

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-16607-92331

**SYSTÉM PRE ZDIEĽANIE POZNATKOV V
BEZPEČNOSTNEJ DOMÉNE VYUŽITÍM ONTOLÓGIE
DIPLOMOVÁ PRÁCA**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-16607-92331

SYSTÉM PRE ZDIEĽANIE POZNATKOV V
BEZPEČNOSTNEJ DOMÉNE VYUŽITÍM ONTOLÓGIE
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.



ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Juraj Puszter**
ID študenta: 92331
Študijný program: aplikovaná informatika
Študijný odbor: informatika
Vedúci práce: Ing. Štefan Balogh, PhD.
Vedúci pracoviska: doc. Ing. Milan Vojvoda, PhD.
Miesto vypracovania: Ústav informatiky a matematiky

Názov práce: **Systém pre zdieľanie poznatkov v bezpečnostnej doméne využitím ontológie**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Zdieľanie poznatkov môže byť kľúčom k včasnej a efektívnej ochrane pred neznámymi útokmi. Pre vytvorenie systému zdieľania je však potrebné zabezpečiť možnosť interakcie mnohých zdrojov informácií do jednotnej platformy pre zdieľanie. Analyzujte existujúce systémy a ich možnosti integrácie a navrhnete vlastné riešenie s využitím ontológie. Implementujte navrhnutý systém a otestujte jej klady a zápory v porovnaní s existujúcimi riešeniami.

Úlohy:

1. Analyzujte systémy zdieľania v bezpečnostnej doméne a vyberte vhodné dáta na zdieľanie.
2. S využitím dostupných nástrojov a nástrojov vykonajte zber dát z internetových zdrojov a iných dostupných zdrojov a transformujte ich do ontologickej reprezentácie.
3. Vytvorte testovaciu vzorku dát z oblasti počítačovej bezpečnosti a ich sémantickú reprezentáciu pre testovanie vyhľadávania požadovaných informácií a ich súvislostí.
4. Optimalizujte sémantický model pre reprezentáciu dát z bezpečnostnej domény v sémantickej databáze z hľadiska efektívnosti ukladania a vyhľadávania.

Zoznam odbornej literatúry:

1. Akbari Gurabi, M., Mandal, A., Popanda, J., Rapp, R., & Decker, S. (2022, August). SASP: a Semantic web-based Approach for management of Sharable cybersecurity Playbooks. In Proceedings of the 17th International Conference on Availability, Reliability and Security (pp. 1-8).
2. Mavroeidis, V., Eis, P., Zadnik, M., Caselli, M., & Jordan, B. (2021, December). On the integration of course of action playbooks into shareable cyber threat intelligence. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 2104-2108). IEEE.

Termín odovzdania diplomovej práce:	12. 05. 2023
Dátum schválenia zadania diplomovej práce:	05. 05. 2023
Zadanie diplomovej práce schválil:	prof. Dr. Ing. Miloš Oravec – garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program :	Aplikovaná informatika
Diplomová práca:	System pre zdieľanie poznatkov v bezpečnostnej doméne využitím ontológie
Autor:	Bc. Juraj Puszter
Vedúci záverečnej práce:	Ing. Štefan Balogh, PhD.
Miesto a rok predloženia práce:	Bratislava 2023

Cieľom tejto práce je vytvorenie systému zdieľania poznatkov v bezpečnostnej doméne. Poznatky sú definované pomocou ontológie. Zabezpečili sme integráciu viacerých zdrojov informácií do jednotnej platformy pre zdieľanie. Súčasťou práce je analýza existujúcich systémov a ich možnosti integrácie do nášho systému. Analyzovali sme vybrané zdroje z ktorých čerpáme dáta. Na základe tejto analýzy sme vytvorili ontologický model do ktorého vkladáme dáta pomocou vytvorených Python skriptov. Opísali sme náš postup implementácie návrhu ontológie a vkladania dát skriptami. Riešenie sme otestovali a zhodnotili.

Kľúčové slová: malware, ontológia, hybridná analýza, MITRE ATT&CK

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION
TECHNOLOGY

Study Programme:	Applied Informatics
Bachelor Thesis:	A system for sharing knowledge in the security domain using ontology
Autor:	Bc. Juraj Puszter
Supervisor:	Ing. Štefan Balogh, PhD.
Place and year of submission:	Bratislava 2023

The goal of this work is to create a knowledge sharing system in the security domain. Knowledge is defined using an ontology. We have ensured the integration of multiple sources of information into a single platform for sharing. Part of this work is the analysis of existing systems and their integration into our system. We have analyzed selected sources from which we draw data. Based on this analysis, we created an ontological model into which we insert data using created Python scripts. We have described our procedure for implementing ontology design and data insertion with scripts. We tested and evaluated the solution.

Key words: malware, ontology, hybrid analysis, MITRE ATT&CK

Vyhlásenie autora

Podpísaný Bc. Juraj Puzster čestne vyhlasujem, že som Diplomovú prácu Systém pre zdieľanie poznatkov v bezpečnostnej doméne využitím ontológie vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

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

V Bratislave dňa 12.05.2023

.....

podpis autora

Pod'akovanie

Ďakujem vedúcemu práce Ing. Štefanovi Baloghovi, PhD. za jeho pomoc, čas a konzultácie pri vypracovávaní tejto práce.

Obsah

Úvod	1
1 Analýza problematiky	3
1.1 Ontológia	3
1.2 Resource Description Framework (RDF)	4
1.3 Protégé	7
1.4 MITRE ATT&CK	9
1.4.1 Technologické domény	10
1.4.2 Taktiky	11
1.4.3 Techniky a podtechniky	11
1.4.4 Procedúry	12
1.4.5 Skupiny	12
1.4.6 Softvér	12
1.4.7 Mitigácie	13
1.4.8 ATT&CK vzťahy medzi hlavnými komponentami	13
1.5 Spôsoby prístupu k MITRE ATT&CK dátam	14
1.6 Hybridná analýza	18
1.7 Existujúce riešenie ontologického modelu pre bezpečnostnú doménu	21
2 Opis riešenia	22
2.1 Ontológia pre MITRE ATT&CK	22
2.2 Naplnenie ontológie MITRE ATT&CK dátami	23
2.3 Rozšírenie ontológie o hybridnú analýzu	29
2.4 Naplnenie ontológie dátami z hybridnej analýzy	30
3 Testovanie riešenia a zhodnotenie výsledkov	33
Záver	34
Zoznam použitej literatúry	35
Prílohy	I

Príloha A:Používateľská príručka II

Príloha B:Tabuľky tried a vzťahov ontológie III

Zoznam obrázkov

Obrázok 1 Trojica RDF grafu (3)	4
Obrázok 2 Príklad RDF grafu.....	5
Obrázok 3 Vývojové prostredie Protégé (7).....	8
Obrázok 4 ATT&CK matica pre doménu Enterprise (8).....	10
Obrázok 5 ATT&CK vzťahy medzi hlavnými komponentami (8)	14
Obrázok 6 ATT&CK príklad vzťahov medzi inštanciami hlavných komponentov (8)	14
Obrázok 7 Príklad práce s Python knižnicou stix2 (9)	15
Obrázok 8 ATT&CK Excel tabuľka (9).....	16
Obrázok 9 ATT&CK Navigator (9)	17
Obrázok 10 ATT&CK Workbench (9)	17
Obrázok 11 Web stránka Hybrid-Analysis.com (13).....	20
Obrázok 12 Príklad správy analýzy z Hybrid-Analysis.com (13)	20
Obrázok 13 Python knižnice použité pri získavaní dát z MITRE ATT&CK	23
Obrázok 14 Načítanie ontológie do objektu	24
Obrázok 15 Uloženie ontológie do súboru.....	24
Obrázok 16 Získanie dát z MITRE ATT&CK	25
Obrázok 17 Funkcia kontrolujúca existenciu dátovej hodnoty.....	26
Obrázok 18 Volanie mapovacích funkcií pre dátové vzťahy domény Enterprise.....	27
Obrázok 19 Úryvok funkcie map_tactics(tactics_data, domain)	27
Obrázok 20 Úryvok funkcie enterprise_map_techniques(techniques_data)	28
Obrázok 21 Funkcia map_software_technique_relation(software_data).....	29
Obrázok 22 Python knižnice použité pri získavaní dát z hybridnej analýzy	30
Obrázok 23 Main() funkcia skriptu získavania hybridnej analýzy	31
Obrázok 24 Úryvok funkcie map_data(all_data)	32
Obrázok 25 Priradenie triedy SampleSummary k triede file alebo url.....	32
Obrázok 26 Naplnená ontológia zobrazená v Protégé.....	32

Zoznam skratiek a značiek

W3C - World Wide Web Consortium

RDF - Resource Description Framework

IRI - International Resource Identifier

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

ASCII - American Standard Code for Information Interchange

JSON - JavaScript Object Notation

HTML - HyperText Markup Language

XML - eXtensible Markup Language

OWL - Web Ontology Language

ICS - Industrial Control System

LSASS - Local Security Authority Subsystem Service

TTP - Taktiky, Techniky a Procedúry

APT - Advanced Persistent Threat

STIX - Structured Threat Information Expression

CTI - Cyber Threat Intelligence

TAXII - Trusted Automated Exchange of Intelligence Information

HTTPS - Hypertext Transfer Protocol Secure

API - Application Programming Interface

SVG - Scalable Vector Graphics

SOC - Security Operations Centre

CERT - Computer Security Incident Response Team

DFIR - Digital Forensics and Incident Response

MAEC - Malware Attribute Enumeration and Characterization

Úvod

V dnešnej dobe je téma digitálnej bezpečnosti veľmi dôležitá a nezanedbateľná. Človek využíva digitálne technológie denne na veľké množstvo služieb. Používa ich napríklad na zábavu, vzdelanie alebo bankovníctvo. Najviac rizikové sú služby kde dochádza ku kontaktu s ľuďmi ako sú sociálne siete alebo email. Pri týchto službách dochádza k zdieľaniu veľkého množstva informácií. Je veľmi jednoduché otvoriť na prvý pohľad bezpečnú, zamaskovanú web stránku alebo dokument, ktorý v sebe obsahuje nebezpečný obsah. Do počítača sa týmto spôsobom dostane malware, ktorý môže mať rôzne dôsledky.

Malware dokáže spôsobiť stratu a odcudzenie dát na našom zariadení. Útočník tak dokáže získať prístup k našim osobným údajom ktoré vie zneužiť. Naše údaje mu umožnia vykonávať akékoľvek digitálne operácie v našom mene, s čím je spojených množstvo problémov. Dokáže sa dostať napríklad na účty sociálnych sietí alebo do bankového účtu.

Aby sme sa vedeli chrániť voči malwaru, potrebujeme byť schopný analyzovať obsah web stránok a súborov. Výsledky analýz treba vyhodnotiť a uložiť vo vhodnej databáze. Vďaka tomu vieme v databáze vyhľadávať už zanalyzované zdroje a zistiť či sú bezpečné alebo nie. Pre efektívne vyhľadávanie je nutné vytvoriť efektívnu znalostnú databázu pomocou ontológie. Táto technológia nám umožní zadefinovať a uchovať získané poznatky z analýz.

Pre jednoznačnú identifikáciu a opis malwaru je potrebné vytvoriť znalostnú bázu. Táto databáza bude obsahovať základné pojmy a znalosti o malwaroch. Vďaka tomu ich budeme vedieť jednoducho kategorizovať, vytvárať medzi nimi vzťahy a filtrovať ich podľa zvolených vlastností. Dáta z analýz a dáta zo znalostnej bázy bude treba prepojiť vhodnými vzťahmi.

Taktiež chceme zabezpečiť integráciu viacerých zdrojov informácií do jednotnej platformy pre zdieľanie. Pre dosiahnutie tohto cieľa je potrebná analýza existujúcich systémov, zdrojov dát a ich možnosti integrácie do nášho systému. Systém bude navrhnutý tak, aby ho bolo možné v budúcnosti vhodne rozširovať o ďalšie poznatky a zdroje informácií.

System musí byť jednoducho použiteľný pomocou používateľského rozhrania. V tejto práci sa venujeme návrhu a implementácie dátovej časti systému. Časť používateľského rozhrania sa nachádza v inej práci.

1 Analýza problematiky

V tejto kapitole analyzujeme technológie a nástroje ktoré použijeme pri riešení zadania práce. Pre pochopenie riešenia zadania je nutné rozumieť problematike ontológií. Rovnako je dôležitá znalosť použitých zdrojov poznatkov, ktoré sme do nášho systému zakomponovali.

1.1 Ontológia

Ontológie opisujú poznatky určitej domény a vzťahy ktoré medzi týmito poznatkami existujú. Aby sme dokázali vytvoriť takýto opis potrebujeme špecifikovať triedy a ich inštancie, vlastnosti, vzťahy, pravidlá, obmedzenia a axiómy. Vďaka tomu ontológie poskytujú zdieľateľnú a opakovane použiteľnú reprezentáciu poznatkov a taktiež dokážu pridať nové poznatky o doméne (1).

Aplikovaním dátového modelu ontológie na jednotlivé fakty môžeme vytvoriť znalostný graf. Znalostný graf je súbor entít kde triedy sú uzly grafu a vzťahy medzi nimi sú hrany grafu. Opisom štruktúry poznatkov ontológiou v doméne pripravíme znalostný graf na uloženie dát. Existujú aj iné metódy pre opis poznatkov ako sú ontológie. Avšak ontológie, vďaka tomu že vyjadrujú vzťahy, umožňujú spájať rôznymi spôsobmi viaceré koncepty s inými konceptami (1).

Ontológie sa používajú pri tvorbe sémantického webu. Preto sú súčasťou štandardov W3C. Poskytujú používateľom potrebnú štruktúru na prepojenie rôznych informácií na webe prepojených dát, anglicky Web of Linked data, čo je koncept využívaný pri sémantickom vyhľadávaní. Ontológie sa používajú na špecifikáciu bežných modelových reprezentácií údajov z distribuovaných a heterogénnych systémov a databáz. Preto umožňujú spoluprácu rôznych databáz, vyhľadávanie medzi databázami a jednoduchú správu poznatkov (1).

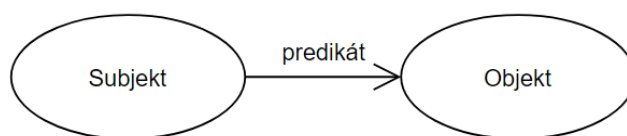
Hlavné charakteristiky ontológií sú zabezpečenie všeobecného chápania informácií a explicitné doménové predpoklady. Výsledkom je vysoká efektivita pri riešení problémov spojených s prístupom k údajom a použitím vo veľkých organizáciách. Vďaka zlepšeniu kvality dát im dokážu organizácie lepšie porozumieť. Jednou z hlavných vlastností ontológií je že sú v nich zabudované základné vzťahy medzi pojmami. Preto dokážu automaticky uvažovať o údajoch a teda odvodzovať vlastnosti. Takéto uvažovanie sa

implementuje ľahko v databázach sémantických grafov. Ontológie fungujú podobne ako ľudia uvažujú a vnímajú vzájomne prepojené koncepty. Navyše taktiež umožňujú zrozumiteľnejšiu a jednoduchšiu navigáciu pri prechádzaní z jedného konceptu do druhého. Ďalšou dobrou vlastnosťou je že ontológie sa dajú jednoducho rozšíriť. Výsledkom je vývoj modelu rastom dát, bez ovplyvnenia závislých procesov a systémov, v prípade že sa niečo pokazí alebo niečo treba zmeniť. Ontológie tiež poskytujú prostriedky pre reprezentáciu akýchkoľvek dátových formátov. Dáta môžu byť štruktúrované, neštruktúrované alebo aj pološtruktúrované. To umožňuje plynulejšiu integráciu dát a jednoduchšiu dátovú analýzu (1).

Keďže ontológie definujú pojmy používané na opis a reprezentáciu domény poznatkov, používajú sa v mnohých aplikáciách na zachytenie vzťahov a posilnenie správy poznatkov. Ontológie sú kostry pre reprezentáciu poznatkov v doméne. Pretože dokážu popísať vzťahy a sú veľmi dobre vzájomne prepojitelné, sú dobrý základ pre modelovanie kvalitných, prepojených a koherentných dát (1).

1.2 Resource Description Framework (RDF)

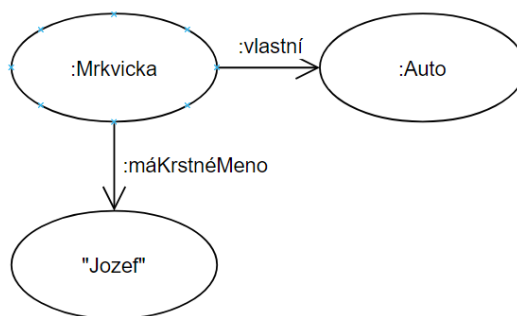
RDF je jazyk pomocou ktorého dokážeme implementovať ontológie. Základnou štruktúrou syntaxe je množina trojíc. Každá trojica sa skladá zo subjektu, predikátu a objektu. Súbor týchto trojíc sa nazýva RDF graf. Subjekty a objekty sú uzlami grafu a predikáty sú hrany medzi nimi. RDF výraz vyjadruje vzťah medzi dvoma zdrojmi a to medzi subjektom a objektom. Vzťah medzi nimi je vyjadrený predikátom. Vzťah je formulovaný jedným smerom, od subjektu k objektu a nazývame ho vlastnosť. Pretože sa RDF výrazy skladajú z troch elementov, nazývame ich trojice. V trojici RDF grafu sa môžu nachádzať tri typy dát a to IRI, literály a prázdne uzly. Na obrázku je znázornená trojica RDF grafu (2) (3).



Obrázok 1 Trojica RDF grafu (3)

Na nasledujúcom obrázku je zobrazený jednoduchý RDF graf ktorý sa skladá z dvoch trojíc. Subjektom je v grafe Mrkvička a objektom je Auto a literál Jozef. Predikáty sú

vlastní a máKrstnéMeno. Z grafu môžeme vyčítať tvrdenie, že Mrkvicka má krstné meno Jozef a tiež že vlastní Auto.



Obrázok 2 Príklad RDF grafu

IRI je jednoznačný identifikátor zdroja. Pojem IRI je zovšeobecnením identifikátora URI. IRI umožňuje použitie znakov iných ako ASCII. IRI sa môže nachádzať na všetkých troch pozíciách trojice (2).

Literály sú základné hodnoty ktoré nie sú IRI. Sú to napríklad reťazce ako mená, dátumy alebo čísla. Každý literál má jednoznačne určený dátový typ aby bolo možné správne interpretovať a analyzovať tieto hodnoty. K reťazcovým literálom je možné pridať jazykovú značku, ktorá udáva jazyk obsahujúceho textu. Literály sa môžu v RDF trojici nachádzať iba na pozícii objektu (2).

IRI a literály sú vo väčšine prípadov dostačujúce pre písanie RDF výrazov. Niekedy ale chceme hovoriť o zdrojoch bez toho, aby sme používali IRI. Môžeme napríklad povedať, že obraz Mona Lisa má v pozadí neidentifikovaný strom, o ktorom vieme, že je to cyprus. Cyprusový strom je teda zdroj bez identifikátora a je reprezentovaný prázdny uzlom. Prázdne uzly sú podobné jednoduchým premenným v algebre. Predstavujú niečo bez toho, aby mali nejakú hodnotu. Prázdne uzly sa môžu v RDF trojici nachádzať na pozícii subjektu a objektu (2).

Zdroje sa rozdeľujú do skupín ktoré nazývame triedy, pričom triedy samotné sú tiež zdroje. Členov triedy nazývame inštalácie triedy. Triedy identifikujeme pomocou IRI a môžu byť opísané pomocou RDF vlastností. RDF rozlišuje medzi triedou a množinou jej inštalácií. S každou triedou je spojená množina jej inštalácií, pričom dve rôzne triedy môžu mať rovnakú množinu inštalácií. Napríklad daňový úrad môže definovať triedu ľudí žijúcich na rovnakej adrese ako adresa vlastníka danej nehnuteľnosti. Pošta môže definovať triedu ľudí, ktorých adresa má rovnaké PSČ ako adresa vlastníka nehnuteľnosti. Je možné, že tieto triedy budú mať presne rovnaké inštalácie, ale budú mať odlišné

vlastnosti. Len jedna z tried má vlastnosť, ktorú definoval daňový úrad a len druhá má vlastnosť, ktorú definoval poštový úrad (4).

RDF definuje množstvo základných vlastností a tried, ktoré je možné použiť pri návrhu ontológie. Medzi týmito vlastnosťami a triedami existujú určité vzťahy, pomocou ktorých je možné automaticky odvodzovať poznatky. Toto odvodzovanie sa vykonáva pomocou dedukčných pravidiel a axiém (5).

V nasledujúcej časti opíšeme niektoré zo základných RDF tried.

`rdfs:Resource` - Je to trieda všetkých vecí opísaných v RDF. Nazývajú sa zdroje a sú inštanciami triedy `rdfs:Resource`. Všetky ostatné triedy sú podtriedami tejto triedy. `rdfs:Resource` je inštanciou `rdfs:Class` (4).

`rdfs:Class` - Trieda RDF zdrojov ktoré sú RDF triedy. `rdfs:Class` je inštanciou `rdfs:Class` (4).

`rdfs:Literal` - Trieda literálov, ako sú reťazce a celé čísla. `rdfs:Literal` je inštanciou `rdfs:Class` (4).

`rdfs:Datatype` - Trieda dátových typov. Všetky inšcie `rdfs:Datatype` zodpovedajú RDF modelu dátových typov. `rdfs:Datatype` je inštanciou aj podtriedou `rdfs:Class`. Každá inšcia `rdfs:Datatype` je podtriedou `rdfs:Literal`. Medzi základné dátové typy patrí napríklad `xsd:string`, `xsd:integer` a `xsd:boolean` (4).

`rdf:Property` - Trieda vlastností RDF. `rdf:Property` je inštanciou `rdfs:Class` (4).

V nasledujúcej časti opíšeme niektoré zo základných RDF vlastností.

`rdfs:range` - Inšcia `rdf:Property`, ktorá sa používa na vyjadrenie, že hodnoty vlastnosti sú inštanciami jednej alebo viacerých tried (4).

`rdfs:domain` - Inšcia `rdf:Property`, ktorá sa používa na vyjadrenie, že akýkoľvek zdroj, ktorý má danú vlastnosť, je inštanciou jednej alebo viacerých tried (4).

`rdf:type` - Inšcia `rdf:Property`, ktorá sa používa na vyjadrenie, že zdroj je inštanciou triedy (4).

`rdfs:subClassOf` - Inšcia `rdf:Property`, ktorá sa používa na vyjadrenie, že všetky inšcie jednej triedy sú inštanciami inej triedy (4).

`rdfs:subPropertyOf` - Inšcia `rdf:Property`, ktorá sa používa na vyjadrenie, že všetky zdroje súvisiace s jednou vlastnosťou súvisia aj s inou vlastnosťou (4).

Na zapisovanie RDF grafov existuje množstvo rôznych jazykov. Rôzne spôsoby zapisovania toho istého grafu však vedú k úplne rovnakým trojiciam, a preto sú logicky ekvivalentné (2).

Jazyk N-Triples poskytuje jednoduchý riadkový, textový spôsob serializácie RDF grafov. Každý riadok predstavuje jednu RDF trojicu (2).

Jazyk Turtle je rozšírením N-Triples. Okrem základnej syntaxe N-Triples, Turtle zavádza množstvo syntaktických skratiek. Je to napríklad podpora pre predpony menného priestoru, zoznamy a skratky pre literály dátových typov. Jazyk Turtle poskytuje kompromis medzi jednoduchosťou písania, jednoduchou analýzou a čitateľnosťou (2).

Keďže syntax jazyka Turtle podporuje iba špecifikáciu jednotlivých grafov bez prostriedkov na ich pomenovanie, existuje jazyk TriG. Tento jazyk je rozšírením jazyka Turtle a umožňuje špecifikáciu viacerých grafov vo forme RDF súboru údajov (anglicky dataset) (2).

N-Quads je jazyk ktorý je jednoduchým rozšírením jazyka N-Triples. Umožňuje výmenu súborov údajov (2).

Jazyk JSON-LD poskytuje JSON syntax pre RDF grafy a súbory údajov. Tento jazyk umožňuje transformáciu JSON dokumentov do RDF s minimálnymi zmenami. Ďalej obsahuje aj univerzálne identifikátory pre objekty JSON, vďaka čomu dokáže JSON dokument odkazovať na iný JSON dokument ktorý sa nachádza niekde inde na webe a rovnako aj schopnosť narábať s dátovými typmi a jazykmi. JSON-LD taktiež poskytuje spôsob serializácie RDF súborov údajov pomocou kľúčového slova @graph (2).

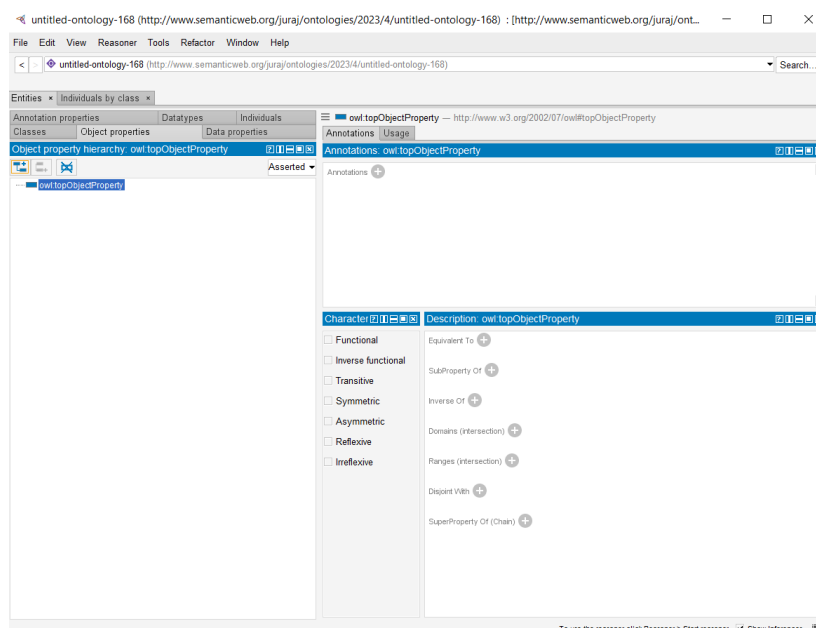
Jazyk RDFa umožňuje vkladanie RDF dát do HTML a XML dokumentov. Vyhľadávacie nástroje dokážu potom tieto údaje pri prehľadávaní webu použiť na obohatenie výsledkov vyhľadávania (2).

RDF/XML jazyk poskytuje XML syntax RDF grafov. Keď bolo RDF pôvodne vyvinuté, toto bola jediná, originálna syntax a dodnes ju niektorý volajú iba RDF. Trojice sú špecifikované vo vnútri XML elementu rdf:RDF (2).

1.3 Protégé

Protégé je vývojové prostredie ontológie OWL. Používateľské rozhranie protégé sa skladá z množstva kariet. Väčšina kariet má spoločný vzhľad a pozostáva zo zoznamov sekcií. Tieto sekcie obsahujú položky týkajúce sa aktuálne vybranej triedy, vlastnosti alebo

jednotlivca, ktorý je nazývaný aj inštancia. Na obrázku vidíme otvorené vývojové prostredie Protégé 5.5.0 na karte správy objektových vlastností (6).



Obrázok 3 Vývojové prostredie Protégé (7)

Na karte **Classes** dokážeme vytvárať a odstraňovať triedy ontológie. Taktiež dokážeme spravovať ich hierarchickú štruktúru. Karty **Individuals** a **Individuals by class** umožňujú správu jednotlivcov, teda inštancií tried. Karta **Annotation properties** umožňuje správu anotácií a karta **Datatypes** správu dátových typov. Na karte **Object properties** spravujeme objektové vzťahy. Podobne na karte **Data properties** spravujeme dátové vzťahy, teda vzťahy s literálmi. Obe karty pre správu vlastností umožňujú nastaviť aj doménu a rozsah vzťahu. Ďalej tu dokážeme nastaviť aj charakteristiky vzťahu (6).

Pre dátové vzťahy dokážeme nastaviť iba jednu charakteristiku, a to je funkčnosť. Funkčný vzťah znamená, že pre každého jednotlivca môže mať vlastnosť najviac jednu hodnotu. Napríklad pre danú osobu môže existovať najviac jeden vzťah, ktorý mu priradzuje jeho meno. Jedna osoba teda nemôže mať viac mien. Ak je viacero jednotlivcov špecifikovaných ako hodnoty vlastnosti, tak tieto hodnoty budú odvodené tak, aby označovali rovnaký objekt (7).

Pre objektové vzťahy dokážeme nastaviť sedem charakteristík. Sú to funkčnosť, inverzná funkčnosť, tranzitívnosť, symetrickosť, asymetrickosť, reflexívnosť a nereflexívnosť (7).

Princíp funkčnosti objektových vzťahov je rovnaký ako pri dátových vzťahoch. Inverzná funkčnosť znamená, že inverzný vzťah vybraného vzťahu je funkčný (7).

Tranzitívny vzťah znamená, že ak jednotlivec x súvisí s jednotlivcom y a jednotlivec y súvisí s jednotlivcom z, potom jednotlivec x bude súvisieť s jednotlivcom z (7).

Symetrickosť udáva že vlastnosť je sama k sebe inverzná. Takže ak jednotlivec x súvisí s jednotlivcom y, potom jednotlivec y musí tiež súvisieť s jednotlivcom x popri tej istej vlastnosti (7).

Asymetrickosť je opak symetrickosti. Udáva že ak jednotlivec x súvisí s jednotlivcom y, potom jednotlivec y nesúvisí s jednotlivcom x popri tej istej vlastnosti (7).

Ak je vlastnosť reflexívna, spôsobí to že každý jednotlivec súvisí sám so sebou prostredníctvom tejto vlastnosti (7).

Nereflexívnosť je opak reflexívnosti. Znamená to že jednotlivec nemôže súvisieť sám so sebou prostredníctvom tejto vlastnosti (7).

1.4 MITRE ATT&CK

MITRE ATT&CK je celosvetovo dostupná báza znalostí a model pre správanie kybernetických protivníkov. Bola vytvorená na základe pozorovaní z reálneho sveta. Odzrkadľuje rôzne fázy životného cyklu útoku protivníka a platformy ktoré ich útokmi zameriavajú. MITRE ATT&CK sa zameriava na to, ako externí protivníci pôsobia vnútri počítačových informačných sietí. Pôvodne vznikol z projektu dokumentácie taktík, techník a procedúr ktoré protivníci využívajú na útok Microsoft Windows. Neskôr sa rozrástol o ďalšie platformy Linux a macOS. Taktiež sa rozrástol aby obsahoval správanie ktoré vedie ku kompromisu prostredia a tiež o ďalšie technologické domény ako mobilné zariadenia a priemyselné riadiace systémy (8).

MITRE ATT&CK sa používa ako základ pre vývoj špecifických modelov hrozieb a metodológií. Využíva sa v súkromnom aj vládnom sektore a tiež v komunite produktov a služieb kybernetickej bezpečnosti. Poskytuje taxonómiu pre útok aj obranu a stal sa užitočným nástrojom pre mnohé disciplíny kybernetickej bezpečnosti. Sprostredkuje informácie o hrozbách, vykonávanie testovania prostredníctvom red teamingu alebo emulácie protivníkov. Zlepšuje sieťovú a systémovú bezpečnosť proti preniknutiu útočníkom. Proces ktorý MITRE použil pri vytvorení ATT&CK a filozofia ktorú vytvorili pre vytváranie nového obsahu sú dôležité pri ďalších snahách o vytvorenie podobných modelov protivníkov a zdrojov informácií. Na obrázku môžeme vidieť ako vyzerá web stránka MITRE ATT&CK (8).

MITRE ATT&CK je behaviorálny model a skladá sa z niekoľkých komponentov. Sú to techniky, taktiky, mitigácie, skupiny a softvér. Základ tvorí súbor techník a podtechník. Tie predstavujú akcie ktoré môžu útočníci vykonať pre dosiahnutie cieľa. Taktiky predstavujú kategórie týchto cieľov pričom techniky a podtechniky pod nich spadajú (8).

Vzťahy medzi technikami, podtechnikami a taktikami sú znázornené pomocou ATT&CK matice, ktorú môžeme nájsť na MITRE ATT&CK webstránke. Na obrázku môžeme vidieť túto maticu pre doménu Enterprise. Každý stĺpec predstavuje jednu taktiku a techniky ktoré ku nej patria. Navyše je možné jednotlivé techniky rozkliknúť a tak zobrazíť detailnejšie podtechniky (8).

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
9 techniques	10 techniques	18 techniques	12 techniques	34 techniques	14 techniques	23 techniques	9 techniques	16 techniques	16 techniques	9 techniques	13 techniques
Drive-by Compromise Exploit Public-Facing Application External Remote Services Hardware Additions Phishing (3) Replication Through Removable Media Supply Chain Compromise (2) Trusted Relationship Valid Accounts (3)	Command and Scripting Interpreter (3) Exploitation for Client Execution Inter-Process Communication (2) Native API Scheduled Task/Job (3) Shared Modules Software Deployment Tools System Services (2) User Execution (2) Windows Management Instrumentation	Account Manipulation (3) BITS Jobs Boot or Logon Autostart Execution (11) Boot or Logon Initialization Scripts (2) Browser Extensions Compromise Client Software Binary Create Account (1) Create or Modify System Process (4) Event Triggered Execution (1) External Remote Services Hijack Execution Flow (13) Implant Container Image Office Application Startup (1) Pre-OS Boot (3) Scheduled Task/Job (1) Server Software Component (1) Traffic Signaling (1) Valid Accounts (1)	Abuse Elevation Control Mechanism (4) Access Token Manipulation (2) BITS Jobs Boot or Logon Autostart Execution (11) Boot or Logon Initialization Scripts (2) Browser Extensions Create or Modify System Process (4) Event Triggered Execution (1) Exploitation for Privilege Escalation Group Policy Modification Hijack Execution Flow (13) Process Injection (11) Scheduled Task/Job (1) Valid Accounts (4)	Abuse Elevation Control Mechanism (4) Access Token Manipulation (2) BITS Jobs Deobfuscate/Decode Files or Information Direct Volume Access Execution Guardrails Exploitation for Defense Evasion File and Directory Permissions Modification (1) Group Policy Modification Hide Artifacts (2) Hijack Execution Flow (13) Impair Defenses (2) Indicator Removal on Host (4) Indirect Command Execution Masquerading (1) Modify Authentication Process (2) Modify Registry Obfuscated Files or Information (3) Pre-OS Boot (3) Process Injection (11) Revert Cloud Instance Rogue Domain Controller Rootkit Signed Binary Proxy Execution (10) Signed Script Proxy Execution (1) Subvert Trust Controls (4) Template Injection Traffic Signaling (1) Trusted Developer Utilities Proxy Execution (1) Unused/Unsupported Cloud Regions Use Alternate Authentication Material (4) Valid Accounts (4) Virtualization/Sandbox Evasion (1) XSL Script Processing	Brute Force (4) Credentials from Password Stores (2) Exploitation for Credential Access Forced Authentication Input Capture (4) Man-in-the-Middle (1) Modify Authentication Process (2) Network Sniffing OS Credential Dumping (3) Steal Application Access Token Steal or Forge Kerberos Tickets (2) Steal Web Session Cookie Two-Factor Authentication Interception Unsecured Credentials (1)	Account Discovery (3) Application Window Discovery Browser Bookmark Discovery Cloud Service Dashboard Cloud Service Discovery Domain Trust Discovery File and Directory Discovery Network Service Scanning Network Share Discovery Network Sniffing Password Policy Discovery Peripheral Device Discovery Permission Groups Discovery (1) Process Discovery Query Registry Remote System Discovery Software Discovery (1) System Information Discovery System Network Configuration Discovery System Network Connections Discovery System Owner/User Discovery System Service Discovery System Time Discovery	Exploitation of Remote Services Internal Spearphishing Lateral Tool Transfer Remote Service Session Hijacking (2) Remote Services (4) Replication Through Removable Media Software Deployment Tools Taint Shared Content Use Alternate Authentication Material (4) Data from Network Shared Drive Data from Removable Media Data Staged (2) Email Collection (2) Input Capture (4) Man-in-the-Browser Man-in-the-Middle (1) Screen Capture Video Capture	Archive Collected Data (3) Audio Capture Automated Collection Clipboard Data Data from Cloud Storage Object Data from Information Repositories (2) Data from Local System Data from Network Shared Drive Data from Removable Media Data Staged (2) Email Collection (2) Input Capture (4) Man-in-the-Browser Man-in-the-Middle (1) Screen Capture Video Capture	Application Layer Protocol (4) Automated Exfiltration Communication Through Removable Media Data Encoding (2) Data Obfuscation (2) Dynamic Resolution (3) Encrypted Channel (2) Fallback Channels Ingress Tool Transfer Multi-Stage Channels Non-Application Layer Protocol Non-Standard Port Protocol Tunneling Proxy (4) Remote Access Software Traffic Signaling (1) Web Service (1)	Account Access Removal Data Destruction Data Encrypted for Impact Data Manipulation (2) Defacement Disk Wipe (2) Endpoint Denial of Service (4) Exfiltration Over Alternative Protocol (1) Exfiltration Over C2 Channel Exfiltration Over Other Network Medium (1) Exfiltration Over Physical Channel (1) Exfiltration Over Web Service (2) Scheduled Transfer Transfer Data to Cloud Account Resource Hijacking Service Stop System Shutdown/Reboot	

Obrázok 4 ATT&CK matica pre doménu Enterprise (8)

1.4.1 Technologické domény

MITRE ATT&CK je organizovaný do technologických domén. Tieto domény predstavujú ekosystém, v ktorom protivník pôsobí. MITRE zatiaľ definovalo tri technologické domény. Doména Enterprise ktorá reprezentuje podnikové siete a cloudové technológie. Doména Mobile pre mobilné komunikačné zariadenia a ICS pre priemyselné riadiace systémy. Pre každú z týchto domén ATT&CK definuje viacero platforiem. Platforma je systém alebo aplikácia v ktorom protivník pôsobí. Môže to byť napríklad Microsoft Windows (8).

Každá technika a podtechnika sa môže vzťahovať na viacero platforiem. Platformy pre doménu Enterprise sú Linux, macOS, Windows, AWS, Azure, GCP, SaaS, Office 365 a Azure AD. Platformy pre doménu Mobile sú Android a iOS (8).

1.4.2 Taktiky

Taktika je taktický cieľ protivníka, je to dôvod na vykonanie akcie. Taktiky slúžia ako užitočné kontextové kategórie pre jednotlivé techniky. Pokrývajú štandardné notácie pre veci ktoré protivníci robia počas ich operácie. Je to napríklad získavanie informácií alebo spúšťanie súborov. Dá sa povedať, že taktiky sa považujú za značky. V závislosti od rôznych výsledkov, ktoré je možné dosiahnuť použitím techniky je technika alebo podtechnika označená jednou alebo viacerými taktikami (8).

Každá taktika obsahuje definíciu ktorá ju popisuje. Definícia slúži ako návod, aby bolo jasné aké techniky do danej taktiky patria. Napríklad, máme definované vykonávanie ako taktiku. Táto taktika predstavuje techniky a podtechniky, ktorých výsledkom je vykonávanie protivníkom riadeného kódu na lokálnom alebo vzdialenom systéme. Taktika vykonávania sa často používa v spojení s taktikou počiatočného prístupu, čo znamená spustenie kódu potom čo bol získaný prístup (8).

Podľa potreby môžeme definovať ďalšie taktiky aby sme vedeli presnejšie popísať ciele protivníka. Aplikácie metodológie modelovania ATT&CK pre iné oblasti si môžu vyžadovať nové alebo odlišné taktiky ktorým priradíme techniky, napriek tomu že sa nové definície taktík môžu do určitej miery prekrývať s existujúcimi (8).

1.4.3 Techniky a podtechniky

Techniky predstavujú ako protivník dosiahne cieľ taktiky vykonaním nejakej akcie. Techniky môžu tiež predstavovať čo protivník získa vykonaním nejakej akcie. Toto je užitočné napríklad pri taktike objavovania, pretože techniky zdôrazňujú aký typ informácií sa protivník snaží získať pri konkrétnej akcii (8).

Podtechniky ďalej rozdeľujú správanie opísané technikami na konkrétnejšie opisy toho, ako sa správanie používa na dosiahnutie cieľa. Napríklad, pri technike získavania prihlasovacích údajov používateľov operačného systému existuje pod touto technikou niekoľko špecifickejších správaní ktoré môžeme opísať ako podtechniky. Je to napríklad prístup k pamäti LSASS alebo manažéra bezpečnostných účtov (8).

Môže byť veľa spôsobov alebo techník ako dosiahnuť cieľ taktiky. Preto je pre každú taktiku priradených množstvo techník. Podobne môže existovať viacero spôsobov vykonávania techniky, preto môže byť k tejto technike vytvorených viacero podtechník (8).

1.4.4 Procedúry

Procedúry sú ďalšou dôležitou časťou konceptu TTP. Sú to špecifické implementácie ktoré protivníci použili pre techniku alebo podtechniku. O procedúrach treba poznamenať dva dôležité aspekty. Procedúra označuje ako protivník používa techniku a podtechniku a tiež môže zahŕňať viacero techník a podtechník. Procedúry zahŕňajú aj špecifické nástroje ktoré sa použili a tiež ako sa použili (8).

1.4.5 Skupiny

Sú to známi protivníci, ktorí sú sledovaný verejnými a súkromnými organizáciami. Informácie o nich sú nahlasované pomocou správ o hrozbách. Skupiny sú definované ako pomenované skupiny hrozieb, skupiny aktérov alebo kampane, ktoré zvyčajne predstavujú cielenú aktivitu s pretrvávajúcou hrozbou. ATT&CK sa primárne zameriava na skupiny APT, aj keď môže zahŕňať aj iné skupiny, ako sú finančne motivovaní aktéri. Skupiny používajú techniky priamo alebo využívajú softvér, ktorý tieto techniky implementuje (8).

1.4.6 Softvér

Protivníci používajú počas prienikov rôzne typy softvéru. Softvér môže predstavovať inštanciu techniky alebo podtechniky. V rámci ATT&CK je ich potrebné kategorizovať, aby ich bolo možné používať pri príkladoch ako sa používajú techniky. Softvér je rozdelený do dvoch hlavných kategórií, nástroje a malware (8).

Nástroje sú komerčný, open-source, vstavaný alebo verejne dostupný softvér, ktorý môže byť použitý obrancom, penetračným testerom, red teamerom alebo protivníkom. Táto kategória zahŕňa softvér, ktorý sa vo všeobecnosti nenachádza v enterprise systémoch, ako aj bežne dostupný softvér ktorý je už súčasťou operačného systému. Príklady takéhoto softvéru sú napríklad PsExec, Metasploit, Mimikatz, ako aj nástroje systému Windows, ako sú Net, netstat, Tasklist atď (8).

Malware je komerčný, closed-source alebo open-source softvér určený na použitie protivníkom na škodlivé účely. Je to napríklad malware PlugX, CHOPSTICK atď (8).

1.4.7 Mitigácie

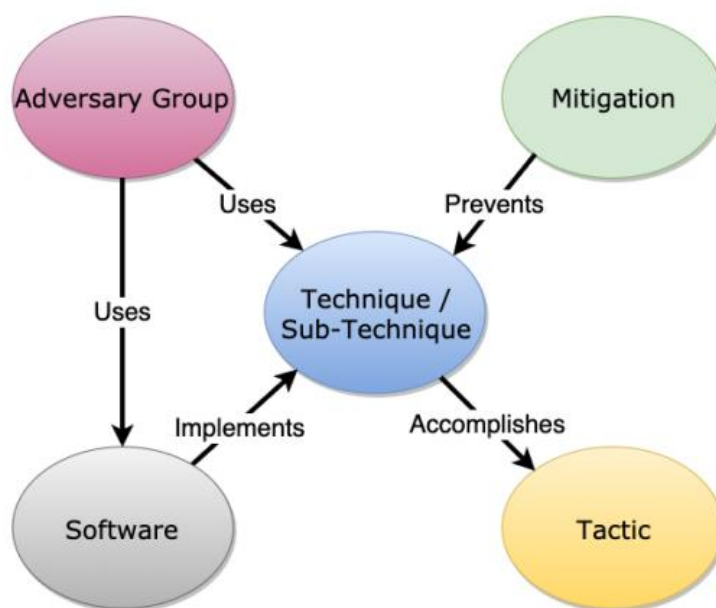
Mitigácie v ATT&CK predstavujú bezpečnostné koncepty a triedy technológií, ktoré je možné použiť na zabránenie úspešnému vykonaniu techniky alebo podtechniky. Pre každú doménu existuje množstvo mitigácií. Pre doménu Enterprise je to napríklad izolovanie aplikácií, zálohovanie dát, prevencia spúšťania súborov a segmentácia siete. Mitigácie sú nezávislé od produktov dodávateľa a popisujú iba kategórie alebo triedy technológií, nie konkrétne riešenia (8).

Zmiernenia sú reprezentované objektmi podobnými skupinám a softvéru, kde vzťahy predstavujú, ako môže mitigácia zmierniť techniku alebo podtechniku. ATT&CK pre doménu Mobile bola prvou znalostnou bázou, ktorá začala používať objektový formát s mitigáciami. V súčasnosti majú domény Enterprise aj Mobile svoje vlastné skupiny mitigácií s minimálnym prekrytím (8).

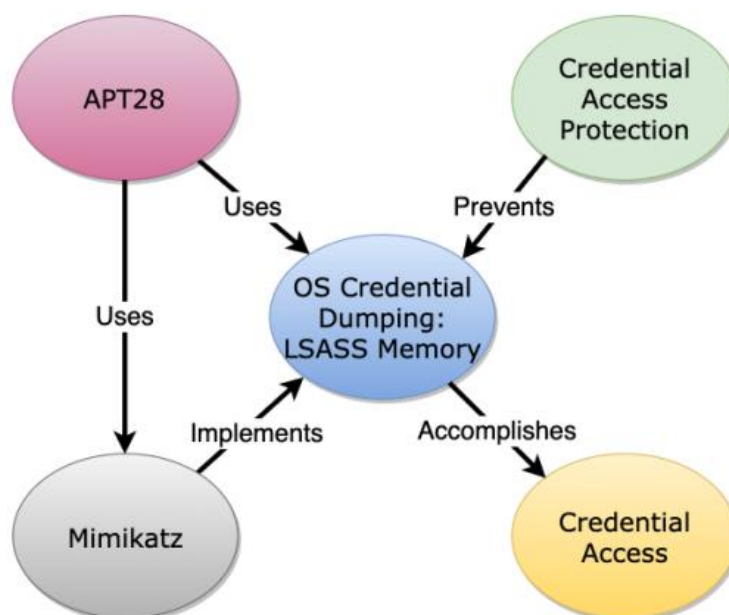
1.4.8 ATT&CK vzťahy medzi hlavnými komponentami

Každý hlavný komponent ATT&CK nejakým spôsobom súvisí s inými hlavnými komponentmi. Tieto vzťahy sú znázornené na diagrame. Na druhom diagrame je znázornený príklad týchto vzťahov medzi inštanciami hlavných komponentov (8).

Z diagramov vieme vyčítať že techniky a podtechniky ktoré predstavujú akcie uskutočňujú taktiky, ktoré predstavujú ciele. Mitigácie sa snažia týmto technikám a podtechnikám zabrániť vo vykonaní danej akcie. Skupiny protivníkov používajú pri ich útokoch techniky a podtechniky. Taktiež používajú softvér, ktorý podobne implementuje techniky a podtechniky (8).



Obrázok 5 ATT&CK vzťahy medzi hlavnými komponentami (8)



Obrázok 6 ATT&CK príklad vzťahov medzi inštanciami hlavných komponentov (8)

1.5 Spôsoby prístupu k MITRE ATT&CK dátam

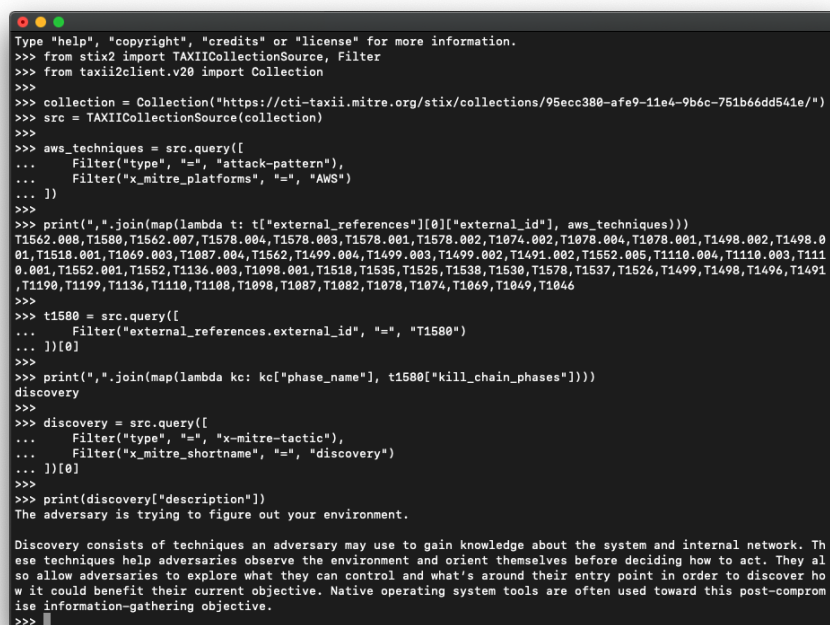
MITRE ATT&CK poskytuje 5 spôsobov ktorými dokážeme pristupovať k dátam. Sú to STIX, Excel, Navigator, Workbench a Python Utility. V tejto podkapitole priblížime všetky tieto spôsoby a opíšeme aké sú ich výhody (9).

STIX je jazyk a serializačný formát používaný na výmenu CTI. Je to strojovo čitateľný formát poskytujúci prístup k znalostnej báze ATT&CK. Súbor údajov ATT&CK je dostupný vo verziách STIX 2.0 a STIX 2.1. STIX poskytuje najpodrobnejšiu reprezentáciu údajov ATT&CK a všetky ostatné reprezentácie sú od neho odvodené (9).

STIX je vhodné použiť v niekoľkých nasledujúcich prípadoch použitia. Ak máme automatizované pracovné postupy, ktoré potrebujú získavať ATT&CK dáta. Ak chceme pomocou Pythonu ušetriť čas automatizáciou, alebo chceme vykonávať pokročilé dotazy. Ak chceme aby naše pracovné postupy boli aktuálne s vyvíjajúcou sa bázou znalostí. Ak chceme rozšíriť súbor údajov ATT&CK o vlastný obsah a používať tento vlastný obsah pomocou nástrojov ATT&CK (9).

S ATT&CK STIX sa najjednoduchšie pracuje v programovacom jazyku Python pomocou knižnice stix2. Avšak pretože je STIX reprezentovaný vo formáte JSON, dokážu s jeho surovým obsahom pracovať aj iné programovacie jazyky (9).

Dáta ATT&CK STIX je možné získať aj priamo z GitHub repozitára, alebo pomocou oficiálneho servera ATT&CK TAXII. TAXII je aplikačný protokol na výmenu CTI cez HTTPS. ATT&CK TAXII poskytuje API pre prístup k znalostnej báze ATT&CK STIX. Na obrázku môžeme vidieť príklad práce s knižnicou STIX (9).



```
Type "help", "copyright", "credits" or "license" for more information.
>>> from stix2 import TAXIICollectionSource, Filter
>>> from taxii2client.v20 import Collection
>>>
>>> collection = Collection("https://cti-taxii.mitre.org/stix/collections/95ecc380-afe9-11e4-9b6c-751b66dd541e/")
>>> src = TAXIICollectionSource(collection)
>>>
>>> aws_techniques = src.query([
...     Filter("type", "=", "attack-pattern"),
...     Filter("x_mitre_platforms", "=", "AWS")
... ])
>>>
>>> print(".".join(map(lambda t: t["external_references"][0]["external_id"], aws_techniques)))
T1562.008,T1580,T1562.007,T1578.004,T1578.003,T1578.001,T1578.002,T1074.002,T1078.004,T1078.001,T1498.002,T1498.0
01,T1518.001,T1069.003,T1087.004,T1562,T1499.004,T1499.003,T1499.002,T1491.002,T1552.005,T1110.004,T1110.003,T111
0.001,T1552.001,T1552,T1136.003,T1098.001,T1518,T1535,T1525,T1538,T1530,T1578,T1537,T1526,T1499,T1498,T1496,T1491
,T1190,T1199,T1136,T1110,T1188,T1098,T1087,T1082,T1078,T1074,T1069,T1049,T1046
>>>
>>> t1580 = src.query([
...     Filter("external_references.external_id", "=", "T1580")
... ])[0]
>>>
>>> print(".".join(map(lambda kc: kc["phase_name"], t1580["kill_chain_phases"])))
discovery
>>>
>>> discovery = src.query([
...     Filter("type", "=", "x-mitre-tactic"),
...     Filter("x_mitre_shortname", "=", "discovery")
... ])[0]
>>>
>>> print(discovery["description"])
The adversary is trying to figure out your environment.

Discovery consists of techniques an adversary may use to gain knowledge about the system and internal network. Th
ese techniques help adversaries observe the environment and orient themselves before deciding how to act. They al
so allow adversaries to explore what they can control and what's around their entry point in order to discover ho
w it could benefit their current objective. Native operating system tools are often used toward this post-comprom
ise information-gathering objective.
>>>
```

Obrázok 7 Príklad práce s Python knižnicou stix2 (9)

Ďalšou možnosťou ako pristupovať k súboru dát ATT&CK je pomocou Excel tabuliek. Tieto tabuľky sú zostavené z ATT&CK STIX a poskytujú pre ľudí prístupnejší pohľad do znalostnej bázy, pričom zároveň tiež podporujú základné možnosti prehľadávania dát (9).

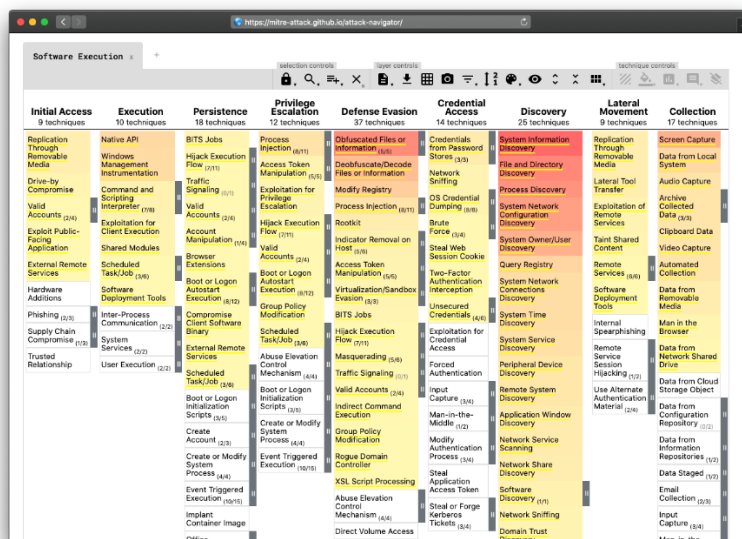
Excel tabuľky je vhodné použiť v niekoľkých nasledujúcich prípadoch použitia. Ak chceme rýchlo triediť, filtrovať a dopytovať súbor údajov v známom používateľskom rozhraní. Ak chceme skúmať obsah súboru údajov bez toho aby sme museli používať webovú stránku ATT&CK. Ak nie je pre nás pohodlné pracovať v programovacom jazyku Python alebo inom programovacom jazyku pre prácu so STIX reprezentáciou (9).

Excel tabuľková reprezentácia ATT&CK súboru údajov zahŕňa hlavné tabuľky, ktoré obsahujú všetky typy objektov a tiež samostatné tabuľky pre každý typ objektu. Tabuľky jednotlivých typov rozdeľujú vzťahy (napríklad príklady procedúr ktoré spájajú skupiny a techniky) do samostatných hárkov zatiaľ čo hlavná tabuľka obsahuje všetky typy vzťahov v jednom hárku. Iné rozdiely medzi ich reprezentáciami nie sú. Na obrázku môžeme vidieť príklad Excel tabuľky pre ATT&CK (9).

ID	name	description	url	created	last modified	version	contributors	associated groups	assoc
G0099	APT-C-36	[APT-C-36](https://attack.mitre.org/)	https://attack.mitre.org/	05 May 2020	14 October 2020	1.0	Jose Luis Sánchez Mar	Blind Eagle	(Citat
G0006	APT1	[APT1](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	22 October 2020	1.3		Comment Crew, Comi	(Citat
G0005	APT12	[APT12](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	30 March 2020	2.1		ONSCALC, DynCalc, IK	(Citat
G0023	APT16	[APT16](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	12 October 2020	1.1			
G0025	APT17	[APT17](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	13 October 2020	1.1		Deputy Dog	(Citat
G0026	APT18	[APT18](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	30 March 2020	2.1		Dynamite Panda, TG-C	(Citat
G0073	APT19	[APT19](https://attack.mitre.org/)	https://attack.mitre.org/	17 October 2018	20 June 2020	1.3	Darren Spruell, FS-ISA	Codoss, Coi	(Citat
G0007	APT28	[APT28](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	06 October 2020	3.0	Drew Church, Splunk;	Fancy Bear, Group 74,	(Citat
G0016	APT29	[APT29](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	22 October 2020	1.4		Cozy Bear, CozyDuke,	(Citat
G0022	APT3	[APT3](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	30 March 2020	1.3		Buckeye, Gothic Pand	(Citat
G0013	APT30	[APT30](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	29 July 2020	1.1			
G0050	APT32	[APT32](https://attack.mitre.org/)	https://attack.mitre.org/	14 December 2017	29 June 2020	2.3	Romain Dumont, ESEI	APT-C-00, OceanLotus	(Citat
G0064	APT33	[APT33](https://attack.mitre.org/)	https://attack.mitre.org/	18 April 2018	01 July 2020	1.3		Elfin, HOLMIUM	(Citat
G0067	APT37	[APT37](https://attack.mitre.org/)	https://attack.mitre.org/	18 April 2018	21 October 2020	1.5	Valerii Marchuk, Cybe	Group123, Reaper, Sc	(Citat
G0082	APT38	[APT38](https://attack.mitre.org/)	https://attack.mitre.org/	29 January 2019	30 March 2020	1.2			
G0087	APT39	[APT39](https://attack.mitre.org/)	https://attack.mitre.org/	19 February 2019	11 August 2020	2.3			
G0096	APT41	[APT41](https://attack.mitre.org/)	https://attack.mitre.org/	23 September 2019	24 June 2020	1.1		Chafar	Active
G0001	Axiom	[Axiom](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	30 March 2020	1.2		Group 72	(Citat
G0060	BRONZE BUTLER	[BRONZE BUTLER](https://attack.mitre.org/)	https://attack.mitre.org/	16 January 2018	25 June 2020	1.1	Trend Micro Incorpor	REDBALDKNIGHT, Tick	(Citat
G0063	BlackOasis	[BlackOasis](https://attack.mitre.org/)	https://attack.mitre.org/	18 April 2018	17 October 2018	1.0			
G0098	BlackTech	[BlackTech](https://attack.mitre.org/)	https://attack.mitre.org/	05 May 2020	06 May 2020	1.0	Tatsuya Daitoku, Cyber	Defense Institute, Inc.	
G0108	BlueMockingbird	[BlueMockingbird](https://attack.mitre.org/)	https://attack.mitre.org/	26 May 2020	25 June 2020	1.0	Tony Lambert, Red Canary		
G0008	Carbanak	[Carbanak](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	28 March 2020	1.1	Anastasios Pingios	Anunak, Carbon Spide	(Citat
G0058	Charming Kitten	[Charming Kitten](https://attack.mitre.org/)	https://attack.mitre.org/	16 January 2018	04 July 2020	1.0			
G0114	Chimera	[Chimera](https://attack.mitre.org/)	https://attack.mitre.org/	24 August 2020	05 October 2020	1.0			
G0003	Cleaver	[Cleaver](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	15 October 2020	1.2		TG-2889, Threat Grou	(Citat
G0080	Cobalt Group	[Cobalt Group](https://attack.mitre.org/)	https://attack.mitre.org/	17 October 2018	23 June 2020	1.2		Cobalt Gang, Cobalt S	(Citat
G0052	CopyKittens	[CopyKittens](https://attack.mitre.org/)	https://attack.mitre.org/	16 January 2018	31 March 2020	1.3			
G0070	Dark Caracal	[Dark Caracal](https://attack.mitre.org/)	https://attack.mitre.org/	17 October 2018	03 June 2020	1.2			
G0079	DarkHydru	[DarkHydru](https://attack.mitre.org/)	https://attack.mitre.org/	17 October 2018	15 May 2020	1.2		Oleg Skulkin, Group-IB	
G0105	DarkVishnya	[DarkVishnya](https://attack.mitre.org/)	https://attack.mitre.org/	15 May 2020	15 May 2020	1.0			
G0012	Darkhotel	[Darkhotel](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	30 March 2020	1.2			
G0009	Deep Panda	[Deep Panda](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	17 April 2020	1.2	Andrew Smith, @jakk	Black Vine, KungFu Kil	(Citat
G0017	DragonOK	[DragonOK](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	22 March 2019	1.0			
G0035	Dragonfly	[Dragonfly](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	14 October 2020	2.0		Crouching Yeti, Energi	(Citat
G0074	Dragonfly 2.0	[Dragonfly 2.0](https://attack.mitre.org/)	https://attack.mitre.org/	17 October 2018	15 October 2020	1.3		Berserk Bear, DYMALL	(Citat
G0031	Dust Storm	[Dust Storm](https://attack.mitre.org/)	https://attack.mitre.org/	31 May 2017	22 March 2019	1.0			
G0066	Elderwood	[Elderwood](https://attack.mitre.org/)	https://attack.mitre.org/	18 April 2018	30 March 2020	1.1	Valerii Marchuk, Cybe	Beijing Group, Eldew	(Citat

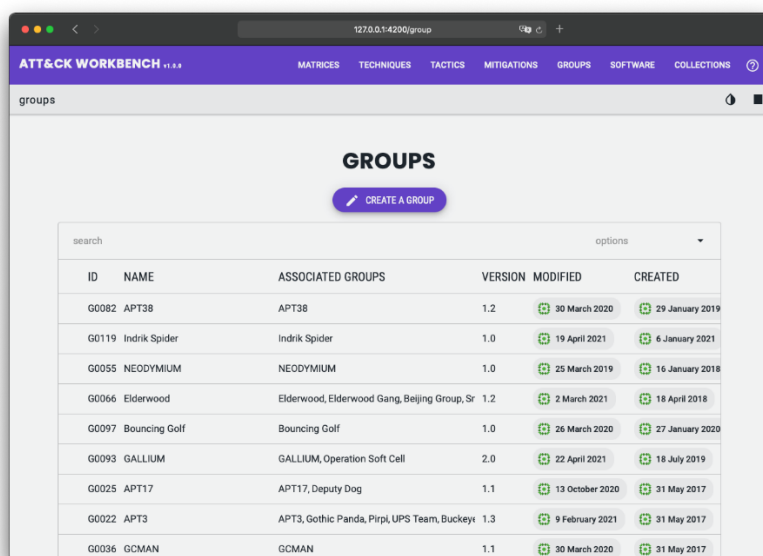
Obrázok 8 ATT&CK Excel tabuľka (9)

ATT&CK Navigator je webový nástroj na anotovanie a skúmanie ATT&CK matíc. Dá sa použiť na vizualizáciu obranného krytia, red/blue team plánovania, frekvencie zistených techník a viac (9).



Obrázok 9 ATT&CK Navigator (9)

ATT&CK Workbench je aplikácia umožňujúca používateľom skúmať, vytvárať, anotovať a zdieľať rozšírenia vedomostnej bázy ATT&CK (9).



Obrázok 10 ATT&CK Workbench (9)

ATT&CK poskytuje množstvo nástrojov pre programovací jazyk Python na získavanie a spracovávanie dát z ATT&CK. Tieto skripty sú užitočné nástroje a tiež môžu slúžiť ako príklad práce s ATT&CK pomocou programov. Okrem samotných vytvorených

skriptov je možné použiť Python knižnicu mitreattack-python pre vytvorenie nových vlastných skriptov. Táto knižnica obsahuje niekoľko modulov (9).

Modul navlayers je zbierka nástrojov pre prácu s vrstvami ATT&CK Navigator. Pre tieto vrstvy poskytuje importovanie, exportovanie a manipuláciu s nimi. Je možné ich čítať zo súborového systému alebo Python slovníkov. Ďalej ich umožňuje kombinovať, upravovať a potom exportovať do Excel tabuliek alebo SVG obrázkov (10).

Modul attackToExcel je zbierka nástrojov na konverziu údajov ATT&CK STIX do Excel tabuliek. Tiež poskytuje prístup k Pandas Dataframom ktoré predstavujú súbor údajov a dajú sa použiť pri analýze (10).

Modul collections je zbierka nástrojov pre prácu s ATT&CK Collections a ich indexami. Poskytuje funkcie na konverziu a sumarizáciu dát v kolekciách a ich indexoch ako aj na generovanie kolekcie z nespracovaného STIX vstupu (10).

Modul diffStix umožňuje vytvárať hlásenia zmien medzi dvoma verziami STIX2 kolekcií ktoré reprezentujú obsah ATT&CK (10).

1.6 Hybridná analýza

Hybridná analýza je inteligentná kombinácia statickej a dynamickej analýzy. Je to technológia, ktorá integruje dáta extrahované za behu z dynamickej analýzy do algoritmu statickej analýzy na detekciu škodlivého správania alebo funkcií. Dynamické pomocné dáta často pripomínajú snímky pamäte, referenčné hodnoty adresy a pridávajú ich ako vstup do sofistikovaného nástroja pre statickú analýzu. Napríklad, ak nečinná kódová sekvencia vykoná nepriame volanie, nebolo by možné vyriešiť adresu volanej funkcie bez poznania hodnoty ktorá je načítaná z miesta v pamäti v okamihu vykonávania. Aj keby sme túto hodnotu poznali, nebolo by možné priradiť adresu volanej funkcie k systémovému volaniu, v prípade ak mapovanie pamäťových odkazov na informácie o symboloch nie je dostupné pre konkrétne prostredie vykonávania (11).

Statická analýza je analýza kódu bez vykonania cieľového dátového obsahu. Cieľový kód, teda vstupné dáta analýzy, môže byť skompilovaný binárny súbor alebo ľuďmi čitateľný formát, ako je zdrojový kód programu, súbory skriptovacieho jazyka alebo akýkoľvek iný typ reprezentácie strojového kódu. Statická analýza sa dá definovať ako metóda, ktorá skúma kód pri absencii vstupných údajov a bez spustenia tohto kódu.

Dokáže odhaliť potenciálne narušenia bezpečnosti, run-time chyby a logické nezrovnalosti (11).

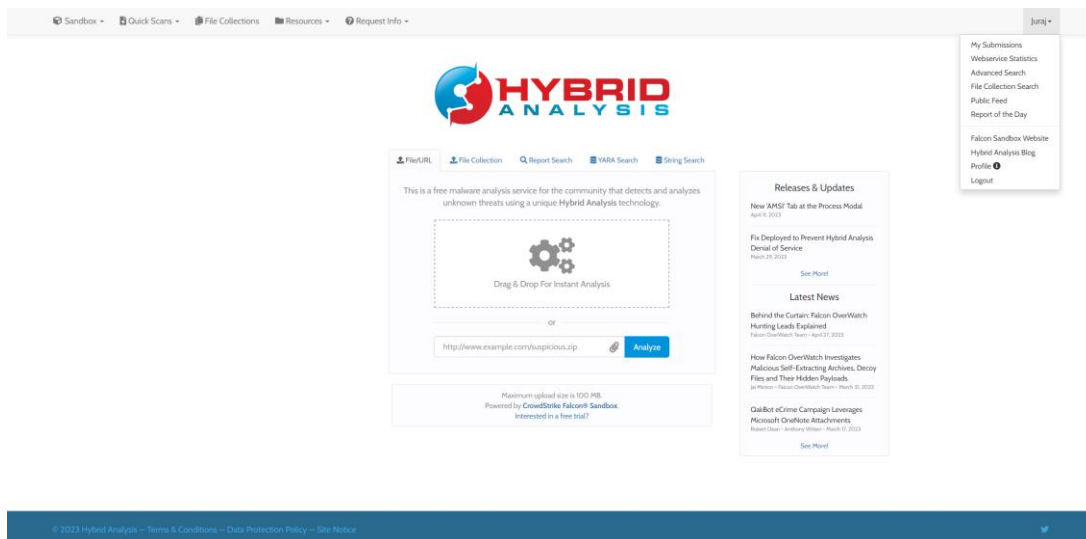
Dynamická analýza je analýza kódu počas vykonávania alebo emulácie cieľového dátového obsahu. Techniky sú zvyčajne implementované pozorovacími nástrojmi. Napríklad detekcia škodlivého správania, detekcia narušenia a pozorovanie výkonu. Používajú sa aj nástroje na analýzu správania ako napríklad sandbox systémy. Jedinou známou technikou používanou na vykonávanie dynamickej analýzy je inštrumentácia cieľového kódu alebo jeho hostiteľa, aby sa profilovalo správanie tohto kódu. Inštrumentácia sa týka techník, ktoré vkladajú dodatočný kód na účely analýzy do cieľového kódu s cieľom merať výkonnosť klienta, detegovať chyby alebo zachytávať tok kódu s cieľom analyzovať určité vzorce správania. Pri analýze škodlivého softvéru sú vzory správania často najzaujímavejšie (11).

Web stránka Hybrid-Analysis.com je bezplatná služba analýzy škodlivého softvéru. Pomocou tejto služby môžeme odosielať súbory na hĺbkovú hybridnú, teda statickú a dynamickú analýzu. Analýza je vykonávaná prostredníctvom Falcon Sandboxu (12).

Falcon Sandbox je vývojová platforma pre analýzy škodlivého softvéru. Má veľmi agilnú architektúru. Môže byť implementovaný ako rozsiahly systém, ktorý automaticky spracováva státisíce súborov alebo ako webová služba pre reakciu na incidenty, forenznú analýzu alebo ako samoobslužný enterprise portál. Vďaka svojmu jednoduchému rozhraniu a veľkým množstvám možností integrácie s inými poskytovateľmi technológií, bez problémov obohacuje pracovný tok reakcie na incidenty SOC a bezpečnostné balíky nástrojov. Podporuje veľké množstvo formátov súborov. V súčasnosti je používaný po celom svete tímami SOC, CERT, DFIR, foreznými laboratóriami pre bezpečnosť informačných technológií, výskumníkmi a poskytovateľmi spravodajských služieb o hrozbách. Denne ho používajú aj viaceré indexy S&P 100, Fortune 500 a vládne agentúry USA (12).

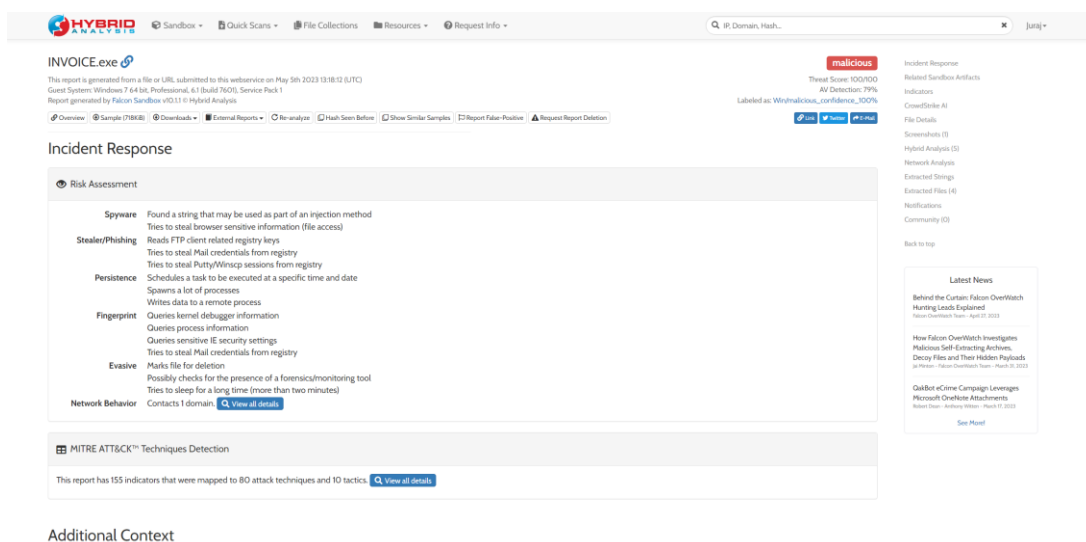
Na obrázku môžeme vidieť webové rozhranie web stránky Hybrid-Analysis.com. Táto web stránka nám umožňuje množstvo funkcií pre prácu s Falcon Sandboxom. Priamo cez stránku dokážeme vkladať súbory, kolekcie súborov a url web adresy na analýzu. Tiež dokážeme vyhľadávať správy vygenerované z vykonaných analýz. Umožňuje použiť aj rozšírené vyhľadávanie, vďaka ktorému môžeme vyhľadávať pomocou množstva filtrov. Ďalej si na stránke môžeme pozrieť takzvaný public feed, ktorý obsahuje 250 posledných

správ za 24 hodín. Ten je vo formáte JSON. Ďalšou zaujímavou funkciou je správa dňa. To je správa ktorú systém vyhodnotil ako najzaujímavejšiu v danom dni. Okrem uvedených funkcií stránka poskytuje mnoho ďalších (13).



Obrázok 11 Web stránka Hybrid-Analysis.com (13)

Na nasledujúcom obrázku vidíme ako vyzerá vygenerovaná správa súboru. Na stránke sa nachádza množstvo informácií. Sú tam napríklad informácie o súbore ako jeho názov, identifikátory, veľkosť a typ. Ďalej sú tam výsledky z analýzy vo forme zistených MITRE ATT&CK techník, indikátorov, skóre hrozby, vyhodnotenie hrozby súboru a mnoho ďalšieho (13).



Obrázok 12 Príklad správy analýzy z Hybrid-Analysis.com (13)

Pracovať s Falcon Sandboxom môžeme aj pomocou API. Táto API obsahuje veľké množstvo volaní. Dokážeme pomocou nej napríklad vkladať súbory a URL na analýzu,

vyhľadávať analýzy, súbory a správy v databáze pomocou rôznych filtrov, zisťovať systémové nastavenia atď. Metóda search/hash a search/ashes umožňujú vyhľadávať zhrnutie pre daný hash, search/terms umožňuje prehľadávať databázu pomocou filtrov a report/summary umožňuje vyhľadať správu. Metóda submit/file umožňuje vkladať súbory na analýzu a metóda quick-scan/file na rýchlu analýzu. Ďalšia zaujímavá metóda je feed/latest ktorou vieme získať JSON výpis posledných 250 správ za 24h (14).

Na to, aby sme vedeli túto API používať je potrebné na Hybrid-Analysis.com web stránke vygenerovať API kľúč. Tento API kľúč má rôzne úrovne autorizácie. Pokiaľ si ho vytvoríme sami, dostaneme obmedzenú úroveň autorizácie s ktorou vieme iba vyhľadávať a vkladať súbory na analýzu. Taktiež máme obmedzený počet API volaní ktoré dokážeme za určitý čas urobiť. Ak by sme potrebovali našu úroveň autorizácie zvýšiť, je možné o to cez túto web stránku požiadať prostredníctvom formulára (14).

1.7 Existujúce riešenie ontologického modelu pre bezpečnostnú doménu

Lukáš Hurtiš sa v jeho Bakalárskej práci venoval podobnej problematike použitia ontológie v bezpečnostnej doméne ako je tá naša. Zaoberal sa problematikou uchovávaní a zdieľaní dát z dynamickej analýzy škodlivého softvéru. Na dynamickú analýzu používal open source nástroj Cuckoo Sandbox. Výsledky analýz ukladal do ontológie. Túto ontológiu vytvoril na základe bezpečnostného štandardu MAEC. Pre samotný prevod z MAEC do ontológie vytvoril vlastnú aplikáciu (15).

MAEC umožňuje popisovať výsledky statickej aj dynamickej analýzy správania sa škodlivého softvéru. Je to podštandard STIX s ktorým má nejaké spoločné vlastnosti (15).

2 Opis riešenia

V tejto kapitole opíšeme naše riešenie problematiky. Opíšeme ako sme zostavili model ontológie a tiež skripty ktorými sme automatizovali napĺňanie dátami.

2.1 Ontológia pre MITRE ATT&CK

Ontológiu sme vytvorili pomocou programu Protégé. Pomenovali sme ju malware a dali sme jej jedinečné IRI <http://stufei/ontologies/malware>. Ontológia je uložená v súbore typu owl. Jazyk ktorým je zapísaná je rdf/xml. Vytvárali sme ju podľa štruktúry dát ktoré vieme získať oficiálnou MITRE ATT&CK knižnicou mitreattack-python.

Ontológia obsahuje päť tried ktoré predstavujú hlavné komponenty MITRE ATT&CK. Sú to triedy Tactic, Technique, Software, Group a Mitigation. Triedy sú navzájom poprepájané objektovými vzťahmi. Týmto vzťahom sa hovorí Object properties. Vytvorili sme dokopy štyri vzťahy.

Prvým vzťahom je vzťah hasSubTechnique. Je to vzťah ktorého domain aj range je trieda Technique. Pomocou tohto vzťahu prepájame techniky s ich podtechnikami.

Druhým vzťahom je vzťah mitigates. Jeho domain je trieda Mitigation a range je trieda Technique. Tento vzťah vyjadruje ktorá mitigácia mitiguje akú techniku.

Tretím vzťahom je vzťah usesSoftware. Domain tohto vzťahu je trieda Group a range je trieda Software. Vzťah vyjadruje, ktorá skupina používa aký softvér.

Štvrtým a zároveň posledným vzťahom medzi triedami je vzťah usesTechnique. Jeho doména sú triedy Group, Software, Tactic a range je trieda Technique. Vyjadruje ktoré skupiny, softvér a taktiky používajú aké techniky.

Všetky triedy našej ontológie majú priradené nejaké dáta. V prípade objektovo orientovaného programovania sa im hovorí atribúty, avšak v ontológiách to funguje inak. V ontológiách sú medzi objektom a dátami vzťahy, ktorým sa hovorí Data properties, teda dátové vzťahy. Domain dátového vzťahu je vždy trieda a range je napríklad string alebo integer, teda nejaký text alebo číslo.

Každá z našich tried má rovnakú hlavičku ktorá je tvorená zo siedmich dátových vzťahov. Všetky dáta sú typu xsd:string.

Vzťah hasId je jedinečný identifikátor inštancie techniky, taktiky, softvéru, skupiny alebo mitigácie. Každá trieda má špecifický formát identifikátora vďaka ktorému je poznat'

o akú triedu sa jedná. Na začiatku identifikátora je vždy veľké začiatkové písmeno triedy za ktorým nasledujú štyri číslice. Pre techniky je to T#### a podtechniky je to T####.###. Pre taktiky je to TA####, pre skupiny G####, pre softvér S#### a pre mitigácie M####.

Ďalší vzťah `hasName` je meno danej inštancie triedy. Vzťah `hasDescription` je popis inštancie, `hasUrl` je webová adresa na inštanciu, `wasCreated` a `wasLastModified` sú údaje o tom kedy bola inštancia vytvorená a naposledy upravená. Nakoniec máme vzťah `hasVersion` ktorý odkazuje na dáta o verzii inštancie, býva vo formáte MAJOR.MINOR.

Okrem hlavičky dát ktoré sme opísali, má každá trieda dátové vzťahy s ďalšími dátami. Tieto dáta sú medzi jednotlivými triedami zväčša rozličné. Navyše sa ešte líšia aj medzi doménami, teda dáta tried z domény Enterprise, Mobile a ICS nemajú medzi sebou rovnakú štruktúru dát. Napriek tomu že sa tieto vzťahy medzi doménami nezhodujú, urobili sme ich zjednotenie do jednej ontológie. Aby sme rozlíšili ktoré inštancie tried patria ku ktorej doméne, pridali sme ďalší dátový vzťah `hasDomain`, ktorý priradzuje k inštanci triedy jej doménu. Všetky dátové vzťahy v tejto ontológii sú funkcionálne, teda pre každú inštanciu triedy môžu existovať maximálne jeden krát. Týmto sme sa vyhli duplicitným vzťahom a nemôže sa stať že jedna inštancia triedy má napríklad viac ako jeden identifikátor. Pre väčšie množstvo dátových vzťahov v ontológií sme tieto vzťahy uviedli v prílohe.

2.2 Naplnenie ontológie MITRE ATT&CK dátami

Na to, aby sme do našej ontológie skopírovali dáta z bázy znalostí MITRE ATT&CK, sme napísali Python skript. Na obrázku môžeme vidieť aké knižnice sme potrebovali použiť.

```
import mitreattack.attackToExcel.attackToExcel as attackToExcel
import mitreattack.attackToExcel.stixToDf as stixToDf
from owlready2 import *
import pandas as pd
```

Obrázok 13 Python knižnice použité pri získavaní dát z MITRE ATT&CK

Na získavanie dát z MITRE ATT&CK sme použili knižnicu `mitreattack-python`. Získané dáta sú v triede `DataFrame`. `DataFrame` sú tabuľky z ktorých vieme jednoducho vyberať hodnoty pomocou názvov stĺpcov. Jednotlivé hodnoty v riadkoch sú dátového typu `str`.

Knižnicu owlready2 sme použili pre prácu s ontológiou. Ontológia je reprezentovaná pomocou tried s ktorými vieme manipulovať. Každá trieda ma atribúty ktoré zodpovedajú vzťahom v ontológii.

Knižnica pandas je pomocná knižnica, ktorá poskytuje funkcie pre prácu s triedou DataFrame.

Na začiatku skriptu načítame ontológiu, ktorú sme vytvorili v programe Protégé. Pre otvorenie použijeme funkciu z knižnice owlready2 a to `get_ontology(path).load()`. Vstupom do funkcie je cesta k owl súboru ontológie a výstupom je objekt triedy `Ontology` ktorý predstavuje našu ontológiu a dokážeme s ním pracovať.

```
ontology = get_ontology("file://malwareTemplate.owl").load()
```

Obrázok 14 Načítanie ontológie do objektu

Na konci skriptu keď s ontológiou dopracujeme ju uložíme pomocou funkcie `save(file, format)`. Vstupom funkcie je názov súboru do ktorého ju chceme uložiť a tiež v akom formáte ju chceme uložiť.

```
ontology.save(file="malware.owl", format="rdxml")
```

Obrázok 15 Uloženie ontológie do súboru

Predtým ako začneme vkladať dáta do ontológie, musíme ich najprv získať pomocou knižnice `mitreattack-python`. Najprv zavoláme funkciu rozhrania `attackToExcel` a to je funkcia `get_stix_data(domain)`. Táto funkcia získa ATT&CK STIX dáta pre doménu ktorú zadáme na vstup funkcie. Výstup funkcie sú dáta vo formáte `MemoryStore` objekt. S týmto objektom zatiaľ nedokážeme veľmi pracovať, a preto ho potrebujeme pretransformovať na objekt typu `DataFrame`. Na to použijeme niekoľko funkcií rozhrania `stixToDf`. Vstup funkcií je objekt typu `MemoryStore`, ktorý sme získali volaním funkcie `get_stix_data(domain)`. Výstup je slovník, ktorý obsahuje objekty typu `DataFrame`. V prípade funkcie `techniquesToDf(attackdata, domain)` je na vstupe funkcie aj doména.

Funkcia `tacticsToDf(attackdata)` vráti slovník obsahujúci iba jeden `DataFrame`, ktorý obsahuje dáta taktík.

Funkcia `techniquesToDf(attackdata, domain)` vráti slovník obsahujúci štyri `DataFrame` objekty. Prvý objekt je `DataFrame` techník. Druhý objekt je `DataFrame` vzťahov medzi softvérom alebo skupinou a technikou. Tento `DataFrame` sa volá príklad procedúr. Tretí objekt je `DataFrame` vzťahov medzi mitigáciou a technikou. Volá sa asociatívne mitigácie. Posledný `DataFrame` obsahuje citácie.

Funkcia `mitigationsToDf(attackdata)` vráti slovník obsahujúci tri `DataFrame` objekty. Prvý `DataFrame` objekt obsahuje dáta mitigácií. Druhý objekt je `DataFrame` vzťahov medzi mitigáciou a technikou. Volá sa riešené techniky. Posledný `DataFrame` objekt obsahuje citácie.

Funkcia `softwareToDf(attackdata)` vráti slovník obsahujúci štyri `DataFrame` objekty. Prvý `DataFrame` objekt obsahuje dáta softvéru. Druhý objekt je `DataFrame` vzťahov medzi skupinou a softvérom. Tento `DataFrame` sa volá asociované skupiny. Tretí objekt je `DataFrame` vzťahov medzi softvérom a technikou. Volá sa použité techniky. Posledný `DataFrame` objekt obsahuje citácie.

Funkcia `groupsToDf(attackdata)` vráti slovník obsahujúci štyri `DataFrame` objekty. Prvý `DataFrame` objekt obsahuje dáta skupín. Druhý objekt je `DataFrame` vzťahov medzi skupinou a softvérom. Tento `DataFrame` sa volá asociatívny softvér. Tretí objekt je `DataFrame` vzťahov medzi skupinou a technikou. Volá sa použité techniky. Posledný `DataFrame` objekt obsahuje citácie.

Niektoré objekty typu `DataFrame` z predchádzajúcich funkcií sú zhodné, majú iba rozdielne názvy. `DataFrame` asociatívnych mitigácií je rovnaký ako `DataFrame` riešených techník. Rovnaký je aj `DataFrame` asociatívne skupiny a asociatívny softvér.

Na obrázku vidíme volania vyššie uvedených funkcií pre doménu `Enterprise`. Rovnako voláme tieto funkcie aj pre domény `Mobile` a `ICS`, ale na vstupe niektorých funkcií meníme názov domény. Návrátové hodnoty funkcií si ukladáme do premenných aby sme ich mohli neskôr použiť na vstupoch funkcií, ktoré tieto dáta budú mapovať do ontológie.

```
# enterprise
attackdata = attackToExcel.get_stix_data("enterprise-attack")
tactics_data = stixToDf.tacticsToDf(attackdata)
mitigations_data = stixToDf.mitigationsToDf(attackdata)
groups_data = stixToDf.groupsToDf(attackdata)
software_data = stixToDf.softwareToDf(attackdata)
techniques_data = stixToDf.techniquesToDf(attackdata, "enterprise-attack")
```

Obrázok 16 Získanie dát z MITRE ATT&CK

Vytvorili sme niekoľko funkcií ktoré používame pre mapovanie dát na ontológiu. Takmer všetky tieto funkcie sú použiteľné pre všetky tri domény. Jedinou výnimkou je funkcia ktorá mapuje techniky. Každá doména má vlastnú takúto funkciu. Dokopy máme vytvorených sedem funkcií ktoré mapujú dátové vzťahy ku objektom, a štyri funkcie ktoré

mapujú objektové vzťahy medzi triedami. Taktiež sme vytvorili jednu pomocnú funkciu pre kontrolu existencie hodnôt.

Pomocná funkcia `is_valid_value(value)` skontroluje pomocou pandas funkcie `isna(value)` či hodnota `value` existuje. Taktiež kontrolujeme, či táto hodnota nie je prázdna porovnaním hodnoty s prázdny reťazcom. Vstupom funkcie je hodnota ktorú kontrolujeme a výstupom je `True` ak hodnota existuje a `False` ak neexistuje. Túto funkciu môžeme vidieť na obrázku.

```
def is_valid_value(value):  
    if pd.isna(value) or value == "":  
        return False  
    return True
```

Obrázok 17 Funkcia kontrolujúca existenciu dátovej hodnoty

Nasledujúcich sedem funkcií funguje tak, že zo slovníkov postupne vyberáme dáta a mapujeme ich na dátové vzťahy, ktoré pridelujeme inštanciam tried našej ontológie. Na vstupe funkcií sú vždy dáta vo forme slovníkov ktoré obsahujú objekty typu `DataFrame`. Funkcie nemajú žiadne návratové hodnoty.

Funkcie `map_tactics(tactics_data, domain)` a `map_mitigations(mitigation_data, domain)` mapujú dátové vzťahy techník a mitigácií. Okrem slovníka je na vstupe funkcie názov domény z ktorej sú dáta.

Funkcia `map_groups(group_data)` a funkcia `map_software(software_data)` mapujú dátové vzťahy skupín a softvéru. Tieto funkcie nemajú na vstupe názov domény, pretože majú rovnaké dáta naprieč všetkým trom doménami. Napriek tomu tieto funkcie fungujú rovnako ako predchádzajúce dve.

`Mobile_map_techniques(techniques_data)`, `ics_map_techniques(techniques_data)` a `enterprise_map_techniques(techniques_data)` sú funkcie ktorými mapujeme dátové vzťahy techník. Nakoľko sa štruktúra dát techník medzi doménami líši, potrebujeme mať ku každej doméne samostatné funkcie. Okrem dátových vzťahov v tejto funkcii mapujeme aj objektový vzťah medzi taktikami a technikami. Toto mapovanie sme zakomponovali do jednej funkcie aby sme zjednodušili a zrýchlili náš algoritmus.

Na obrázku môžeme vidieť ako postupne voláme jednotlivé mapovacie funkcie pre dátové vlastnosti domény `Enterprise`. Rovnakým spôsobom ich voláme aj pre ostatné domény, meníme iba parameter funkcií a funkciu pre mapovanie techník.

```

map_tactics(tactics_data, "enterprise")
map_groups(groups_data)
map_mitigations(mitigations_data, "enterprise")
map_software(software_data)
enterprise_map_techniques(techniques_data)

```

Obrázok 18 Volanie mapovacích funkcií pre dátové vzťahy domény Enterprise

Na nasledujúcom obrázku je úryvok kódu funkcie `map_tactics(tactics_data, domain)`. Na začiatku zo vstupného slovníka vyberieme DataFrame dát taktík zavolaním funkcie `get("tactics")` na slovníku. Z týchto dát vyberieme hodnoty jednotlivých stĺpcov zavolaním funkcie `get(column_name)` na DataFrame objekte. Tieto hodnoty predstavujú naše dátové vzťahy. Uložíme ich do premenných vo formáte pola numpy zavolaním funkcie `to_numpy()` na DataFrame objekte. Následne postupne iterujeme cez hodnoty jednotlivých stĺpcov, kontrolujeme ich existenciu pomocou našej pomocnej funkcie `is_valid_value(value)` a ak táto kontrola skončí vyhodnotením `True` tak vytvoríme pre inštanciu taktiky tento vzťah. Samozrejme na začiatku iterácie vždy musíme najprv vytvoriť inštanciu taktiky. Pri vytvorení jej najprv pridelíme jednoznačný identifikátor a doménu z ktorej je. Jednoznačný identifikátor nie je iba dátový vzťah, ale je aj identifikátor inštancie objektu ktorý zadávame pri vytváraní inštancie. Rovnakým spôsobom fungujú aj naše ostatné funkcie ktoré mapujú dátové vzťahy.

```

def map_tactics(tactics_data, domain):
    tactics = tactics_data.get("tactics")
    id_col = tactics.get("ID").to_numpy()
    name_col = tactics.get("name").to_numpy()
    description_col = tactics.get("description").to_numpy()
    url_col = tactics.get("url").to_numpy()
    created_col = tactics.get("created").to_numpy()
    last_modified_col = tactics.get("last modified").to_numpy()
    version_col = tactics.get("version").to_numpy()
    for i in range(len(tactics.index)):
        tactic = ontology.Tactic(id_col[i])
        tactic.hasId = id_col[i]
        tactic.hasDomain = domain
        if is_valid_value(name_col[i]):
            tactic.hasName = name_col[i]

```

Obrázok 19 Úryvok funkcie `map_tactics(tactics_data, domain)`

Vo funkciách ktoré implementujú dátové vzťahy techník na konci implementujú aj objektový vzťah medzi technikami a taktikami. Na obrázku vidíme úryvok kódu funkcie

`enterprise_map_techniques(techniques_data)`. Hodnotu názvov taktík rozdelíme podľa čiarky zavolaním funkcie `split(", ")` aby sme dostali zoznam názvov taktík. Následne cez jednotlivé názvy iterujeme. Pre každý názov taktiky vyhladáme jednu inštanciu objektu taktiky pomocou funkcie `search_one(type=ontology.Tactic, hasName=tactic)`. Na vstupe funkcie je aký typ objektu hľadáme a vzťah podľa ktorého hľadáme. Vyhľadávame podľa mena taktiky. Nájdenú taktiku priradíme objektovým vzťahom k technike. Keďže sa jedná o nefunkcionálny vzťah musíme ho priradovať pomocou funkcie `append(instance)`. Tento vzťah vyjadruje, ktorá taktika používa ktorú techniku.

```
if is_valid_value(tactics_col[i]):  
    for tactic in tactics_col[i].split(", "):  
        tactic_inst = ontology.search_one(type=ontology.Tactic, hasName=tactic)  
        tactic_inst.usesTechnique.append(technique)
```

Obrázok 20 Úryvok funkcie `enterprise_map_techniques(techniques_data)`

Následujúce štyri funkcie fungujú tak, že zo slovníkov postupne vyberáme dáta ktoré hovoria o vzťahoch medzi triedami a mapujeme ich na objektové vzťahy, ktoré prideliujeme inštanciám tried našej ontológie. Na vstupe funkcií sú dáta vo forme slovníkov ktoré obsahujú objekty typu `DataFrame`. Funkcie nemajú žiadne návratové hodnoty.

Funkcia `map_mitigation_technique_relation(mitigations_data)` mapuje objektové vzťahy medzi mitigáciami a technikami. Vzťah hovorí o tom ktorá mitigácia mitiguje ktorú techniku.

Funkcia `map_group_technique_relation(groups_data)` mapuje objektové vzťahy medzi skupinami a technikami. Vzťah hovorí o tom ktorá skupina používa ktoré techniky.

Funkcia `map_software_technique_relation(software_data)` mapuje objektové vzťahy medzi softvérom a technikami. Vzťah hovorí o tom ktorý softvér používa ktorú techniku.

Funkcia `map_group_software_relation(groups_data)` mapuje objektové vzťahy medzi skupinou a softvérom. Vzťah hovorí o tom ktorá skupina je asociovaná s ktorým softvérom.

Na obrázku je funkcia `map_software_technique_relation(software_data)`. Na začiatku zo vstupného slovníka vyberieme `DataFrame` objektových vzťahov zavolaním funkcie `get("techniques used")` na slovníku. Z týchto dát vyberieme hodnoty stĺpcov zdroja a cieľa zavolaním funkcie `get("source ID")` a `get("target ID")` na `DataFrame` objekte. Tieto hodnoty predstavujú naše objektové vzťahy. Uložíme ich do premenných vo formáte pola

numpy zavolaním funkcie `to_numpy()` na `DataFrame` objekte. Následne postupne iterujeme cez hodnoty jednotlivých stĺpcov a vytvárame objektové vzťahy. Pre vytvorenie tohto vzťahu musíme najprv vyhľadať inštancie podľa `source` a `target id`. Využili sme že ak vytvárame novú inštanciu pomocou identifikátora už existujúcej inštancie, pôvodné dáta inštancie zostanú a budú sa iba pridávať nové dáta. Nakoľko priradujeme nefunkcionálny objektový vzťah používame funkciu `append(instance)`. Rovnakým spôsobom fungujú aj naše ostatné funkcie ktoré mapujú objektové vzťahy.

```
def map_software_technique_relation(software_data):
    software_technique_relations = software_data.get("techniques used")
    software = software_technique_relations.get("source ID").to_numpy()
    techniques = software_technique_relations.get("target ID").to_numpy()
    for i in range(len(software_technique_relations.index)):
        software_inst = ontology.Software(software[i])
        technique_inst = ontology.Technique(techniques[i])
        software_inst.usesTechnique.append(technique_inst)
```

Obrázok 21 Funkcia `map_software_technique_relation(software_data)`.

Všetky vyššie uvedené funkcie voláme postupne v `main()` funkcii pre všetky domény. Do funkcií vkladáme podľa domény vhodné vstupné parametre.

2.3 Rozšírenie ontológie o hybridnú analýzu

Do našej malware ontológie ktorú sme vytvorili sme zlúčili ontológiu Lukáša Hurtiša. Ako sme už v analýze spomínali, je to ontológia na základe MAEC štandardu a niektoré prvky použijeme aj pre náš zápis dát. Pred tým ako sme ju zlúčili sme cez textový editor upravili IRI ontológie aby sa zhodovalo s tou našou. Následne sme ontológie zlúčili pomocou nástroja Protégé.

Triedy ktoré zo zlúčenej ontológie využijeme sú `url` a `file`. Dátové vzťahy ktoré využijeme sú identifikátory `md`, `sha1`, `sha256`, `sha512` a tiež vzťahy `size`, `key` a `value`.

Ďalej sme pridali niekoľko vlastných tried. Základnou triedou je `SampleSummary`. Táto trieda predstavuje správu analýzy a obsahuje všetky naše dáta z hybridnej analýzy. Objektový vzťah `hasSampleSummary` spája túto triedu správy s triedou `file` alebo `url`. Podľa toho ku ktorej triede je priradená vieme určiť či sa jedná o analýzu súboru alebo analýzu URL adresy.

K triede `SampleSummary` je pomocou objektových vzťahov pripojených deväť ďalších tried, ktoré majú k sebe pripojené ďalšie dátové prípadne objektové vzťahy. Sú to

triedy Submission, MachineLearningModel, CrowdStrikeAi, Certificate, ExtractedFile, FileMetadata, Process, MitreAttckSignature a Signature. Pripojené sú objektovými vzťahmi hasSubmission, hasMachineLearningModel, hasCrowdStrikeAi, hasCertificate, hasExtractedFile, hasFileMetadata, hasProcess, hasMitreAttckSignature a hasSignature.

Trieda MachineLearningModel má k sebe pripojenú triedu KeyStringValue objektovým vzťahom hasKeyStringValue. Trieda CrowdStrikeAi má k sebe pripojené triedy ExecutableProcessMemoryAnalysis a AnalysisRelatedUrl. Sú pripojené vzťahmi hasExecutableProcessMemoryAnalysis a hasAnalysisRelatedUrl.

Trieda Process má k sebe pripojených osem ďalších tried. Sú to triedy FileAccess, CreatedFile, RegistryAccess, Handle, Stream, ScriptCall, ProcessFlag a AmsiCall. Sú pripojené vzťahmi hasFileAccess, hasCreatedFile, hasRegistryAccess, hasHandle, hasStream, hasScriptCall, hasProcessFlag a hasAmsiCall. Z týchto tried má trieda Stream pripojenú triedu KeyStringValue vzťahom hasKeyStringValue. Podobne má aj trieda ScriptCall pripojenú triedu ScriptCallParameter vzťahom hasScriptCallParameter.

Všetky vyššie uvedené triedy majú množstvo dátových vzťahov, ktoré pre ich množstvo uvedieme v prílohe. Niektoré vzťahy sú funkcionálne a niektoré nie, podľa toho či jedna inštancia môže mať tento vzťah priradený viac ako jeden krát.

Vytvorili sme ešte jeden veľmi dôležitý objektový vzťah a to je vzťah hasAttckTechnique. Tento vzťah majú triedy MitreAttckSignature a Signature a spájajú ich s triedou Technique. Pomocou tohto vzťahu prepojíme signatúry s MITRE ATT&CK technikami a tým nám vznikne prepojenie medzi MITRE ATT&CK bázou znalostí a výsledkami hybridnej analýzy.

2.4 Naplnenie ontológie dátami z hybridnej analýzy

Pre získavanie dát z hybridnej analýzy sme napísali ďalší Python skript. Na obrázku môžeme vidieť knižnice ktoré sme použili. Na volanie Hybrid Analysis API sme použili Python knižnicu hybrid_analysis_api ktorá túto API zabaľuje do funkcií a tým nám uľahčuje prácu (16). Knižnicu owlready2 používame rovnako ako pri skripte MITRE ATT&CK pre prácu s ontológiou.

```
from owlready2 import *  
from hybrid_analysis_api import HybridAnalysis
```

Obrázok 22 Python knižnice použité pri získavaní dát z hybridnej analýzy

Na obrázku vidíme main() funkciu. Na začiatku získame posledných 250 správ zavolaním API obaľujúcej metódy feed_latest(). Následne výsledky rozdelíme na tri dávky, nakoľko API ktoré budeme volať jedným volaním dokáže spracovať iba sto správ. Z výsledkov vyberieme do pola jednoznačné identifikátory správ job_id. Tieto id následne dáme na vstup funkcie report_summary(job_ids). Volaním tejto funkcie získame podrobné správy ktoré namapujeme do našej ontológie pomocou funkcie map_data(all_data).

```
def main():
    result = ha.feed_latest()

    data1 = result['data'][0:100]
    data2 = result['data'][100:200]
    data3 = result['data'][200:]

    job_ids1 = [sample_summary.get('job_id') for sample_summary in data1]
    job_ids2 = [sample_summary.get('job_id') for sample_summary in data2]
    job_ids3 = [sample_summary.get('job_id') for sample_summary in data3]

    map_data(ha.report_summary(job_ids1))
    map_data(ha.report_summary(job_ids2))
    map_data(ha.report_summary(job_ids3))

    ontology.save(file="malware.owl", format="rdxml")
```

Obrázok 23 Main() funkcia skriptu získavania hybridnej analýzy

Funkcia map_data(all_data) má na vstupe dáta ktoré mapujeme do našej ontologickej triedy SampleSummary. V tejto funkcii mapujeme všetky objektové aj dátové vzťahy objektu SampleSummary ktorá predstavuje správu analýzy. Každá naša ontologická trieda ktorú v rámci tejto funkcie mapujeme má vlastnú mapovaciu funkciu. Sú to napríklad funkcie map_submission(data), map_machine_learning_model(data), map_process(data), map_crowdstrike_ai(data), map_mitre_attck(data) a ďalšie. Na vstupe funkcie sú dáta ktoré mapujeme a na výstupe je namapovaný objekt ontologickej triedy.

Na obrázku je úryvok funkcie map_data(all_data). Na začiatku iterujeme cez jednotlivé záznamy správ. Pre každý záznam vytvoríme inštanciu ontologickej triedy. Potom podľa názvov dátových elementov v správe tieto dáta priradíme pomocou vzťahov k inštanciám tried ontológie. Pri dátových vzťahoch nám stačí priamo namapovať získané hodnoty, ale pri objektových vzťahoch musíme najprv individuálne vytvoriť inštancie tried a následne na nich namapovať dáta. Pred mapovaním dát vždy najprv overíme či daný element existuje. Podobným spôsobom fungujú aj mapovacie funkcie jednotlivých tried.

```
def map_data(all_data):
    for data in all_data:
        sample_summary = ontology.SampleSummary(data.get('job_id'))
        sample_summary.hasJobId = data.get('job_id')
        if data.get('classification_tags'):
            sample_summary.hasClassificationTag = data.get('classification_tags')
        if data.get('tags'):
            sample_summary.hasTag = data.get('tags')
        for value in data.get('submissions'):
            sample_summary.hasSubmission.append(map_submission(value))
        for value in data.get('machine_learning_models'):
            sample_summary.hasMachineLearningModel.append(map_machine_learning_model(value))
        if data.get('crowdstrike_ai') is not None:
            sample_summary.hasCrowdStrikeAi = map_crowdstrike_ai(data.get('crowdstrike_ai'))
```

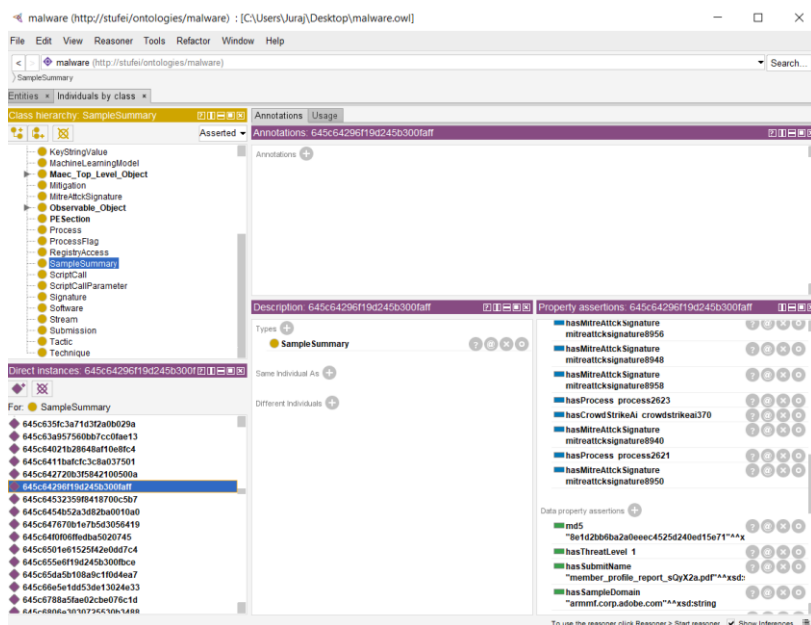
Obrázok 24 Úryvok funkcie map_data(all_data)

Na obrázku je koniec funkcie map_data(all_data). Na mapované správy teda inštancie tried SampleSummary sme priradili pomocou objektového vzťahu hasSampleSummary k triede file alebo url. Toto priradenie sme urobili podľa elementu url_analysis ktorý udáva, či bola analýza vykonaná na súbore alebo URL adrese.

```
if data.get('url_analysis') is not None and data.get('url_analysis'):
    url = ontology.url()
    url.hasSampleSummary.append(sample_summary)
else:
    file = ontology.file()
    file.hasSampleSummary.append(sample_summary)
```

Obrázok 25 Priradenie triedy SampleSummary k triede file alebo url

Na obrázku môžeme vidieť ako vyzerá naplnená ontológia v programe Protégé. Zobrazujeme vzťahy inštancie objektu SampleSummary.



Obrázok 26 Naplnená ontológia zobrazená v Protégé

3 Testovanie riešenia a zhodnotenie výsledkov

Riešenie sme testovali najmä kontrolou obsahu ontológie pomocou programu Protégé. Štruktúra ontológie sa musela zhodovať so štruktúrou MITRE ATT&CK a Hybrid Analysis.

Ďalší test správnej štruktúry ontológie sme vykonali písaním skriptov ktoré naplňajú ontológiu dátami. Pri písaní skriptu sa nám miestami vyskytli logické chyby v štruktúre ontológie alebo preklepy v názvoch tried či vzťahov, ktoré skončili chybovou hláškou. Všetky tieto chyby sme úspešne opravili. Skripty sme opakovane spúšťali v rôznych časoch aby sme dostali rôzne dáta z hybridnej analýzy, žiadne ďalšie chybové hlášky sa nevyskytli.

Nakoniec sme porovnali obsah dát ontológie pomocou programu Protégé s obsahom dát z MITRE ATT&CK aj Hybrid Analysis.

S výsledkami sme spokojný, podarilo sa nám do ontológie preniesť všetky dáta ktoré sme chceli. Skripty ktoré sme napísali sú funkčné a nepadajú. Páči sa nám ako vieme jednoducho prezerat' naše dáta, vďaka tomu že sú ontológie výbornou ľuďmi čitateľnou reprezentáciou dát. Určite je to prehľadnejšie ako napríklad iba JSON súbory. Vďaka prehľadnosti na týchto dátach dokážeme jednoducho robiť analýzu. Dokážeme jednoducho analyzovať vzťahy medzi dátami.

Našou prácou sme dokázali že je možné zdieľať poznatky z bezpečnostnej domény pomocou ontológie. Dokázali sme možnosť vytvorenia znalostnej bázy o škodlivých softvéroch a následne priradenie výsledkov konkrétnych hybridných analýz k tejto báze.

Našu ontológiu je možné v prípade potreby v budúcnosti jednoducho rozšíriť, upraviť jej štruktúru aj pridať ďalšie dáta z iných zdrojov.

Záver

V práci sme splnili všetky stanovené ciele. Analyzovali sme problematiku ontológií a rdf grafov. Opísali sme použitie vývojového nástroja pre ontológie Protégé.

Analyzovali sme zdroje dát MITRE ATT&CK a Hybrid Analysis, z ktorých sme získavali dáta do našej ontológie. Nakoľko MITRE ATT&CK tvorí pre nás bázu znalostí o škodlivom softvéri, podrobnejšie sme analyzovali aj jednotlivé komponenty a vzťahy medzi nimi. Ďalej sme analyzovali spôsoby ako dokážeme získavať dáta zo zdrojov. Opísali sme niekoľko možných riešení z ktorých sme si pri implementácii vybrali. Tiež sme opísali akým spôsobom dokážeme vykonávať a vyhľadávať hybridné analýzy.

Analyzovali sme aj existujúce riešenie ontologického modelu z bezpečnostnej domény z ktorého ontológiu sme integrovali do nášho riešenia.

V kapitole opisu riešenia sme opísali ako sme navrhli štruktúru našej ontológie a tiež Python skripty, ktoré sme vytvorili pre vkladanie dát zo zdrojov do našej ontológie. Na záver sme naše riešenie otestovali a zhodnotili.

Do prílohy sme vytvorili používateľskú príručku, aby bolo jasné ako použiť naše skripty. Tiež sme do prílohy vytvorili tabuľky, ktoré jasne znázorňujú triedy a vzťahy našej ontológie.

Python skripty, ontológiu aj tento dokument je možné stiahnuť na webovej adrese https://github.com/j-pusztter/malware_ontology.

Zoznam použitej literatúry

1. **ontotext.** ontotext. *What are Ontologies?* [Online] [Dátum: 02. 05 2023.] <https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/>.
2. **W3C.** W3C. *RDF 1.1 Primer.* [Online] 24. 06 2014. [Dátum: 02. 05 2023.] <https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/>.
3. **W3C.** W3C. *RDF 1.1 Concepts and Abstract Syntax.* [Online] 25. 02 2014. [Dátum: 05. 02 2023.] <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
4. **W3C.** W3C. *RDF Schema 1.1.* [Online] 25. 02 2014. [Dátum: 02. 05 2023.] <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.
5. **W3C.** W3C. *An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL (March 2001).* [Online] 18. 12 2001. [Dátum: 02. 05 2023.] <https://www.w3.org/TR/2001/NOTE-daml+oil-axioms-20011218>.
6. **Stanford University.** Protégé 5 Documentation. *Getting started.* [Online] [Dátum: 03. 05 2023.] <http://protegeproject.github.io/protege/getting-started/>.
7. **Stanford University.** Protégé 5 Documentation. *Object Property Characteristics.* [Online] [Dátum: 03. 05 2023.] <http://protegeproject.github.io/protege/views/object-property-characteristics/>.
8. **Strom, Blake E., a iní.** MITRE ATT&CK. *Papers.* [Online] 03 2020. [Dátum: 04. 05 2023.] https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf.
9. **The MITRE Corporation.** MITRE ATT&CK. *Accessing ATT&CK Data.* [Online] [Dátum: 06. 05 2023.] <https://attack.mitre.org/resources/working-with-attack/>.
10. **The MITRE Corporation.** mitre-attack. *mitreattack-python.* [Online] [Dátum: 06. 05 2023.] <https://github.com/mitre-attack/mitreattack-python>.
11. **Miller, Jan.** Hybrid Analysis. *Hybrid Analysis - NextGen Technology for Advanced Malware Payload Detection.* [Online] 07 2014. [Dátum: 07. 05 2023.] <https://hybrid-analysis.blogspot.com/2014/07/>.
12. **Hybrid Analysis.** Hybrid Analysis. *Frequently Asked Questions (FAQ).* [Online] [Dátum: 07. 05 2023.] <https://www.hybrid-analysis.com/faq>.
13. **Hybrid Analysis.** Hybrid Analysis. *Hybrid Analysis.* [Online] [Dátum: 07. 05 2023.] <https://www.hybrid-analysis.com/>.
14. **Hybrid Analysis.** Hybrid Analysis. *Falcon Sandbox Public API.* [Online] [Dátum: 05. 07 2023.] <https://www.hybrid-analysis.com/docs/api/v2>.

15. **Hurtiš, Lukáš.** *Ontologický model pre bezpečnostnú doménu - Bakalárska práca.* FEI-5382-81464, 2019.

16. **Duarte, Felipe.** GitHub. *Hybrid Analysis API.* [Online] [Dátum: 10. 05 2023.] https://github.com/dark0pcodes/hybrid_analysis_api.

Prílohy

Príloha A:Používateľská príručka.....	II
Príloha B:Tabuľky tried a vzťahov ontológie	III

Príloha A: Používateľská príručka

V súbore `malwareTemplate.owl` sa nachádza prázdna ontológia. Túto ontológiu je možné naplniť MITRE ATT&CK dátami spustením skriptu `attck.py`. Výsledkom je nový súbor `malware.owl`.

Súbor `malware.owl` naplníme dátami hybridnej analýzy spustením skriptu `hybrid.py`. Pre správnu funkčnosť skriptu je nutné aby tento súbor obsahoval dáta z MITRE ATT&CK. Výsledkom je naplnený `malware.owl` súbor.

Aby bolo možné oba skripty spustiť, treba mať v rovnakom priečinku v ktorom spúšťame skript umiestnený vstupný súbor. Pre skript `attck.py` je to súbor `malwareTemplate.owl` a pre skript `hybrid.py` je to `malware.owl`.

Príloha B:Tabuľky tried a vzt'ahov ontológie

Technique

hasId	xsd:string
hasName	xsd:string
hasDescription	xsd:string
hasUrl	xsd:string
wasCreated	xsd:string
wasLastModified	xsd:string
hasVersion	xsd:string
hasDomain	xsd:string
hasContributors	xsd:string
hasDataSources	xsd:string
hasDefensesBypassed	xsd:string
hasDetection	xsd:string
hasEffectivePermissions	xsd:string
hasImpactType	xsd:string
hasMtcId	xsd:string
hasPermissionsRequired	xsd:string
hasPlatforms	xsd:string
hasRelationshipCitations	xsd:string
hasSystemRequirements	xsd:string
hasTacticType	xsd:string
supportsRemote	xsd:boolean
hasSubTechnique	Technique

Tactic

hasId	xsd:string
hasName	xsd:string
hasDescription	xsd:string
hasUrl	xsd:string
wasCreated	xsd:string

wasLastModified	xsd:string
hasVersion	xsd:string
hasDomain	xsd:string
usesTechnique	Technique

Software

hasId	xsd:string
hasName	xsd:string
hasDescription	xsd:string
hasUrl	xsd:string
wasCreated	xsd:string
wasLastModified	xsd:string
hasVersion	xsd:string
hasAliases	xsd:string
hasContributors	xsd:string
hasPlatforms	xsd:string
hasRelationshipCitations	xsd:string
hasType	xsd:string
usesTechnique	Technique

Group

hasId	xsd:string
hasName	xsd:string
hasDescription	xsd:string
hasUrl	xsd:string
wasCreated	xsd:string
wasLastModified	xsd:string
hasVersion	xsd:string
hasAssociatedGroups	xsd:string
hasAssociatedGroupsCitations	xsd:string
hasContributors	xsd:string
hasRelationshipCitations	xsd:string
usesSoftware	Software

usesTechnique	Technique
---------------	-----------

Mitigation

hasId	xsd:string
hasName	xsd:string
hasDescription	xsd:string
hasUrl	xsd:string
wasCreated	xsd:string
wasLastModified	xsd:string
hasVersion	xsd:string
hasDomain	xsd:string
hasRelationshipCitations	xsd:string
mitigates	Technique

file a url

hasSampleSummary	SampleSummary
------------------	---------------

SampleSummary

hasJobId	xsd:string
hasClassificationTag	[xsd:string]
hasTag	[xsd:string]
hasSubmission	[Submission]
hasMachineLearningModel	[MachineLearningModel]
hasCrowdStrikeAi	CrowdStrikeAi
hasEnvironmentId	xsd:integer
hasEnvironmentDescription	xsd:string
size	xsd:integer
hasType	xsd:string
hasTypeShort	[xsd:string]
hasTargetUrl	xsd:string
hasState	xsd:string
hasErrorType	xsd:string

hasErrorOrigin	xsd:string
hasSubmitName	xsd:string
md5	xsd:string
sha1	xsd:string
sha256	xsd:string
sha512	xsd:string
hasSsdeep	xsd:string
hasImphash	xsd:string
hasEntrypoint	xsd:string
hasEntrypointSection	xsd:string
hasImageBase	xsd:string
hasSubsystem	xsd:string
hasImageFileCharacteristic	[xsd:string]
hasDllCharacteristic	[xsd:string]
hasMajorOsVersion	xsd:integer
hasMinorOsVersion	xsd:integer
hasAvDetect	xsd:integer
hasVxFamily	xsd:string
hasAnalysisStartTime	xsd:string
hasThreatScore	xsd:integer
hasThreatLevel	xsd:integer
hasVerdict	xsd:string
hasCertificate	[Certificate]
hasSampleDomain	[xsd:string]
hasCompromisedHost	[xsd:string]
hasHost	[xsd:string]
hasTotalNetworkConnections	xsd:integer
hasTotalProcesses	xsd:integer
hasTotalSignatures	xsd:integer
hasExtractedFile	[ExtractedFile]
hasFileMetadata	FileMetadata

hasProcess	[Process]
hasMitreAttckSignature	[MitreAttckSignature]
hasNetworkMode	xsd:string
hasSignature	[Signature]

Submission

hasSubmissionId	xsd:string
hasFileName	xsd:string
hasUrl	xsd:string
wasCreated	xsd:string

MachineLearningModel

hasName	xsd:string
hasVersion	xsd:string
hasStatus	xsd:string
hasKeyStringValue	KeyStringValue
wasCreated	xsd:string

KeyStringValue

key	xsd:string
value	xsd:string

CrowdStrikeAi

hasExecutableProcessMemoryAnalysis	ExecutableProcessMemoryAnalysis
hasAnalysisRelatedUrl	AnalysisRelatedUrl

ExecutableProcessMemoryAnalysis

hasVerdict	xsd:string
hasFilename	xsd:string
hasAddress	xsd:string
hasFlags	xsd:string
hasFileProcess	xsd:string
hasFileProcessPid	xsd:integer
hasFileProcessSha256	xsd:string
hasFileProcessDiscPathway	xsd:string

AnalysisRelatedUrl

hasUrl	xsd:string
hasVerdict	xsd:string
hasType	xsd:string

Certificate

hasOwner	xsd:string
hasIssuer	xsd:string
hasSerialNumber	xsd:string
md5	xsd:string
sha1	xsd:string
isValidFrom	xsd:string
isValidUntil	xsd:string

ExtractedFile

hasName	xsd:string
hasFilePath	xsd:string
size	xsd:integer
sha1	xsd:string
sha256	xsd:string
md5	xsd:string
hasTypeTag	[xsd:string]
hasDescription	xsd:string
hasRuntimeProcess	xsd:string
hasThreatLevel	xsd:integer
hasThreatLevelReadable	xsd:string
hasAvLabel	xsd:string
hasAvMatched	xsd:integer
hasAvTotal	xsd:integer

FileMetadata

hasFileComposition	[xsd:string]
hasImportedObject	[xsd:string]

hasFileAnalysis	[xsd:string]
hasTotalFileCompositionsImports	xsd:integer

Process

hasUid	xsd:string
hasParentuid	xsd:string
hasName	xsd:string
hasNormalizedPath	xsd:string
hasCommandLine	xsd:string
sha256	xsd:string
hasAvLabel	xsd:string
hasAvMatched	xsd:integer
hasAvTotal	xsd:integer
hasPid	xsd:string
hasIcon	xsd:string
hasFileAccess	FileAccess
hasCreatedFile	CreatedFile
hasRegistryAccess	RegistryAccess
hasMutant	[xsd:string]
hasHandle	Handle
hasStream	Stream
hasScriptCall	ScriptCall
hasProcessFlag	ProcessFlag
hasAmsiCall	AmsiCall

FileAccess

hasType	xsd:string
hasPath	xsd:string
hasMask	xsd:string

CreatedFile

hasFile	xsd:string
hasNullByte	xsd:boolean

Registry

hasOperation	xsd:string
hasPath	xsd:string
key	xsd:string
value	xsd:string
hasStatus	xsd:string
hasStatusHumanReadable	xsd:string

Handle

hasHandleId	xsd:integer
hasType	xsd:string
hasPath	xsd:string

Stream

hasUid	xsd:string
hasFileName	xsd:string
hasHumanKeywords	xsd:string
hasInstructions	[xsd:string]
wasExecuted	xsd:boolean
hasKeyStringValue	[KeyStringValue]

ScriptCall

hasClsId	xsd:string
hasDispatchId	xsd:string
hasStatus	xsd:string
hasResult	xsd:string
hasScriptCallParameter	[ScriptCallParameter]
hasMatchedMaliciousSignature	[xsd:string]

ScriptCallParameter

hasName	xsd:string
value	xsd:string
hasComment	xsd:string
hasArgumentNumber	xsd:integer

hasMeaning	xsd:string
------------	------------

ProcessFlag

hasName	xsd:string
hasData	xsd:string
hasImage	xsd:string

AmsiCall

hasAppName	xsd:string
hasFileName	xsd:string
hasRawScriptContent	xsd:string

MitreAttckSignature

hasMaliciousIdentifiersCount	xsd:integer
hasMaliciousIdentifier	[xsd:string]
hasSuspiciousIdentifiersCount	xsd:integer
hasSuspiciousIdentifier	[xsd:string]
hasInformativeIdentifiersCount	xsd:integer
hasInformativeIdentifier	[xsd:string]
hasAttckTechnique	Technique

Signature

hasThreatLevel	xsd:integer
hasThreatLevelHuman	xsd:string
hasCategory	xsd:string
hasIdentifier	xsd:string
hasSignatureType	xsd:integer
hasRelevance	xsd:integer
hasName	xsd:string
hasDescription	xsd:string
hasOrigin	xsd:string
hasCapeId	xsd:string
hasAttckTechnique	Technique