hlbších vedomostí), aktivisti, kriminálnici alebo štátmi podporované organizácie, malvér má vždy za úlohu negatívne ovplyvniť obeť, alebo získať prístup k jej dátam. Na krádež údajov stačí jeden program avšak na získanie dobrej reputácie firmy treba roky námahy a aj preto je ochrana pred malvérom jednou z najdôležitejších oblastí počítačovej obrany. Dokonca ani systémy, ktoré nie sú pripojené k počítačovým sieťam nie sú úplne v bezpečí, keďže malvér sa na ne môže dostať prostredníctvom USB kľúča alebo iného prenosového média.

Momentálne sa na detekciu malvéru v antivírusových nástrojoch využíva niekoľko klasických spôsobov, zväčša založených na signatúrach malvéru a na heuristike. Tieto spôsoby sú pomerne jednoduché a účinné, ale len v niektorých prípadoch. Systémy postavené na rozpoznávaní signatúr, napríklad nie sú vhodné na detekciu útokov pomocou zero day zraniteľností, cielených útokov, polymorfných malvérov a podobne. Nedostatkom heuristických systémov je schopnosť detegovať iba malvér správajúci sa podľa určitých vzorov správania, ktoré poznajú a preto nedokážu detegovať malvér ako kernelové rootkity. Z toho dôvodu tieto metódy samostatne nie sú dostatočné na detekciu všetkých druhov malvéru.

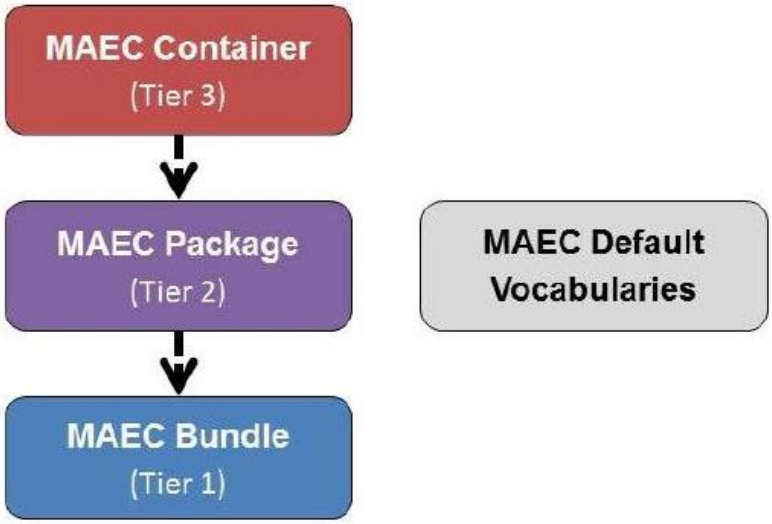
Aj toto je jeden z dôvodov, prečo vznikol MAEC. MAEC je štandardizovaný jazyk na zdieľanie štruktúrovaných informácii o malvéri založených na popise správania, atribútov a vzorov útokov. Na obrázku nižšie je možné vidieť dátový model MAEC-u.



Obr. 02 MAEC dátový model.

Cieľom MAEC-u je poskytnúť základ pre zmenu vo výskume malvéru a zdieľanie informácii týkajúcich sa malvéru. Jeho hlavná snaha je odstránenie nejednoznačnosti a nepresnosti pri popise a zbavenie sa závislosti na popise len niektorých signatúr. Vďaka MAEC -u je možné vylepšiť komunikáciu ohľadom malvéru medzi nástrojmi a ľuďmi a nástrojmi medzi sebou, zabrániť duplicitným analýzam vďaka zdieľaniu informácii a umožniť tak rýchlejší vývoj prostriedkov na boj s malvérom.

MAEC dáta sa primárne ukladajú vo forme XML dokumentu, ktorý sa skladá z rôznych elementov. Všetky dáta bývajú zapuzdrené v jednom z troch typov dátových modelov, ktorými sú MAEC Container, MAEC Package a MAEC Bundle. Každý z týchto dátových modelov je určený na iné použitie a obsahuje iné typy dát o malvéri. MAEC Bundle je určený na zápis a zdieľanie informácií o jednej inštancii malvéru. MAEC Package je určený na uchovávanie elementov zvaných Malware subject, ktoré slúžia na popísanie konkrétnych malvérových inštancií a analýz vykonaných nad týmito inštanciami spolu so zistenými informáciami. Posledný dátový model je MAEC Container, ktorý umožňuje zdieľanie akýchkoľvek MAEC dát. Momentálne tento model zväčša slúži na zdieľanie a zapuzdrenie dát typu MAEC Package.



Obr. 03 MAEC dátové modely.

Ako sme už spomínali v predchádzajúcej kapitole, MAEC je postavený na CybOX-e. Okrem neho je však postavený aj na MMDEF (IEEE ICSG's Malware Metadata Exchange Format).

Keďže kompletná bezpečnostná analýza hrozieb vyžaduje aj popis zraniteľností a potenciálnych cieľov exploitácie, dokáže MAEC využiť vo svojich dokumentoch aj referencie na štandardy ako Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE) a Common Platform Enumeration (CPE).

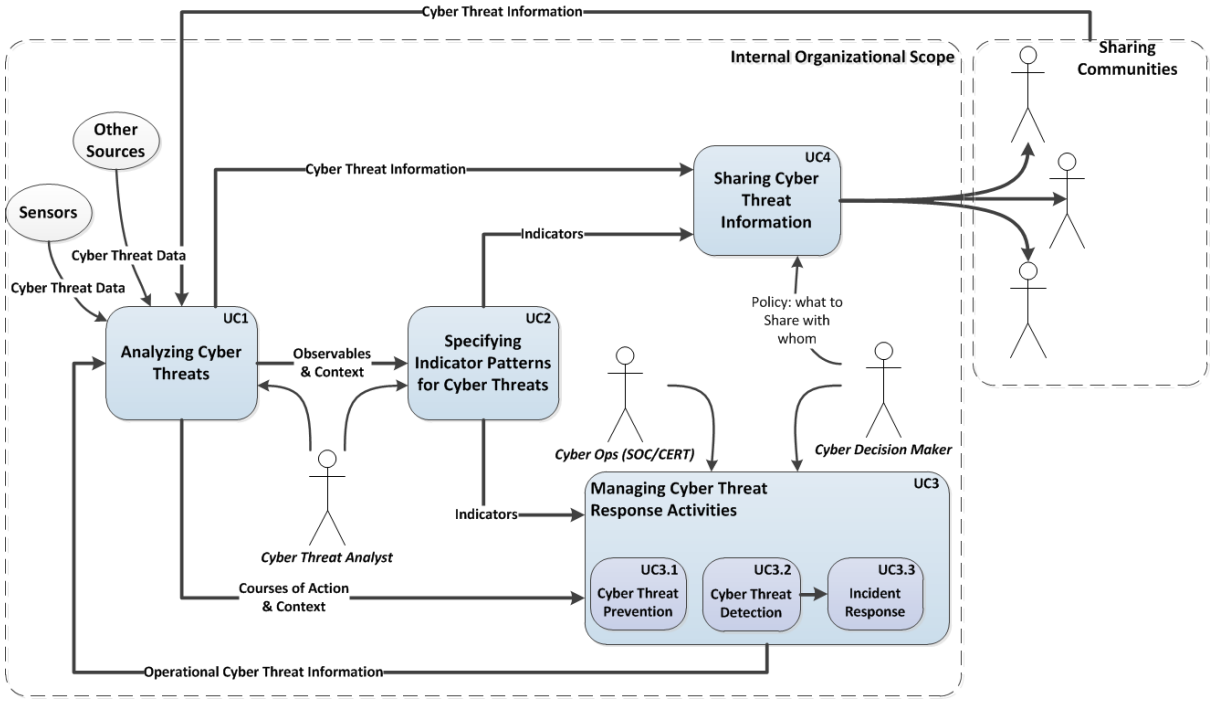
1.2.1 STIX (Structured Threat Information Expression)

Ako sme už spomínali, pre organizácie je čím ďalej tým dôležitejšie mať vždy aktuálne informácie o kyber hrozbách. Jedným z kľúčových komponentov v dosiahnutí tohto je zdieľanie informácii s partnerskými organizáciami a inými dôveryhodnými subjektami. Takéto zdieľanie sa využívalo už dávnejšie avšak nie vždy boli informácie dostatočne detailné a kompletné.

Zatiaľ čo získavanie a zdieľanie kyber bezpečnostných informácii môže pomôcť s rozdelením záťaže pri analýze údajov a následne lepším určením bezpečnostných priorít, hlavným problémom v tomto procese je využitie štandardného zápisu údajov.

Za týmto účelom bol vytvorený STIX, rapídne vyvíjaný jazyk, ktorý vznikol ako komunitná snaha na definovanie a vývoj štruktúrovaného jazyka na popis informácii o kyber hrozbách. Ako aj jazyky, ktoré sme spomínali v predchádzajúcich kapitolách, aj tento sa snaží o detailný popis, možnosti zdieľania, automatického spracovania a zároveň sa snaží zostať čitateľný aj pre ľudí. Napriek tomu, že tento jazyk je pomerne nový, veľké množstvo organizácii zaoberajúcich sa kyber hrozbami ho už začalo používať, alebo zvažuje jeho použitie. Vďaka tomu, že sa jedná o open-source projekt, je možné aby prispievatelia pridávali časti, ktoré sú pre nich dôležité.

Najčastejšie sa využíva na analýzu kyber hrozieb, manažment reakcií na kyber hrozby a zdieľanie informácií. Na obrázku nižšie je možné vidieť diagram prípadov použitia STIX.



Obr 04. STIX use case diagram.

STIX je rozdelený na niekoľko takzvaných základných konceptov kyber hrozieb. Na obrázku nižšie je možné vidieť architektúru STIX-u, pričom jednotlivé ikony znázorňujú tieto koncepty. Šípky medzi ikonami znázorňujú vzťahy. Hviezdičky pri texte na šípkach znázorňujú kardinalitu vzťahu, pričom hviezdička znamená, že vzťah nemusí existovať vôbec alebo môže byť mnohonásobný.

1.2.2 Iné bezpečnostné štandardy

Okrem nami použitých štandardov existuje na trhu aj množstvo iných, väčšinou tiež voľne dostupných kyber bezpečnostných štandardov. Vybrali sme niektoré najznámejšie a najpoužívanejšie a teraz si o nich povieme niečo bližšie.

CAPEC (Common Attack Pattern Enumeration and Classification)

Snahou CAPEC-u je poskytnúť verejne dostupný katalóg útočných vzorov, ktoré sú intuitívne klasifikované, spolu s popisom útokov. Bol vytvorený Oddelením domobrany Spojených štátov (U.S. Department of Homeland Security) v roku 2007 a odvtedy sa stal štandardným mechanizmom pre identifikáciu, zber a zdieľanie útočných vzorov. Je to open-source projekt a prispieva doň aj komunita.

CIQ (Customer Information Quality)

CIQ je jazyk slúžiaci na reprezentáciu informácií o jednotlivcoch a organizáciách. Jeden zo STIX elementov zvaný STIX Identity obsahuje rozšírenie na popis útočníkov, obetí a zdrojov informácii práve pomocou tohto jazyka.

CPE (Common Platform Enumeration)

Je štandardizovaný spôsob pomenovania aplikácií, operačných systémov a hardvérových platforiem. Slúži na identifikáciu rizík a zraniteľností IT manažérmi a pomáha robiť včasné rozhodnutia o inštalácii, obnovovaní alebo odstránení softvéru. Využíva ho aj MAEC na popis softvéru použitého pri analýzach a tiež na identifikáciu spôsobu využitej virtualizácie.

CVE (Common Vulnerabilities and Exposures)

CVE je slovník názvov pre verejne známe kyber bezpečnostné zraniteľnosti. CVE identifikátori zjednodušujú zdieľanie dát medzi rôznymi bezpečnostnými databázami a nástrojmi. Momentálne je CVE pod správou MITRE.

CWE (Common Weakness Enumeration)

Jedná sa o formálny zoznam alebo slovník známych počítačových zraniteľností, ktoré sa môžu objavovať v návrhu, kóde, architektúre alebo implementácii softvéru a môžu teda

viesť k exploitácii. Cieľom tohto štandardu je poskytnúť základ pre identifikáciu zraniteľností, zmiernenie ich následkov a predchádzanie vzniku zraniteľností.

IODEF (Incident Object Description Format)

Jedná sa o štandard vyvinutý pre účely výmeny informácií o incidentoch. Tento štandard je možné využiť spolu so STIX za účelom zápisu informácií o incidente.

MMDEF (Malware Metadata Exchange Format)

Je formát na zápis a zdieľanie informácií o malvéri a je momentálne vyvíjaný organizáciou IEEE. Originálny formát bol vyvíjaný skupinou antivírusových firiem za účelom rozšírenia zdieľanej databázy malvérových vzoriek s informáciami o vzorkách. MAEC využíva niektoré časti tohto formátu.

OpenIOC (Open Indicators of Compromise)

Je to rozšíriteľná XML schéma na popis technických charakteristík, ktoré identifikujú známe riziká, metódy útokov a dôkazy o útokoch. OpenIOC bol vytvorený bezpečnostnou firmou MANDIANT. Jedna časť STIX Indicator je postavená na tejto schéme.

OVAL (Open Vulnerability and Assessment Language)

Je snaha komunity zaoberajúcej sa informačnou bezpečnosťou o štandardizáciu posudzovania a reportovania stavu počítačových systémov. OVAL obsahuje jazyk na kódovanie systémových špecifikácií. Cieľom tohto štandardu je poskytnúť firmám presné informácie, vďaka ktorým by boli schopné zlepšiť svoju bezpečnosť.

TAXII (Trusted Automated eXchange of Indicator Information)

Je otvorený transportný mechanizmus, ktorý štandardizuje automatickú výmenu údajov týkajúcich sa kyber hrozieb. Jeho cieľmi sú bezpečné a rýchle zdieľanie informácií. Je to hlavný transportný mechanizmus výmeny dát pre STIX.

VERIS (Vocabulary for Event Recording and Incident Sharing)

VERIS je framework navrhnutý tak, aby poskytoval jednotný jazyk na popis bezpečnostných incidentov a ich dopadov v štruktúrovanej forme. Je podobný elementu

STIX Incidents, avšak VERIS je určený výhradne na popis stratégií a rizikový manažment po incidente. Veľká časť kódu STIX Incidents bola založená na kóde VERIS-u.

1.3 Ontológia

Existuje viac definícii toho, čo je to ontológia, pričom niektoré si protirečia. Jednou z nich je, explicitné formálne pomenovanie a definícia typov, vlastností a vzťahov medzi entitami, ktoré existujú v určitej doméne skúmania. Ontológie sa skladajú z primitív, ktorými sú triedy (classes), atribúty(attributes, properties), vzťahy(relationships, relations), individuály(individuals, instances).

Individuály sú základnými komponentami ontológie a sú to inštancie objektov. Inštancie reprezentujú konkrétne objekty ako ľudí, zvieratá, molekuly atď., ako aj abstraktné objekty ako sú konkrétne čísla a slová.

Triedy alebo tiež typy, sú abstraktné skupiny alebo množiny objektov. Triedy môžu klasifikovať individuály, iné triedy, alebo kombináciu oboch. Príkladmi tried sú napríklad osoba (reprezentuje triedu ľudí), dopravný prostriedok, auto, vec(reprezentuje triedu všetkých vecí), atď. Triedou sú napríklad tiež vína, pričom nejaké konkrétne víno je inštanciou tejto triedy. Triedy môžu mať aj podtriedy, podobne ako pri objektovo orientovanom programovaní. Podtriedy sú vždy špecifickejšie než ich nadtriedy. Napríklad trieda vína má podtriedy červené, biele a ružové vína. Pričom tie sa dajú ďalej deliť na suché a sladké atď.

Ďalším primitívom v ontológii sú atribúty, ktoré sú podobné atribútom v objektovom programovaní. Môžu ich reprezentovať triedy alebo individuály ako sú napríklad reťazce, čísla alebo premenné, ktoré obsahujú prídavné informácie o objekte.

Vzťahy alebo relácie medzi ontologickými objektmi špecifikujú aký je vzťah medzi nimi. Typicky sa jedná o konkrétnu triedu, ktorá špecifikuje vzťah medzi dvoma objektmi. Napríklad vzťah medzi objektmi Mária Terézia a Jozef II je, že Jozef bol synom Márie. Vzniká tým zároveň aj opačný vzťah a to, že Mária je rodičom/matkou Jozefa. Relácie sú hlavnou črtou ontológie a vyjadrujú sémantiku domény. Množina použitých relačných typov (triedy relácii) vyjadrujú expresívnosť jazyka, v ktorom je ontológia vyjadrená.

Jednými z najdôležitejších typov relácii je relácia is subclass of, teda vyjadruje vzťah medzi dvoma triedami, pričom jedna trieda je podtriedou druhej. Napríklad minerálka je podtriedou nápojov, pričom minerálka je trieda označujúca všetky minerálne vody a nápoj je trieda označujúca všetky možné nápoje.

Druhým veľmi dôležitým typom je type, ktorý vyjadruje, že individuál je inštanciou nejakej triedy. Napríklad kofola je typu nápoj, pričom kofola je konkrétny typ nápoja a nápoj je trieda všetkých nápojov.

Definície týchto primitív obsahujú informácie o ich význame a obmedzeniach ich aplikácie. Pri porovnaní s databázami je ontológia niečo ako abstrakcia analogická ku hierarchickým a relačným databázam, zameraná na modelovanie znalostí o individuáloch.

1.3.1 Reasoning

Jednou z dôležitých vlastností ontológií je podpora reasoning-u (uvažovania). Jedná sa o automatizovaný proces, pri ktorom program zvaný reasoner dokáže dedukovať z ontológie informácie, ktoré neboli explicitne dané. Reasoning však nie je vždy možný a to hlavne v prípadoch keď ontológia obsahuje nejednoznačnosti alebo konflikty, alebo keď je ontológia príliš objemná.

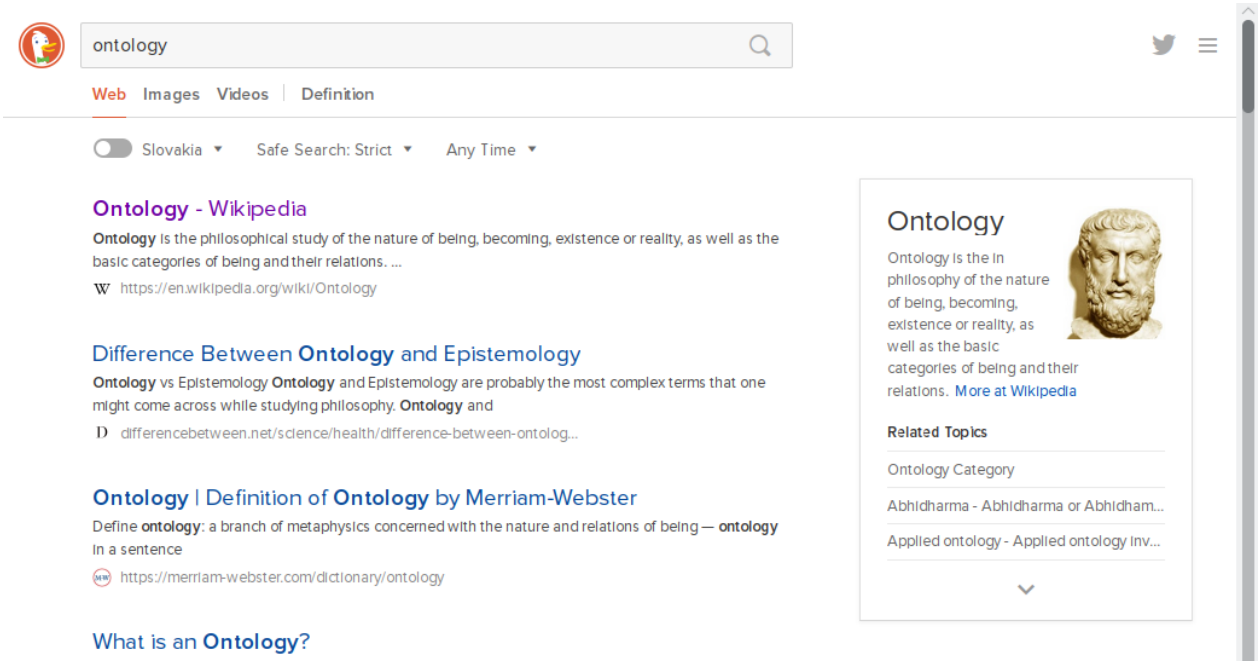
1.3.2 Využitie ontológií

Za posledné roky sa vývoj ontológií presúval z organizácií špecializujúcich sa na umelú inteligenciu do počítačov bežných užívateľov. Ontológia sa na internete stala bežnou. Dnes sa využíva ontológia od obrovských taxonómmi kategorizujúcich web stránky (Yahoo) až po kategorizáciu tovaru v elektronických obchodoch (Amazon).

Na zápis ontológií sa využíva niekoľko jazykov, pričom asi najznámejším z nich RDF (Resource Description Framework), ktorý je vyvíjaný W3C konzorciom a je primárne určený na využitie na webových stránkach tak, aby bol spracovateľný botmi, hľadajúcimi informácie. Ďalším subjektom vyvíjajúcim jazyk na reprezentáciu ontológií je aj DARPA, ktorá vyvíja svoj jazyk zvaný DAML (DARPA Agent Markup Language), ktorý je rozšírením RDF.

Ďalšie uplatnenie si ontológia našla aj v medicíne, kde medicínske zariadenia začínajú využívať ontológie obsahujúce obrovské množstvo štandardizovaných, štrukturovaných termínov ako napríklad SNOMED alebo Unified Medical Language System. SNOMED je najväčším viacjazyčným medicínskym slovníkom na svete.

S ontológiou sa však tiež stretávame pri vyhľadávaní na stránke Google alebo DuckDuckGo, kde pri zadaní vyhľadávanej frázy sa zobrazí na boku alebo na vrchu stránky popis vyhľadávanej frázy (viď obrázok nižšie).



Obr. 05 Využitie dát zo sémantických webov vo vyhľadávači DuckDuckGo.

Pri vyhľadávaní využívajú totiž dáta z takzvaných sémantických webov. Jedná sa vlastne o webové stránky obsahujúce sémantické dáta (ontológiu), určené práve na účely využitia informácií automatizovanými systémami ako sú napríklad boti. Sémantické weby sú zväčša písané vo formáte RDF. Okrem vyhľadávačov tieto informácie využívajú aj rôzni mobilní asistenti ako napríklad Google Now, ktorí využívajú tieto weby na vyhľadávanie štrukturovaných informácii napríklad o počasí alebo o krajinách. Využívajú pritom aj znalosti získané zo sémantickej verzie wikipédie, zvanej dbpedia. Jedná sa vlastne o stránku, ktorá sa postupne snaží obsiahnuť všetky informácie z wikipédie vo forme sémantického webu.

Na získavanie informácii zo sémantických webov a z RDF úložísk všeobecne, sa využívajú dotazy. Napríklad na dotazovanie RDF ontológií sa využíva jazyk SPARQL, ktorý štruktúrou pripomína klasické SQL. Okrem získavania informácií je možné pomocou SPARQL vykonávať skoro všetky činnosti, ktoré dokáže SQL nad databázami, čiže môžeme aj vkladať, vymazávať aj upravovať dáta.

1.3.3 Ciele ontológie

Cieľmi ontológie sú:

1. Zdieľanie jednotnej štruktúry zápisu informácií je najčastejším dôvodom využitia ontológie.

Napríklad ak niekoľko medicínskych webových stránok obsahuje rovnako štruktúrované medicínske informácie vďaka jednotnej ontológii, boti alebo softvéroví agenti sú potom schopní tieto informácie zbierať zo všetkých stránok, ktoré ju využívajú.

2. Umožňuje znovupoužitie znalostí domény, čo znamená, že ak vytvoríme jednu ontológiu dostatočne detailne, výskumníci z inej oblasti môžu využiť časti z našej domény na vytvorenie vlastnej, pričom rovnaké časti našich ontológií zostávajú kompatibilné.

3. Oddelenie doménových znalostí od operačných umožňuje ich zovšeobecnenie. Napríklad môžeme popísať úlohu konfigurácie produktu z komponentov podľa určitej špecifikácie a implementovať program, schopný takejto konfigurácie bez ohľadu na produkt a použité komponenty.

1.3.4 Ontologické syntaxe

Na zápis ontológií sa využíva niekoľko syntaxí zápisu, pričom najznámejšími a najpoužívanejšími sú RDF, turtle, n3 a OWL.

RDF (Resource Description Framework) je štandardná syntax pre dátovú výmenu na internete. Pôvodne sa jednalo o model metdátových dát. Dnes je to však všeobecne používaná metóda na modelovanie a zdieľanie dát pomocou takzvaných web resourcov, teda webových zdrojov. Samotné RDF nie je syntax, ale ako RDF sa označuje aj zápis RDF vo forme XML. Správne by sa mala táto syntax označovať RDF/XML. RDF reprezentuje informácie ako trojice subjektov, predikátov a objektov. Každá z týchto položiek je vyjadrená takzvaným Web URI, čo je vlastne jednoznačný identifikátor.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#"
xmlns:eric="http://www.w3.org/People/EM/contact#" xmlns:rdf="http://www.w3.org/1999/02
/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:mailbox rdf:resource="mailto:e.miller123(at)example"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:personalTitle>Dr.</contact:personalTitle>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.w3.org/People/EM/contact#me">
    <rdf:type rdf:resource="http://www.w3.org/2000/10/swap/pim/contact#Person"/>
  </rdf:Description>
</rdf:RDF>
```

Obr. 06 Ukážka RDF/XML syntaxe.

OWL (Web Ontology Language) je rodina jazykov na reprezentáciu znalostí a na vytváranie ontológií. OWL je postavené na XML štandarde a využíva tiež RDF. OWL rodina obsahuje viac syntaxí a špecifikácií s podobnými menami. Najnovším OWL štandardom je OWL 2, ktorý si rýchlo našiel cestu do sémantických editorov ako Protégé.

Turtle (Terse RDF Triple Language) je formát slúžiaci na zápis dát v RDF dátovom modeli, pričom táto syntax pripomína zápis SPARQL dotazov. Ako sme spomínali RDF dáta sú zapísané ako trojice, ale v turtle syntaxi je možné zoskupovať trojice a vytvárať tak skrátené zápisy.

N3 (Notation3) je spôsob zápisu RDF, ktorý nie je zapisovaný vo forme XML a kladie si za úlohu to, aby bol ľahko čitateľný aj pre ľudí. Obsahuje aj vlastnosti nad rámec RDF. Turtle syntax je zjednodušená N3 syntax, ktorá obsahuje iba RDF vlastnosti.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
    <dc:title>Tony Benn</dc:title>
    <dc:publisher>Wikipedia</dc:publisher>
  </rdf:Description>
</rdf:RDF>
```

Obr. 07 ukážka N3 syntaxe.

1.4 Ontológia v oblasti kyber bezpečnosti

Alarmujúce množstvo kyber útokov, ktoré stále rastie hovorí o tom, že aktuálne využívané spôsoby obrany sú nedostatočné a preto sa uvažuje nad inými spôsobmi obrany. Jedným z takýchto obranných mechanizmov by mohol byť ontologický systém zameraný na oblasť kyber bezpečnosti. Tento systém by mohol byť schopný detekcie hrozieb ako sú zero-day exploity alebo rootkity. Ontologické systémy sú totiž založené na sémantike a potenciálne by mohli vedieť rozlišovať medzi normálnym a škodlivým využívaním počítačového systému.

Ďalším problémom, ktorý sme už spomínali pri kyber bezpečnostných štandardoch je to, že každý bezpečnostný nástroj využíva vlastný formát zápisu údajov. Toto by okrem už spomenutých štandardov bolo možné riešiť aj prostredníctvom technológií využívaných v sémantických weboch a teda využitím ontológie. Dáta by tak boli reprezentované vo formáte, ktorý nielen že je čitateľný pre stroje, ale tiež strojmi pochopiteľný. Ďalšou výhodou je aj možné využitie už existujúcich ontológií, ktoré by sa dali využiť na presnejší popis a tiež na rozšírenie možností popisu. Sémantické technológie využívajú na popis konceptov, takzvané URI, čo sú jednoznačné identifikátory, vďaka ktorým ak sú definované správne, nemôže dochádzať k nesprávnemu pochopeniu ako v ľudskej reči. Príkladom nesprávneho pochopenie môže byť napríklad slovo je, ktoré môže označovať činnosť jedenia ale môže sa aj jednať o slovo odvodené od slova byť. Na prvý pohľad nám nemusí byť zrejmé, o ktorý význam slova sa jedná, avšak toto slovo môže obsahovať atribúty, ktoré nám o ňom povedia niečo bližšie a podľa nich to budeme vedieť určiť.

Ontologický systém by samozrejme nebol a dokonalý a vyžadoval by množstvo úsilia na jeho tvorbu a podľa 2 by zrejme nebol dnes schopný ani boja proti špičkovému malvéru, ale potenciálne by mohol byť schopný rozpoznávania malvéru v pomere 80:20, teda 80% malvéru by mohol dokázať automaticky rozpoznať a zvyšných 20% by muselo byť spracovaných manuálne ľuďmi.

V súčasnosti existuje niekoľko rôznych kyber bezpečnostných ontológii, ktoré sú voľne dostupné, niektoré dokonca pod open-source licenciami. Niektoré z týchto ontológií boli dokonca podporované organizáciami ako US. Department of Homeland Security alebo DARPA, avšak nepodarilo sa nám nájsť žiadnu dostupnú ontológiu, ktorá by bola stále vyvíjaná. Dokonca aj väčšina materiálov, z ktorých sme čerpali boli staré 2 až 4 roky. Je

možné, že tieto organizácie stratili záujem o ďaľší vývoj týchto ontológií alebo prešli na systém closed-source a teda nezverejňujú ďalej zdrojové kódy a ani ďalšie články.

V nasledujúcich podkapitolách sa budeme venovať jednotlivým ontológiám, ktoré sme sa rozhodli otestovať a vyhodnotiť, z dôvodu, že spĺňali naše hlavné kritéria a boli postavené na kyber štandardoch CybOX, MAEC a STIX.

1.4.1 UCO (A Unified Cybersecurity Ontology)

UCO je ontológia vyvíjaná na Marylandskej univerzite. Táto ontológia obsahuje a integruje znalostné schémy z rôznych kyber bezpečnostných štandardov. Víziou tohto projektu je, aby podobne ako dbpedia slúži ako základ pre všeobecné znalosti, slúžilo UCO ako základ pre oblasť kyber bezpečnosti (6). UCO tiež dúfa v to, že sa bude jednať o stále sa rozvíjajúci projekt, ktorý bude obsahovať stále nové dáta z oblasti kyber bezpečnosti.

Základnými črtami UCO je použitie najvyužívanejších kyber bezpečnostných štandardov ako CybOX, MAEC, STIX, CWE, OVE, CPE, OpenIOC a ďalších. V porovnaní s inými bezpečnostnými ontológiami, bolo UCO namapované na niekoľko už existujúcich bezpečnostných ontológii, aby zabezpečil ich interoperabilitu. Niektorými z nich sú napríklad STUCCO a Cloud User Ontology, čo sú bezpečnostné ontológie, ale nie sú postavené na kyber bezpečnostných štandardoch CybOX, MAEC ani STIX, takže sa nimi nebudeme zaoberať.

Vízie UCO sa však žiaľ zrejme nevyplnili, nakoľko pri študovaní materiálov sme zistili, že dátum poslednej úpravy na github stránke bol takmer pred rokom a jednalo sa o iniciálny commit.

1.4.2 ICAS

Účelom vytvorenia ICAS kyber bezpečnostnej ontológie bolo jej využitie v TAPIO (Targeted Attack Premonition using Integrated Operational data) nástroji. Jedná sa o nástroj, ktorý by bol schopný extrahovať dáta z počítačov napríklad na jednej sieti a všetky tieto dáta skombinoval do jedného sémantického grafu, čím by umožnil bezpečnostným tímom rýchlejšie hľadanie prieniku do systému pri útoku. Automatizácia takéhoto systému by zvýšila prehľadnosť dát a tiež znižovala náklady a straty pri útoku. Tento nástroj spolu s ICAS ontológiou bol vyvíjaný organizáciou DARPA, ale dátum poslednej úpravy bol minulý rok, takže sa zrejme tiež jedná o opustený projekt.

1.4.3 Daedafusion

O tejto ontológii sa nám podarilo nájsť najmenej informácií, no aj tak sme sa nakoniec rozhodli využiť práve ju. Oficiálna stránka ani nijaký dokument, ktorý by ju popisoval neexistuje. Dostupné sú iba zdrojové kódy z github stránky a licencia, pod ktorou je táto ontológia uverejnená. Po dlhšom hľadaní sa nám podarilo nájsť autora a podľa jeho Google+ stránky tento projekt začal vyvíjať pred niekoľkými rokmi v snahe založenia firmy, ktorá by sa zaoberala práce oblasťou kyber ontologickej bezpečnosti. Toto sa mu však nepodarilo a tak zostali dostupné akurát zdrojové kódy ontológie, ktorú za ten čas vyvinuli. Tieto kódy sú pomerne staré a v čase písania boli naposledy aktualizované v roku 2015.

1.5 Ontologické úložiská

Ako sme už spomínali, ontológie majú veľmi široké využitie. Avšak samotná reprezentácia dát nestačí, potrebujeme mať dáta niekde uložené a potrebujeme k nim vedieť aj pristupovať. Za účelom ukladania dát vznikli takzvané ontologické úložiská a na získavanie dát z nich slúžia ontologické dotazy, o ktorých sme si už niečo povedali v predchádzajúcich kapitolách. Za účelom ukladania ontologických dát sa často využívajú špeciálne databázy ako napríklad Virtuoso, ale dajú sa využiť aj klasické relačné databázy ako MySQL alebo PL/SQL alebo NoSQL databázy ako MongoDB. My, v našej praktickej časti budeme využívať ontologickú databázu Virtuoso, a preto si o nej teraz niečo povieme.

Virtuoso je moderná vysoko výkonná objektovo-relačná SQL databáza určená pre profesionálne využitie, ktorá poskytuje podporu SPARQL. Okrem ontologickej databázy sa dá využiť aj ako aplikačný server alebo ako relačná databáza. Pre naše účely bude potrebná iba ontologická časť servera. Rozhodli sme sa pre využitie Virtuso databázy, pretože sa jedná o profesionálny produkt, ktorý ale existuje aj v open-source verzii a je využívaný napríklad ako úložisko pre dbpediu.

2 Opis riešenia

V prvej časti tohto popisu praktickej časti práce si povieme o tom, ktorý ontologický model sme sa rozhodli vybrať a prečo, ďalej si povieme o tom aké sú požiadavky na naše programové riešenie. Popíšeme tiež proces vytvárania nášho riešenia spolu so všetkým čo je potrebné na jeho chod.

2.1 Ontológia

Z počiatku sme študovali materiály týkajúce sa tvorby ontológie a plánovali jej vytvorenie. Našťastie sa nám však podarilo nájsť až niekoľko existujúcich ontológií, ktoré aspoň čiastočne spĺňali naše kritériá. Našimi hlavnými kritériami pri výbere ontológie bolo, aby bola postavená na kyber bezpečnostných štandardoch CybOX, MAEC a STIX a aby tieto štandardy čo najlepšie pokrývala a teda obsahovala z nich čo najviac a aby sa jednalo o ontológiu opd open-source licenciou. Podarilo sa nám nájsť 3 takéto ontológie a boli nimi UCO, ICAS a Daedafusion, ktoré sme si už popísali v teoretickej časti práce. Po ich preštudovaní sme si uvedomili, že vytvorenie podobnej ontológie by nebolo možné v rozsahu tejto práce ako vedľajšia úloha, keďže na týchto ontológiách pracovali tímy odborníkov a stále nie sú dokončené. Na to aby sme určili, ktorá z ontológií bude najvhodnejšia, sme museli preštudovať zdrojové kódy.

ICAS sme študovali ako prvý, práve kvôli tomu, že sa jednalo o ontológiu zastrešovanú organizáciou DARPA a mohla teda poskytovať pevné zázemie, napriek tomu, že nebola dlhý čas aktualizovaná. Zistili sme však, že sa jednalo o veľmi nekompletnú ontológiu, ktorá neobsahovala zo CybOX-u takmer nič a vo väčšine zdrojových kódov ostatných štandardov chýbalo veľké množstvo pojmov. Podľa nás sa zrejme jednalo len o iniciálne zdrojové kódy, ktoré mali byť postupne dokončené avšak vývojári sa rozhodli prestať na nich pracovať alebo prešli na closed-source model vývoja, čo je dosť možné keďže túto ontológiu zastrešovala organizácia DARPA.

Druhou skúmanou bola UCO, ktorá taktiež pôsobila profesionálnym dojmom, keďže bola vyvíjaná na Marylandskej univerzite a bolo k nej dostupných aj niekoľko publikácií. Po preskúmaní kódu sme však zostali opäť sklamaní, pretože ontológia neobsahovala pomerne veľké množstvo pojmov zo CybOX-u. MAEC a STIX sa zdali byť kompletné alebo bola to práve absencia CybOX-u, ktorá nás odradila od využitia tejto ontológie.

Po preštudovaní zdrojových kódov posledného kandidáta, Daedafusion, sme si uvedomili, že sa jedná o najkompletnejšie riešenie, čo sa týka obsiahlosti, pretože obsahovalo všetky pojmy CybOX-u, STIX-u aj MAEC-u a dokonca aj ešte nejaké navyše. Táto ontológia obsahovala okrem týchto troch štandardov aj štandardy Argos, CAPEC, CCE, CIQ, CPE, CVE, CWE, DC, GEO, OpenIOC, OVAL, Provenance a SCAP. Tieto štandardy neboli dôležité pre naše riešenie, ale rozhodli sme sa ich tam ponechať kvôli potenciálnemu použitiu v budúcnosti.

Napriek absencii dostupnej literatúry k Daedafusion ontológii, sme sa pre jej kompletnosť rozhodli použiť práve túto ontológiu. Ontológia bola zapísaná v syntaxi RDF/XML.

2.2 Požiadavky na programové riešenie

Cieľom nášho riešenia bolo vytvorenie programu, schopného pracovať s ontológiou a mapovať dáta získané z webových stránok alebo agentových systémov zapísané v niektorom z kyber štandardov CybOX, MAEC alebo STIX na ontológiu ako individuály. Riešenie by malo pozostávať teda z reálnych dát obsahujúcich popis bezpečnostných incidentov alebo malvéru, ďalej programu, ktorý dokáže tieto dáta mapovať na ontológiu a nakoniec ontológie, ktorá bude uložená v ontologickom úložisku a bude možné na ňu posielať SPARQL dotazy.

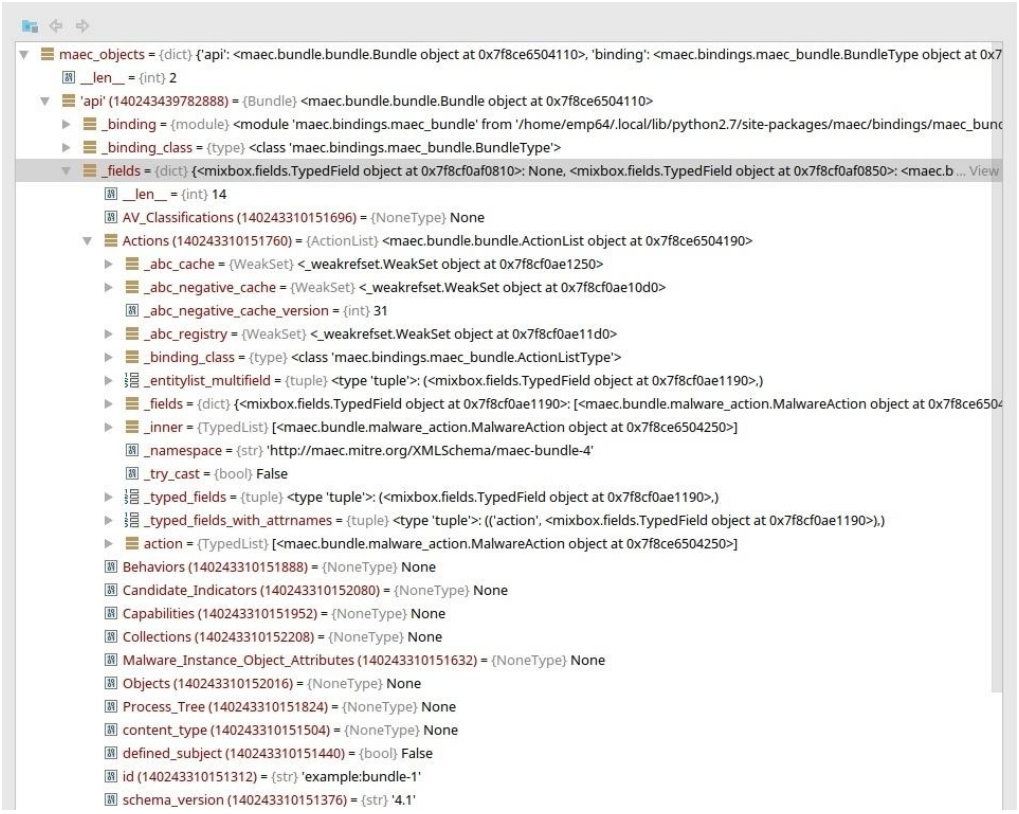
2.3 Programové riešenie

Na vytvorenie programového riešenia sme sa rozhodli zvoliť jazyk Python verziu 2.x. Hlavný dôvod voľby tohto jazyka bola dostupnosť Python API pre všetky kyber bezpečnostné štandardy, ktoré využívame v našom riešení. Ďalším dôvodom bola jeho jednoduchá rozšíriteľnosť o ďalšie moduly ako napríklad modul pre prácu s ontológiami. Pri programovaní sme sa držali štandardného formátovania kódu PEP 8.

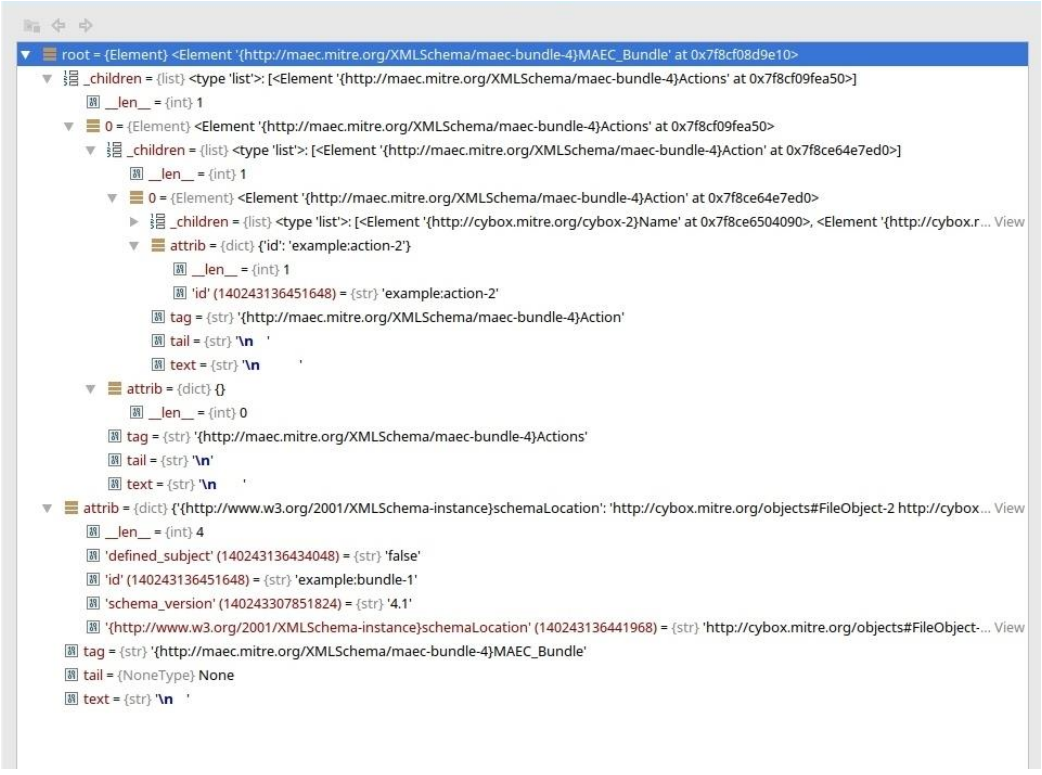
2.4 Parsovanie vstupných dát

Vstupnými dátami pre nás boli XML súbory zapísané v štandarde MAEC a STIX. STIX dáta sa nám podarilo získať z oficiálnej stránky STIX (https://stix.mitre.org/language/version1.0.1/samples.html), ktorá obsahovala príklady určené práve na testovanie. MAEC dáta sme našli na oficiálnej stránke MAEC-u (https://maec.mitre.org/language/examples_v4.0.1.html), kde boli tiež dostupné testovacie XML súbory.

Ako sme už spomínali CybOX, STIX aj MAEC majú dostupné moduly pre jazyk python a umožňujú jednoduché vytváranie tzv. balíkov, ktoré sa dajú následne pomocou týchto modulov aj zapísať v rôznej forme ako napr. vo forme XML súboru alebo JSON súborov. Na vytváranie balíkov sú tieto moduly naozaj ideálne a je možné k nim nájsť množstvo dokumentácie. Horšie je to však s parsovaním. Pokúšali sme sa nájsť dokumentáciu alebo tutoriály k parsovaniu týchto balíčkov, avšak jediný príklad použitia čo sme našli bol v dokumentácii a bola to iba ukážka toho, ako načítať balík zo súboru. My sme potrebovali tento balík prehľadávať do hĺbky (keďže balíky sú štruktúrované ako stromy) a postupne všetky atribúty mapovať na príslušné ontologické prvky. Keď sme si však prehliadali štruktúru balíkov cez debugger, zistili sme, že ich štruktúra je veľmi komplikovaná a meniaca sa podľa vnorených objektov. Po neúspešných pokusoch na vytvorenie spôsobu parsovania týchto súborov sme sa rozhodli pre jednoduchší prístup a to využitie klasického XML parsera na prácu so vstupnými dátami namiesto špecifických modulov CybOX, STIX a MAEC. Štruktúra ktorú sme museli parsovať sa tým zjednodušila a sprehľadnila a elementy boli uložené v XML strome vždy rovnakým spôsobom aj keď sa jednalo o rôzne typy. Kľúčové pre nás bolo získanie stromovej štruktúry elementov, ich názvov, popisu a hodnôt. Skontrolovali sme úplnosť informácií obsiahnutých v XML a zistili sme, že všetky tieto informácie máme k dispozícii aj použitím tohto spôsobu extrakcie informácií. Vytvorili sme funkciu na výpis celého stromu v čitateľnej forme, vďaka čomu sme mohli kontrolovať funkčnosť parsovania. Funkcia, ktorú sme vytvorili na toto parsovanie sa nazýva parse() a zahŕňa v sebe 3 funkcie na parsovanie a to parse_cybox(), parse_maec a parse_stix(). Každá táto funkcia má za úlohu parsovanie súborov zapísaných v danom štandarde.



Obr. 08 XML strom vstupného súboru pri použití MAEC modulu na parsovanie.



Obr. 09 XML strom vstupného súboru pri použití XML modulu na parsovanie.

2.5 Práca s ontologickými súbormi

Ďalšou úlohou bolo upraviť ontológiu do formátu aký sme potrebovali a nájsť spôsob, ktorým by bolo možné pracovať v pythone s ontológiou.

Na prácu s ontológoiu a SPARQL dotazmi sme využili modul zvaný RDFlib. Tento modul dokáže pracovať s rôznymi syntaxami ontológii ako RDF/XML, turtle, n3, n4 a owl. Jeho inštalácia bola jednoduchá a využili sme inštalátor balíkov pre python, pip.

```
$ pip install rdflib rdflib-sparql
```

Po inštalácii stačilo rdflib modul importovať do programu pomocou import rdflib.

Okrem ontologických súborov, dokáže pomocou rôznych pluginov využívať aj iné ontologické úložiská ako napríklad mysql databázu, virtuoso databázu a plsql databázu. Dáta sú načítané do grafu, ktorý slúži ako medzivrstva medzi pythonom a podporovanými ontologickými úložiskami.

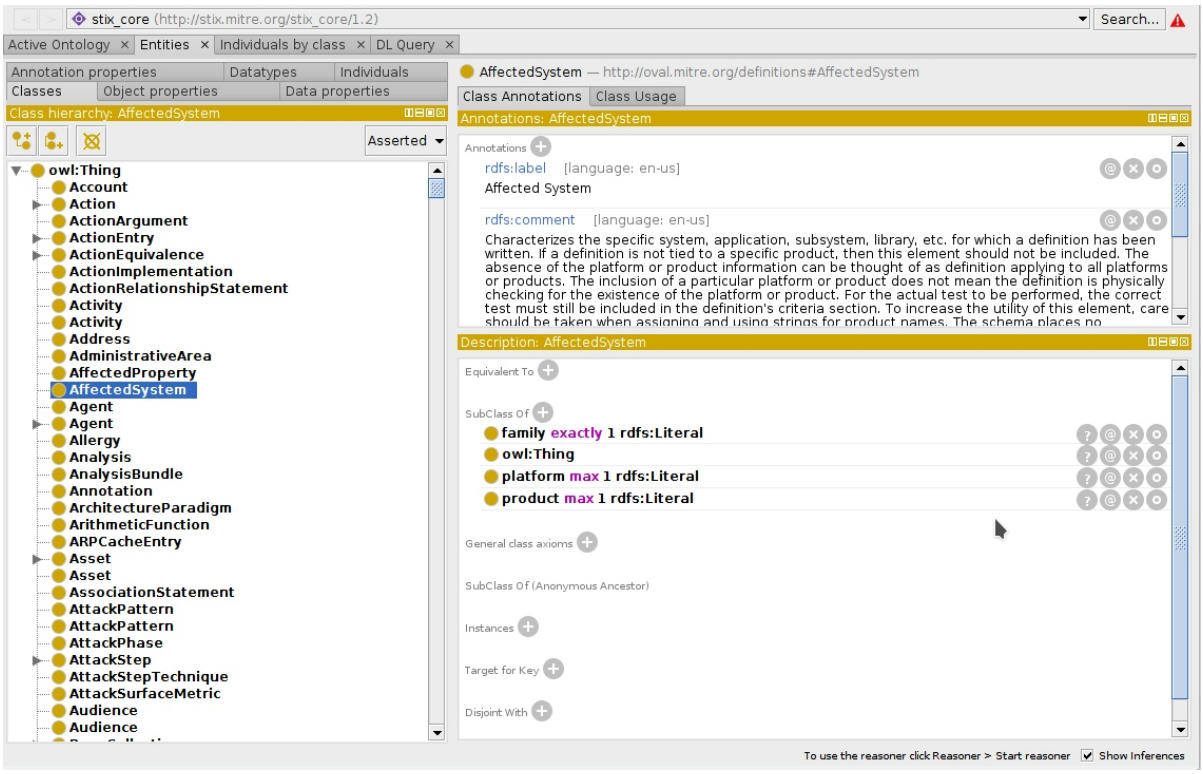
Tento graf je primárne rozhranie na prácu s RDF dokumentami pomocou RDFlib a reprezentuje ho trieda Graph. Jedná sa o neutriedený kontajner poskytujúci operáciu ako pridanie trojice (triple) a tiež vyhľadávanie trojíc. Tento graf je vlastne reprezentáciou RDF grafu, zloženého z RDF trojíc. Množina uzlov, ktoré sa v tomto grafe nachádzajú je vlastne množinou subjektov a objektov. Graf poskytuje množstvo operácií ako aj vyhľadávanie trojíc, union, merge, intersect a iné.

Daedafusion ontológia, ktorú sme sa rozhodli použiť pre náš projekt bola zložená z množstva RDF/XML súborov, vďaka čomu boli kyber bezpečnostné štandardy rozdelené do niekoľkých častí a umožňovalo to lepšiu prehľadnosť a jednoduchšiu editáciu v editore. Pre nás však RDF/XML formát nebol ideálny a to z dôvodu, že sa dal ťažko manuálne editovať a čítať. Rozhodli sme sa preto pre konverziu tejto ontológie na formát turtle, ktorý je ľahšie čitateľný. Problém však nastal v tom, že táto ontológia obsahovala okrem CybOX-u, MAEC-u a STIX-u aj iné štandardy, ktoré sme sa rozhodli v ontológii ponechať. Vznikol tak problém ako prekonvertovať 131 súborov na turtle syntax a nestráviť pritom veľa času. Našťastie vďaka dokumentácii k RDFlib sme zistili, že tento modul dokáže načítavať ontológiu aj vo forme RDF/XML a zároveň dokáže exportovať ontológiu do turtle syntaxe.

Vytvorili sme preto jednoduchý program nazvaný OntologyConvertor.py, ktorý dokázal konvertovať ľubovoľnú ontológiu zapísanú v RDF/XML, turtle, owl, n3 a n4 syntaxi na ľubovoľnú inú syntax.

Naším prvotným zámerom bolo načítavanie ontológie zo súborov pri každom použití do špeciálneho grafu RDFlib a následná práca s ontológiou. Po niekoľkých pokusoch sme si však uvedomili, že spúšťanie celého systému je veľmi pomalé, pretože rekurzívne prechádzanie a načítavanie veľkého množstva súborov trvá pomerne dlho. V budúcnosti by sa tento čas ešte predĺžil, a to z dôvodu, že do ontológie sme plánovali vkladať dáta z bezpečnostných agentov.

Prvé riešenie, ktoré nám napadlo bolo vytvorenie jedného ontologického súboru. To sme zrealizovali, pričom došlo k redukcii veľkosti súboru, avšak na rýchlosti načítavania to veľmi nepridalo. Potrebovali sme lepšie riešenie, ktoré by mohlo poslúžiť aj budúcnosti pri potenciálnom nadviazaní na túto prácu. Rozhodli sme sa pre použitie ontologického úložiska Virtuoso. Keďže sa jedná o profesionálne riešenie pre ontológie, bolo nám jasné, že to bude pre nás postačujúce riešenie.



Obr. 10 Daedafusion ontológia načítaná v Protégé editore.

2.6 Ontologická databáza Virtuoso

Ďalším krokom v našom riešení bola inštalácia Virtuoso databázy. Existujú dve verzie tejto databázy, jedna je komerčná a druhá je open-source. Keďže komerčná je spoplatnená a open-source verzia obsahuje všetko čo potrebujeme, bola voľba jasná.

Databázu sme nechceli inštalovať na priamo na náš hosťovský systém a preto sme rozhodli vytvoriť si virtuálny stroj s GNU/Linuxom, na ktorý sme túto databázu plánovali nainštalovať.

Keďže náš hosťovský systém bol Debian, pri inštalácii virtualboxu sme využili repozitáre tejto distribúcie. Inštaláciu sme vykonali pomocou príkazov zapísaných nižšie.

```
$ apt-get update
$ apt-cache search virtualbox
$ apt-get install virtualbox virtualbox-dkms virtualbox-ext-pack
```

Keď bola inštalácia virtualboxu hotová, vytvorili sme virtuálny stroj na ktorý sme nainštalovali distribúciu Ubuntu. Konkrétne sa jednalo o verziu 16.04 s predĺženou podporou v 64 bitovej verzii. Túto distribúciu sme zvolili kvôli jej jednoduchej použiteľnosti a veľkému množstvu dostupných materiálov a fór v prípade, že by nastali komplikácie. Pri tvorbe virtuálneho stroja sme všetky parametre ponechali tak ako boli prednastavené, akurát veľkosť RAM sme zmenili na 2GB, pre prípad, že by databáza potrebovala viac pamäte.

Ďalej nasledovala inštalácia Virtuoso databázy, spolu s rozšíreniami pre prácu so SPARQL, ktorú sme vykonali nasledovne:

```
# obnovenie repozitárov
$ apt update
# inštalácia Virtuoso servera
$ apt install virtuoso-server virtuoso-vad-{isparql,ods,tutorial}
```

Po inštalácii sa Virtuoso server sám spustí. V prípade, že by sa nespustil dá sa spustiť pomocou príkazu

```
$ /usr/bin/virtuoso-t
```

Konfiguračný súbor pre Virtuso je v /etc/virtuoso-opensource-6.1/virtuoso.ini a jeho databáza je uložená v /var/lib/virtuoso-opensource-6.1/db/.

Keďže sme využili virtualbox na vytvorenie virtuálneho servera a sieť je štandardne nastavená na NAT, bolo tiež potrebné otvoriť a presmerovať porty tak, aby sme mali prístup k webovému rozhraniu z hosťovského počítača. Museli sme povoliť porty 8890 a 1111, ktoré Virtuoso využíva. Port 8890 využíva pre webové rozhranie a port 1111 využíva na komunikáciu cez ODBC.

Po inštalácii by malo byť dostupné webové rozhranie na linke localhost:8890/conductor/.

Na prácu s Virtuosom sme využívali SPARQL konzolu, ktorá je prístupná na linke localhost:8890/sparql/.

2.6.1 Python Virtuoso

Po inštalácii Virtuoso databázy sme chceli otestovať možnosti komunikácie medzi naším programom a virtuoso databázou. Na to aby sme mohli používať virtusoso v našom pythonovom programe, potrebovali sme knižnicu, ktorá s týmto serverom dokáže spolupracovať. Po krátkom hľadaní sme zistili, že knižnica RDFlib, ktorú sme používali už na konverziu ontológii, je možné využiť aj na dotazovanie Virtuoso databázy, pomocou jazyka SPARQL, ak doinštalujeme Virtuoso plugin. Postupovali sme podľa dokumentácie (27). Pokúsili sme sa teda nainštalovať takzvaný virtuoso connector, ktorý nám mal umožniť komunikáciu medzi pythonom a virtuso databázou.

```
$ pip install virtuoso sqlalchemy rdflib
```

Tento pokus však zlyhal, pretože pythonový modul Virtuoso v čase písania tohto textu nebol kompatibilný so štandardnou verziou pyodbc modulu, ktorá sa nachádza v repozitároch pip a je potrebná na komunikáciu s touto databázou. Inštalácia cez pip vždy skončila chybou, spôsobenou nekompatibilnými verziami. S touto chybou sme strávili veľa času, ale nakoniec sa nám podarilo nájsť git repozitár pre Virtuoso python modul, ktorý odkazoval aj na upravenú verziu pyodbc, ktorá bola kompatibilná s Virtuoso pluginom. Po zostavení a nainštalovaní týchto modulov sme zistili, že je nutné ešte doinštalovať na hosťovský systém ODBC(Open Database Connectivity) balík, ktorý zabezpečuje spojenie s databázou.

```
$ sudo apt-get install unixodbc-dev virtuoso-opensource-6.1-bin
```

Bolo tiež nutné upraviť konfiguračné súbory ODBC.

```
$ sudo nano /etc/odbcinst.ini
[VOS]
Description     = Virtuoso
Driver          = /usr/lib/x86_64-linux-gnu/odbc/virtodbc.so
Address         = localhost:1111
UserName        = dba
User            = dba
```

```
$ sudo nano /etc/odbc.ini
[VOS]
Description     = Virtuoso
Driver          = /usr/lib/x86_64-linux-gnu/odbc/virtodbc.so
Address         = localhost:1111
UserName        = dba
User            = dba
```

Inštalácia virtuoso modulov potrebných pre chod Virtuoso pluginu

```
$ pip install werkzeug future
```

Stiahnutie a inštalácia pyodbc modulu:

```
$ git clone https://github.com/maparent/pyodbc.git
$ cd pyodbc
$ sudo python setup.py build install
```

Nakoniec po inštalácii, sme sa rozhodli otestovať spojenie s databázou. Toto však opäť zlyhalo, napriek tomu, že plugin bol funkčný a mal splnené všetky závislosti. Problémom mohla byť nesprávna konfigurácia ODBC súborov v systéme, ale taktiež nekompatibilita tohto pluginu s novou verziou Virtuoso databázy. Nedarilo sa nám prísť na to čo bolo problémom, a preto sme sa z dôvodov nedostatku času a blížiaceho sa konca semestra rozhodli využiť náhradné riešenie.

Týmto riešením bolo využitie pythonovského modulu SPARQLWrapper, ktorý dokáže dotazovať pomocou SPAQRL databázy, bez potreby konfigurácie systémového ODBC. Oproti Virtuoso modulu tu bola nevýhoda toho, že sme pristupovali k SPARQL úložisku na nižšej úrovni, a preto sme v konečnom dôsledku museli viac programovať. Pomocou tohto pluginu sa nám podarilo rýchlo pripojiť k databáze a vďaka vyššej dostupnosti materiálov k tomuto modulu na internete. Napriek našim obavám sa ukázalo, že využívanie tohto modulu je veľmi jednoduché a na pripojenie k Virtuoso databáze, vykonanie dotazu a získanie výsledných dát stačili iba tri príkazy:

```
sparql = SPARQLWrapper("http://localhost:8890/sparql")
sparql.setQuery(queryString)
sparql.setReturnFormat(JSON)
```

SPARQLWrapper vracia dáta vo formáte JSON.

2.6.2 Inicializácia ontológie

Ďalším našim krokom bolo vloženie celej ontológie do Virtuoso databázy. Toto bola ďalšia ťažká úloha, keďže v oficiálnej dokumentácii sme našli iba spôsob ako importovať ontológiu z už existujúcej URI (resp. URL). Opäť sa nám nedarilo nájsť spôsob ako importovať ontológiu zo súboru do Virtuoso databázy. Po dlhšom hľadaní sa nám podarilo nájsť jeden tutoriál, vďaka ktorému sa nám podarilo importovať časť dát. Pri dotaze sme však zistili, že dáta boli nesprávne vložené. V tomto čase už sme nemali čas na ďalšie pokusy, a preto sme sa rozhodli využiť iba súbory ako úložisko pre ontológiu.

2.7 Výsledný program

Keďže sa nám nepodarilo využiť Virtuoso databázu, sústredili sme sa na použitie jedného ontologického súboru, obsahujúceho celú ontológiu (ako sme spomínali v predchádzajúcich kapitolách, spojili sme všetky ontologické súbory do jedného) ako naše ontologické úložisko. Súbor pre naše riešenie znamenal pomalé načítavanie pri každom spustení programu a tiež postupom času hrozilo ešte väčšie spomalenie v prípade ukladania väčších objemov dát. Avšak pre naše testovacie účely v malom meradle to postačovalo.

Ako sme už spomínali, mali sme už hotovú funkciu a parsovanie XML súborov, a preto jediná časť programu, ktorá nám chýbala bolo spojenie s ontológiou. Vytvorili sme funkciu load_data(), ktorá slúžila na načítanie ontológie zo súboru, pričom sme vytvorili aj funkciu load_data_recursive, ktorá slúžila na načítanie viacerých súborov s ontológiou.

Nakoniec sme sa pokúsili vytvoriť funkciu insert_input_data(), ktorá mala mapovať názvy typov a elementov v XML súboroch na názvy typov a elementov v ontológii. Po preskúmaní typov, sme však zistili, že názvy v štandardoch sa líšia od názvov v ontológii. Riešenie by si teda vyžadovalo použitie slovníka ktorý by pároval názvy zo štandardov s názvami v ontológii. Toto sa nám však už nepodarilo zrealizovať.

Program priložený na DVD teda dokáže načítavať a parsovať XML súbory zapísané podľa štandardov a dokáže aj pracovať s ontológiou. Nedostatkom tohto programu však zostáva absencia funkcie, ktorá by dokázalo mapovať vstupné dáta na ontológiu.

Súbory, ktoré sme používali na testovanie programu a tiež kompletné ontológie vrátane celej Daedafusion ontológie v jednom súbore sú dostupné na priloženom DVD.

2.7.1 Popis programov

V tejto časti sme popísali výsledné programy a funkcie, ktoré obsahujú. Prvým z nich je program OntologyConvertor.py, ktorý slúži na konverziu ontologických súborov na iné formáty a tiež spájanie ontologických súborov do jedného. Druhý program je program Dip.py, čo je hlavný program, ktorý spracováva vstupné súbory zapísane v kyber bezpečnostných štandardoch a pracuje s ontologickou databázou.

OntologyConvertor.py

print_help() - táto funkcia slúži na výpis konzolových argumentov a ich možného použitia.

get_format(extension) - vracia formát syntaxe ontológie, podľa vstupnej koncovky.

get_extension(syntax) - vracia koncovku súboru, podľa toho o akú syntax sa jedná.

get_name(dirname) - pomocná funkcia na odstránenie prebytočných znakov cesty.

convert() - funkcia slúžiaca na konverziu súboru na inú syntax.

parse_filename(filename) - rozdelí názov súboru zapísaný v premennej filename na časť obsahujúcu názov a časť obsahujúcu koncovku a tieto dve časti vracia ako návratovú hodnotu.

recursive_convert(dir, extensions_to_convert, syntax) - rekurzívne prehľadáva priečinok dir, vytvára pomocné priečinky a konvertuje súbory na syntax danú premennou syntax.

recursive_convert_merge(dir, extensions_to_convert, syntax, outputfile) - rekurzívne prehľadáva priečinok dir, pričom pridáva všetky ontologické súbory do jedného grafu a na konci skonvertuje tento graf na syntax danú premennou syntax a zapíše ho s názvom super + dir.

main() - hlavná funkcia, ktorá obsahuje parsovanie vstupných argumentov a telo celého programu.

Dip.py

verbose - je globálna premenná určená na zapnutie/vypnutie kontrolných výpisov.

supported_extensions - list obsahujúci zoznam podporovaných ontologických súborových formátov. Podporované sú súbory typu .ttl, .rdf a .n3.

print_help() - táto funkcia slúži na výpis konzolových argumentov a ich možného použitia.

parse_filename(filename) - rozdelí názov súboru zapísaný v premennej filename na časť obsahujúcu názov a časť obsahujúcu koncovku a tieto dve časti vracia ako návratovú hodnotu.

get_format(extension) - vracia formát syntaxe ontológie, podľa vstupnej koncovky.

get_extension(syntax) - vracia koncovku súboru, podľa toho o akú syntax sa jedná.

load_data(filename) - slúži na načítanie ontológie zo súboru, obsahujúceho ontológiu v syntaxi RDF/XML, turtle alebo n3 do grafu. Návratová hodnota tejto funkcie je práve tento graf.

save_data(filename, syntax) - ukladá ontologický súbor v syntaxi danej premennou syntax.

print_graph(graph) - slúži na výpis grafu a teda všetkých objektov a subjektov v ňom obsaihnutých.

graph_union(g1, g2) - slúži na union(spojenie) grafov.

graph_search(graph) - vyhľadávanie v grafe.

virtuoso() - slúži na pripojenie sa k Virtuoso databáze. Napriek tomu, že túto funkciu sme nevyužívali, nechali sme ju v našom konečnom riešení, pre prípadné použite v budúcnosti.

execute_query(query) - vykonanie dotazu nad ontológiou.

extract_field_name(s) - pomocná funkcia pri parsovaní.

recursive_parse_cybox(node, level) - funkcia na rekurzívne prehľadanie CybOX XML stromu.

parse_cybox(filename) - funkcia slúžiaca na načítanie a parsovanie CybOX XML súboru.

recursive_parse_maec(node, level) - funkcia na rekurzívne prehľadanie MAEC XML stromu.

parse_maec(filename) - funkcia slúžiaca na načítanie a parsovanie MAEC XML súboru.

recursive_parse_stix(node, level) - funkcia na rekurzívne prehľadanie STIX XML stromu.

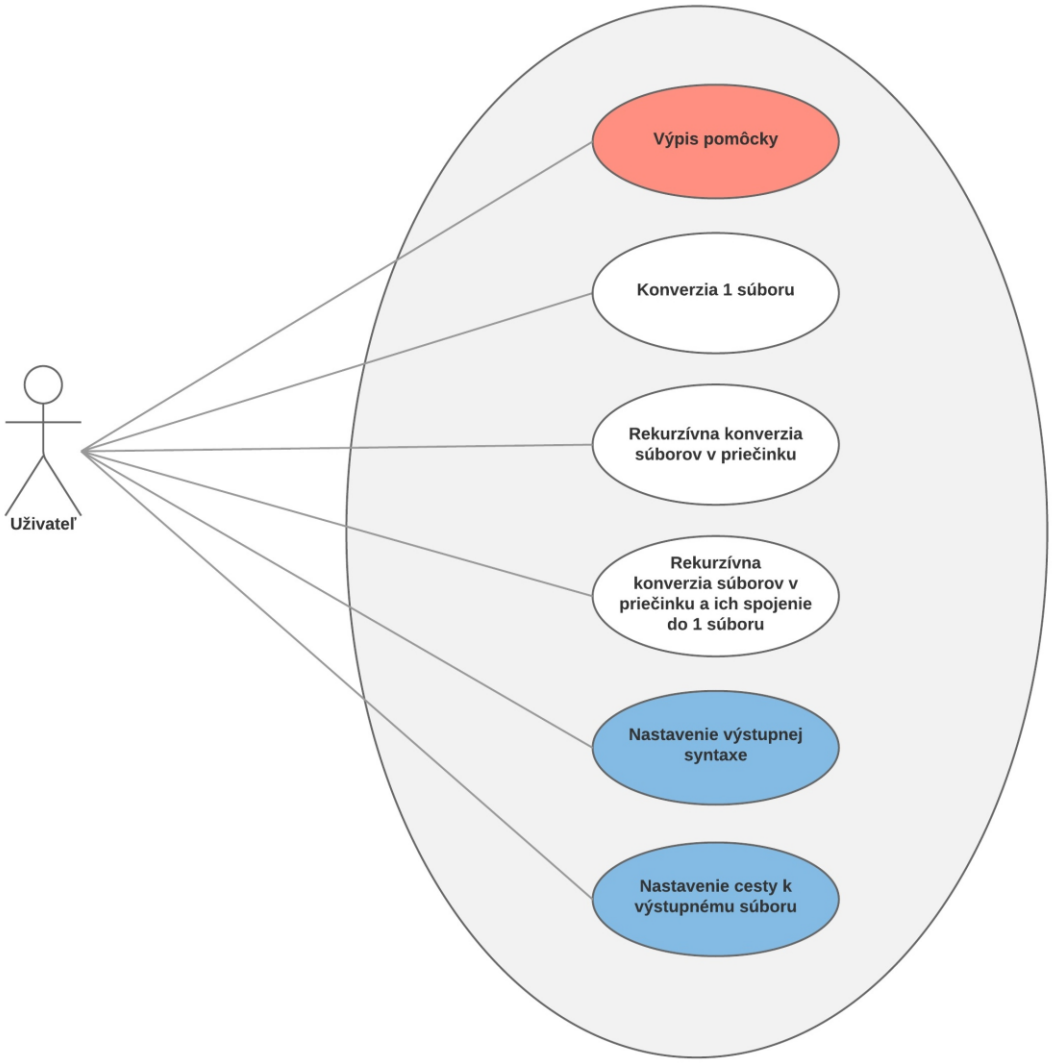
parse_stix(filename) - funkcia slúžiaca na načítanie a parsovanie STIX XML súboru.

parse() - všeobecná funkcia slúžiaca na spustenie parsovacej funkcie podľa obsahu XML súboru.

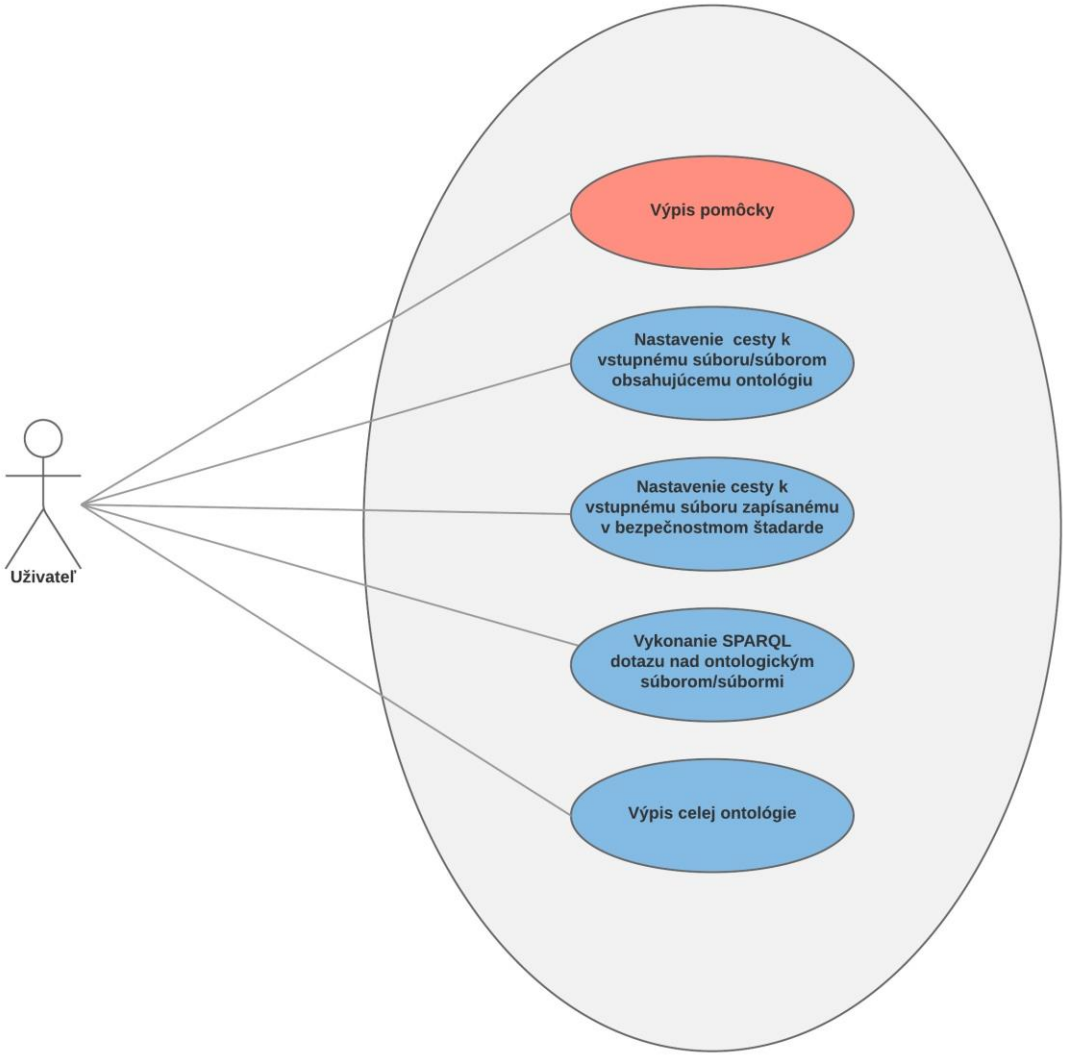
insert_input_data(data) - funkcia, ktorá má za úlohu vkladať do ontológie nové dáta, získané z parsovacích funkcií

main() - hlavná funkcia, ktorá obsahuje parsovanie vstupných argumentov a telo celého programu.

Use case diagramy použitia programov



Obr. 11 Use case diagram použitia programu Dip.py.



Obr. 12 Use case diagram použitia programu OntologyConvertor.py.

3 Zhodnotenie

Naším pôvodným cieľom bolo vytvorenie škálovateľného systému schopného mapovania dát zapísaných vo formáte kyber bezpečnostných štandardov do ontológie. Toto sa nám však podarilo iba čiastočne. Nečakané komplikácie, ktoré nastali pri použití ontologickej databázy nás značne zdržali. Boli sme nútení upustiť od použitia databázy a ako úložisko našej ontológie sme zvolili súbor. Toto nové riešenie má oproti nášmu pôvodnému nižšiu rýchlosť, avšak je jednoduchšie použiteľné a vyžaduje len malé množstvo systémov knižníc.

Kvôli zdržaniu pri snahe o parsovanie vstupných XML súborov pomocou knižníc, ktoré na to boli špeciálne určené sa nám nepodarilo úplne dokončiť programové riešenie. Program dokáže načítavať a parsovať vstupná dáta zapísané podľa štandardov MAEC a STIX. Dokáže tiež načítať ontológiu a pracovať s ňou pomocou SPARQL dotazov. Čo ale nedokáže, je mapovať vstupné dáta na ontológiu. Ukázalo sa totiž, že napriek tomu, že je ontológia postavená na už spomínaných štandardoch, názvy typov a elementov v nej boli pozmenené. Na mapovanie by teda bolo nutné vytvoriť funkciu schopnú párovať všetky názvy použité v štandarde s názvami použitými v ontológii.

S výsledkami práce sme pomerne spokojný aj keď nie všetko vyšlo tak ako sme dúfali a práca je pripravená na potenciálne pokračovanie.

Záver

Cieľom práce bolo preskúmanie a zhrnutie informácií týkajúcich sa využitia ontológie v oblasti kyber bezpečnosti a tiež vytvorenie systému určeného na mapovanie vstupných dát zapísaných vo formáte kyber bezpečnostných štandardov. Prvý cieľ sa nám podarilo splniť v časti Analýza problému, no druhý cieľ sa nám podarilo splniť len čiastočne. Naše programové riešenie dokáže pracovať so vstupnými XML súbormi zapísanými podľa štandardov a dokáže pracovať aj s ontológiou. Čo však je nedokáže, je mapovať vstupné dáta do ontológie. Tento nedostatok vznikol kvôli tomu, že sme narazili na množstvo komplikácii pričom jednou z nich bola aj nekompatibilita názvov použitých v štandarde a názvov v ontológii.

Zoznam použitej literatúry

1 Gammons, B. (január 2017).
6 Must-Know Cybersecurity Statistics for 2017. Získané máj 05, 2017, z https://blog.barkly.com/cyber-security-statistics-2017

2 Oltramari, A., Cranor, L. F., Walls, J. R., McDaniel, P. (2014).
Building an Ontology of Cyber Security. Získané z http://ceur-ws.org/Vol-304/STIDS2014_T08_OltramariEtAl.pdf

3 (20.10.2016).
Cybercrime Reaches New Heights in the Third Quarter. Získané máj 9, 2017, z http://www.pandasecurity.com/mediacenter/pandalabs/pandalabs-q3/

4 Paganini, P. (január 2012).
Israel under a cyber warfare escalation. Získané máj 09, 2017 , z http://securityaffairs.co/wordpress/1747/cyber-warfare-2/israel-under-a-cyber-warfare-escalation.html

5 How Ransomware Spreads and Works. Získané z http://combofix.org/how-ransomware-spreads-and-works.php

6 Syed, Z., Padia, A. a kolektív (február 2016).
UCO: A Unified Cybersecurity Ontology. Získané z http://ebiquity.umbc.edu/_file_directory_/papers/781.pdf

7 Beck, D., Kirilov, I., Chase, P. (12. jún 2014).
The MAEC Language overview. Získané z https://maec.mitre.org/about/docs/MAEC_Overview.pdf

8 Barnum, S. (20. február 2014).
Standardizing Cyber Threat Intelligence Information with the STIX.
Získané apríl 08, 2017, z
https://stixproject.github.io/getting-started/whitepaper/

9 (január 2017).
CAPEC Overview. Získané apríl 08, 2017, z
https://capec.mitre.org/about/index.html

10 (2017).
OASIS Customer Information Quality. Získané apríl 08, 2017, z
https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ciq

11 MITRE (2017).
About CPE. Získané apríl 08, 2017, z
https://cpe.mitre.org/about/

12 MITRE (2017).
About CVE. Získané apríl 08, 2017, z
https://cve.mitre.org/about/

13 MITRE (2017).
CWE Overview. Získané apríl 08, 2017, z
https://cwe.mitre.org/about/

14 MITRE
About STIX. Získané november 10, 2016 , z
https://stixproject.github.io/about/

15 (02. január 2014).
Ties to Existing Standards. Získané máj 25, 2016 z
https://maec.mitre.org/about/standards.html

16 OpenIOC. Získané máj 10, 2017 z
http://www.openioc.org/

17 (09. február 2016).
OVAL. Získané máj 10, 2017 z
https://oval.mitre.org/

18 Obitko, M. (2007).
Reasoning. Získané máj 10, 2017 z
http://www.obitko.com/tutorials/ontologies-semantic-web/reasoning.html

19 (25. február 2014).
Resource Description Framework. Získané máj 10, 2017 z
https://www.w3.org/RDF/

20 Resource Description Framework. Získané máj 10, 2017 z
https://en.wikipedia.org/wiki/Resource_Description_Framework

21 Ontology components. Získané máj 11, 2017 z
https://en.wikipedia.org/wiki/Ontology_components

22 Liu, L., Tamer, M. O. (2009).
Ontology. Získané máj 11, 2017 z
http://tomgruber.org/writing/ontology-definition-2007.htm

23 Obrst, L., Chase, P., Markeloff, R. (2012).
Developing an Ontology of the Cyber Security Domain. Získané november 17, 2016 z
https://pdfs.semanticscholar.org/860d/3d4114711fa4ce9a5a4ccf362b80281cc981.pdf

24 Salem, M. B., Wacek, C.
Enabling New Technologies for Cyber Security
Defense with the ICAS Cyber Security Ontology. Získané z
http://stids.c4i.gmu.edu/papers/STIDS_2015_T06_BenSalem_Wacek.pdf

25 OpenLink Documentation. Získané marec 03, 2017 z
http://docs.openlinksw.com/virtuoso/virtwhydoi/

26 RDFlib documentation. Získané z
https://rdflib.readthedocs.io/en/stable/

27 Virtuoso from Python, Získané z
https://pythonhosted.org/virtuoso/

Prílohy

Príloha A: DVD nosič s programovou časťou práce

DVD príloha obsahuje digitálnu kópiu tejto práce spolu s programovým riešením, ontológiami a testovacími XML súbormi.