# Software pro podporu rozhodování při řešení bezpečnostního incidentu

Lukáš Sadlek, Michal Javorník, Martin Husák

2020

## Abstrakt

Tento dokument popisuje stejnojmenný výstup projektu "*Výzkum nástrojů pro hodnocení kybernetické situace a podporu rozhodování CSIRT týmů při ochraně kritické infrastruktury*" (VI20172020070) řešeného v programu *Bezpečnostní výzkum České republiky* v letech 2017-2020 na Masarykově univerzitě. Dokument obsahuje popis výsledku, návod k instalaci, uživatelskou dokumentaci a programátorskou dokumentaci.

# Obsah

# Úvod

Tento dokument obsahuje dokumentaci k výsledku *"Software pro podporu rozhodování při řešení bezpečnostního incidentu"* (dále Decide), který je výstupem projektu "*Výzkum nástrojů pro hodnocení kybernetické situace a podporu rozhodování CSIRT týmů při ochraně kritické infrastruktury*" (VI20172020070, dále označován jako projekt CRUSOE) řešeného v programu *Bezpečnostní výzkum České republiky* v letech 2017-2020 na Masarykově univerzitě.

Cílem výše zmíněného projektu bylo vytvořit nástroje, které pomáhají specialistům v kyberbezpečnostním týmu zmapovat a zorientovat se v aktuální kyberbezpečnostní situaci v počítačové síti, rychle a dobře rozhodnout o postupu řešení probíhajících incidentů a navržené řešení implementovat. Sada vytvořených nástrojů odráží myšlenkový koncept takzvaného OODA cyklu, který sestává ze čtyř fází - Observe, Orient, Decide a Act. OODA cyklus byl navržen a popsán jako obecný postup pro sběr informací a podporu rozhodování. Uživatel OODA cyklu iteruje jednotlivými fázemi, díky čemuž formalizuje své myšlení a je tak schopen efektivně a rychle rozhodovat a konat bez zbytečných chyb. Nejprve je třeba nasbírat informace o prostředí (fáze Observe), následně se v nich zorientovat (fáze Orient), dále rozhodnout o vhodném dalším postupu (fáze Decide) a nakonec tento postup realizovat (fáze Act). V rámci projektu CRUSOE byl pro každou fázi OODA cyklu vytvořen nástroj, který umožňuje uživatelům cíle dané fáze implementovat. *"Software pro podporu rozhodování při řešení bezpečnostního incidentu"* implementuje fázi Decide, tedy fázi podpory rozhodování.

Software Decide slouží k výběru nejodolnější konfigurace informačního systému nebo počítačové sítě s ohledem na bezpečnostní požadavky a požadavky na zajištění kritických procesů organizace, která informační systém nebo počítačovou síť provozuje. Software zohledňuje aktuální situace v počítačové síti včetně zranitelností systémů v síti a pozice útočníka. Software Decide pravidelně kontroluje informace o počítačové síti nebo informačním systému, který je předmětem ochrany kyberbezpečnostního týmu. Ve vstupních datech očekává seznam zařízení včetně provozovaných služeb a software, přehled zranitelností a pozice útočníka (seznam zařízení a úrovní oprávnění, které mohou být v rukou útočníka) a formální popis požadavků na procesy dané organizace. Formální popis požadavků obsahuje seznam kritických procesů uvnitř organizace, seznam prvků kritické infrastruktury dané organizace a jejich vzájemné mapování. Software tyto vstupy zpracuje a porovná možné konfigurace prvků kritické infrastruktury, které umožňují, aby všechny kritické procesy uvnitř organizace byly plně funkční. Jednotlivým konfiguracích je dopočítáno skóre, které určuje pravděpodobnost úspěšného útoku, který by ohrozil procesy v organizaci. Konfigurace s nejnižší pravděpodobností napadení je vybrána jako doporučovaná. Výstupem software Decide je tedy doporučení na (re)konfiguraci počítačové sítě nebo informačního systému tak, aby pravděpodobnost úspěšného útoku byla co nejnižší. Rekonfigurace dané sítě či systému není součástí implementace a vždy závisí na finálním rozhodnutí uživatele, zda se doporučením řídit. Výstupy software Decide jsou však využitelné nástrojem implementujícím fázi Act.

Tento dokument sestává z pěti částí. Po úvodu obsahujícím i shrnutí technických a ekonomických parametrů výsledku následuje popis výsledku vysvětlující principy, na kterých je software postaven. V dalších částech jsou uvedeny návod k instalaci, uživatelská dokumentace a programátorská dokumentace.

## Technické parametry výsledku

Software implementuje sadu algoritmů pro podporu rozhodování kyberbezpečnostního týmu při ochraně počítačové sítě nebo informačního systému. Software zpracovává seznam zařízení, služeb a zranitelností a zařízení a oprávnění, která mohou být v rukou útočníka. Dalším vstupem je formální reprezentace služeb, které zajišťují kritické procesy pro organizaci provozující danou počítačovou síť nebo informační systém, a jejich mapování na zařízení a služby v síti. Na základě daných informací software vytváří seznam možných konfigurací, které umožňují fungování kritických procesů. Ke každé konfiguraci je dopočítáno skóre, které určuje pravděpodobnost úspěšného útoku. Výstupem software je doporučení nejodolnější konfigurace.

Software je distribuován jako open-source pod licencí MIT, vlastníkem výsledku je Masarykova univerzita, IČO 00216224.

Kontaktní osoba: RNDr. Martin Husák, Ph.D.
Ústav výpočetní techniky
Masarykova Univerzita
Šumavská 416/15
Brno 602 00
e-mail: husakm@ics.muni.cz
tel: +420 549 49 4857

## Ekonomické parametry výsledku

Tržní segment představují zejména organizace provozující kritickou informační infrastrukturu nebo jinou infrastrukturu s vysokými požadavky na důvěrnost, dostupnost a integritu. Výsledek umožňuje uživatelům (provozovatelům sítí a služeb) vybrat nejodolnější konfiguraci výpočetní infrastruktury v daném čase za daných podmínek, což umožňuje eliminovat kyberbezpečnostní hrozby a předcházet útokům s nejzávažnějšími dopady na infrastrukturu. Využití výsledku tak má především preventivní dopady před škodami způsobenými kyberútoky. Efektivita vytvořeného software závisí na kvalitě vstupních dat poskytnutých uživatelem a zda a jak uživatel nastaví infrastrukturu podle doporučení, které je výstupem software. Využití výsledku je licencováno bez poplatku.

# Popis výsledku

S nárůstem složitosti a rozsahu IT infrastruktury je stále komplikovanější ji efektivně chránit jako celek. Obranné činnosti je potřeba zacílit na kritické aktivity, které se na infrastruktuře odehrávají. Jinými slovy, je potřeba detailně zmapovat jednotlivé mise, jejich podpůrné procesy, priority a zejména bezpečnostní požadavky kladené na jednotlivé procesy z hlediska důvěrnosti, integrity a dostupnosti. Klíčovou otázkou je, jak posoudit bezpečnostní situaci mise, resp. jak kvantifikovat (odvodit, spočítat) odolnost mise ve smyslu pravděpodobnosti narušení bezpečnostních požadavků některého z procesů tvořících misi.

Rozhodovací proces je založen na následujících předpokladech a charakteristikách:

- Princip rozhodovacího procesu předpokládá, že ke splnění cílů sledované mise vede více různých variant konfigurace podpůrné infrastruktury.
- Cílem rozhodovacího procesu je nalezení takové konfigurace podpůrné infrastruktury (takových opatření proti útoku), která udrží kritické procesy sledované mise co nejdéle ve stavu, kdy nedošlo k narušení bezpečnostních požadavků.
- Proces pro podporu rozhodování integruje interní specifické informace o přípustných konfiguracích mise a externí informace o zranitelnostech, vektorech útoků a bezpečnostních metrikách z veřejně dostupných zdrojů.
- Primární uplatnění je v situaci, kdy infrastruktura čelí útoku. Výpočetní algoritmy použité v rámci rozhodovacího procesu lze využít i proaktivně, tj. například při simulaci potenciálních variant útoku s cílem odhalení zranitelností komponent infrastruktury majících kritický dopad na sledovanou misi.
- Klíčovou roli rozhodovacího procesu hraje model dekompozice mise. Hierarchický model umožňuje srozumitelně a v souvislostech vyjádřit důsledky jednotlivých variant rozhodnutí, a tím zvýšit transparentnost rozhodovacího procesu.
- Princip procesu vychází z rozdělení rolí mezi tým zajišťující kybernetickou bezpečnost a management instituce, který na základě znalosti specifické problémové oblasti (domény) rozhoduje na úrovni řízení procesů tvořících sledovanou misi.
- Rozhodovací proces umožňuje kvantifikovat bezpečnostní situaci, tj. umožňuje kvantitativní porovnání bezpečnosti dostupných konfigurací mise a zejména identifikaci nejbezpečnější varianty.
- Algoritmický princip metody, který je v jednotlivých kapitolách textu podrobně popsán je následující. Navržená metoda je svým principem optimalizační úloha, jejímž cílem je nalezení optimálního řešení optimalizační (penalizační) funkce, která je definovaná na množině všech přípustných řešení daného problému. Množina přípustných řešení je formálně určena dekompozičním modelem dané mise. Model obsahuje všechny relevantní podpůrné entity mise, jejich vzájemné vazby a bezpečnostní požadavky. Vstupem metody je model dekompozice mise, spolu s abstraktními znalostmi získanými z veřejně dostupných zdrojů. Tyto vstupní informace a dostupné algoritmy umožní zkonstruovat Bayesovskou síť založenou na grafu útoku. Využitím inferenčních mechanismů nad Bayesovskou sítí dokážeme kvantifikovat jednotlivé varianty rozhodnutí a řešit danou optimalizační úlohu.

## Použité metody

Rozhodovací proces je založen na třech hlavních teoretických metodách, grafu útoku, Bayesovské síti a optimalizačním problému. Následuje jejich stručné představení.

### Graf útoku

Na graf útoku lze formálně pohlížet jako na množinu stavů systému a množinu jejich přechodových vztahů. Počáteční stav systému představuje stav před zahájením útoku. Přechodové vztahy představují možné kroky útočníka. Přechodové vztahy lze doplnit informacemi o pravděpodobnosti, že útočník zvolí konkrétní krok (mohou například reprezentovat míru úsilí potřebného k realizaci daného kroku). Pokud útočník úspěšně podnikne všechny kroky vedoucí k přechodu z počátečního stavu systému do některého z cílových stavů, považujeme útok za úspěšný. Cílový stav znamená kompromitaci systému ve smyslu narušení jeho důvěrnosti, integrity nebo dostupnosti.

Pro generování grafu útoku lze využít spektrum dostupných algoritmů. V rámci řešení projektu jsme zvolili nástroj MulVAL, nástroj pro analýzu zranitelností v síti. Skládá se ze skenerů, které jsou instalovány na každém počítači v síti, a analyzátoru, který následně zpracovává výsledky a informace o zranitelnosti. Mezi hlavní výhody tohoto nástroje patří to, že dokáže automaticky zpracovat specifikace zranitelností z veřejně dostupných zdrojů, a pak zejména polynomiální složitost algoritmu pro sestavení grafu útoku.

Informace o důsledcích zranitelností byly převzaty z databáze NVD a hodnot CVSS skóre dané zranitelnosti. Dopad každé zranitelnosti je možné rozdělit do následujících čtyř kategorií: ztráta důvěrnosti, ztráta dostupnosti, ztráta integrity a eskalace oprávnění. Tyto zranitelnosti jsou detekovány pomocí skenerů zranitelností, například nástrojů popsaných v rámci výsledku “Software pro evidenci zranitelností v počítačové síti” projektu CRUSOE.

Grafy útoků umožňují komplexní popis možných scénářů útoků, tj. podávají kvantitativní informaci o bezpečnostním stavu systému. Grafy útoku neumožňují formální vyjádření nejistoty spojené s chováním útočníka tedy možnost kvantifikovat bezpečnostní stav systému. Doplněním údajů o lokálních pravděpodobnostech, resp. lokálních podmíněných pravděpodobnostech potenciálních kroků útočníka lze graf útoku následně transformovat do struktury Bayesovské sítě. Chybějící kvantitativní informace o bezpečnosti systému lze následně odvodit z této struktury.

### Bayesovská síť

Bayesovská síť je struktura spadající do skupiny pravděpodobnostních grafových modelů. Bayesovskou síť lze popsat jako orientovaný acyklický graf, kde jednotlivé uzly grafu reprezentují náhodné veličiny a orientované hrany reprezentují kauzální vztahy mezi těmito náhodnými veličinami. Každé proměnné, do které vede alespoň jedna orientovaná hrana je přiřazena tabulka, která ve formě lokálních podmíněných pravděpodobností kvantifikuje vztah mezi touto proměnnou a náhodnými veličinami ze kterých vychází orientované hrany. Pro kvantifikaci lokálního bezpečnostního rizika lze využít relevantních metrik definovaných v rámci CVSS a transformovaných do podoby podmíněných pravděpodobností.

Graf útoku lze takto rozšířit o údaje o pravděpodobnostech kvantifikujících vztahy mezi jednotlivými uzly grafu útoku, tj. vztahy mezi souvisejícími pozicemi útočníka. Navržený proces dynamicky kalkuluje aktuální riziko a porovnává přípustné alternativy možných opatření vedoucích k zvýšení odolnosti systému proti útoku.

### Optimalizační problém

Hledáme řešení optimalizační úlohy, v tomto případě stochastického problému, tj. řešení (konfiguraci mise) s největší pravděpodobností zachování bezpečnostních požadavků mise v rámci explicitně formulovaných konfiguračních alternativ mise (zachování funkčních požadavků). Struktura Bayesovské sítě nám umožňuje kvantifikovat (marginální) pravděpodobnost úspěšného narušení bezpečnosti (požadavků kladených na danou konfiguraci procesů mise) zneužitím zranitelností softwarových komponent tvořících misi. Na množině všech přípustných řešení problému máme tedy definovanou reálnou optimalizační funkci. Výpočet dokáže seřadit dostupné konfigurace na základě kritéria odolnosti proti útoku s ohledem na aktuální pozici útočníka. Výpočet zejména určí nejvýhodnější (nejodolnější konfiguraci) protiopatření. Konečné rozhodnutí o volbě protiopatření a rekonfiguraci systému provede management instituce.

## Modelování misí

Model misí v rámci organizace je součástí vstupů software pro podporu rozhodování. Mise je třeba identifikovat, dekomponovat a formálně popsat, aby bylo možné ji zohlednit při podpoře rozhodování.

### Model dekompozice mise

Předpokládáme, že mise je tvořena jedním nebo více procesy. S těmito procesy pracujeme jako s klíčovými aktivy. Jednotlivé podpůrné procesy lze následně mapovat na potřebné podpůrné IT služby a tyto lze následně mapovat na podpůrné softwarové komponenty a jejich specifické interakce.

Tento koncept předpokládá nastavení požadavků na důvěrnost, integritu a dostupnost na úrovni jednotlivých procesů a následně pak mapování těchto požadavků na relevantní softwarové komponenty a jejich interakce.

Předpokládáme, že cílů mise (souladu s funkčními požadavky) lze dosáhnout prostřednictvím více alternativ konfigurace podpůrných procesů, IT služeb a softwarových komponent. Přípustné konfigurace mise jsou definovány pomocí orientovaného grafu. Pravidla jsou vyjádřena speciálními AND/OR uzly a příslušnými hranami v grafu.

Na misi pohlížíme ze čtyř různých úhlů pohledu. Rozlišujeme funkční požadavky mise, požadavky na zabezpečení mise, přípustnou konfiguraci mise a odolnost mise.

**Funkční požadavky mise.** Misi lze formálně popsat jako systém přípustných kombinací funkčních požadavků kladených na její podpůrné procesy, IT služby, softwarové komponenty a jejich interakce.

**Bezpečnostní požadavky mise.** Bezpečnostní požadavky mise lze formálně popsat jako systém bezpečnostních požadavků (vyjádřený jako požadovaná úroveň důvěrnosti, integrity a dostupnosti), kladených na jednotlivé procesy mise.

**Přípustná konfigurace mise.** Přípustná konfigurace mise musí být vždy v souladu s definovanými funkčními požadavky mise. Formálně je vyjádřena jako specifická konfigurace všech podpůrných entit a jejich interakcí. Alternativy konfigurace mise jsou v grafové notaci vyjádřeny pomocí speciálních uzlů AND/OR, tj. reprezentují AND/OR logiku. Uzel AND v konkrétní konfiguraci mise vyžaduje přítomnost všech zmíněných uzlů představujících podpůrné IT služby resp. počítačové komponenty. OR uzly v konfiguraci mise vyžadují přítomnost alespoň jednoho ze zmíněných uzlů představujících podpůrné IT služby respektive počítačové komponenty.

**Odolnost mise.** Odolnost mise je její schopnost pokračovat v činnosti, resp. schopnost plnit požadované cíle mise (definované funkční požadavky) při dané konfiguraci a při zohlednění aktuální pozice útočníka. Odolnost mise kvantifikujeme ve smyslu pravděpodobnosti narušení bezpečnostních požadavků některého z procesů tvořících misi.

## Příklad dekompozice mise

Následující případová studie modelového lékařského pracoviště z oblasti zpracování medicínských obrazových informací ilustruje navrhovanou metodu. Nejprve stručně popíšeme problémovou doménu, následně pak model dekompozice mise.

Zpracování medicínských obrazových informací na lékařském pracovišti představuje široké spektrum vzájemně propojených činností nad spektrem podpůrných IT procesů mnoha poskytovatelů zdravotnických i IT služeb. IT aplikační prostředí se skládá z aplikací specifických pro danou doménu, provozovaných ve speciálně nakonfigurovaných počítačových sítích.

Model dekompozice mise je tvořen předdefinovanými pravidly popisujícími všechny přípustné konstelace dílčích komponent, které vyhovují funkčním požadavkům kladeným na misi. Bezpečnostní požadavky kladené na jednotlivé podpůrné procesy reflektují právní, etické, smluvní či jiné specifické požadavky. Souhrnně je lze vyjádřit jako požadované úrovně důvěrnosti, integrity a dostupnosti.

Klíčovými procesy mise jsou zejména procesy umožňující vyšetření pacienta na akviziční modalitě (CT, MRI, rentgen, mamografický screening atd.), základní diagnostika, odborné konzultace, případně další specifické zpracování obrazové informace.

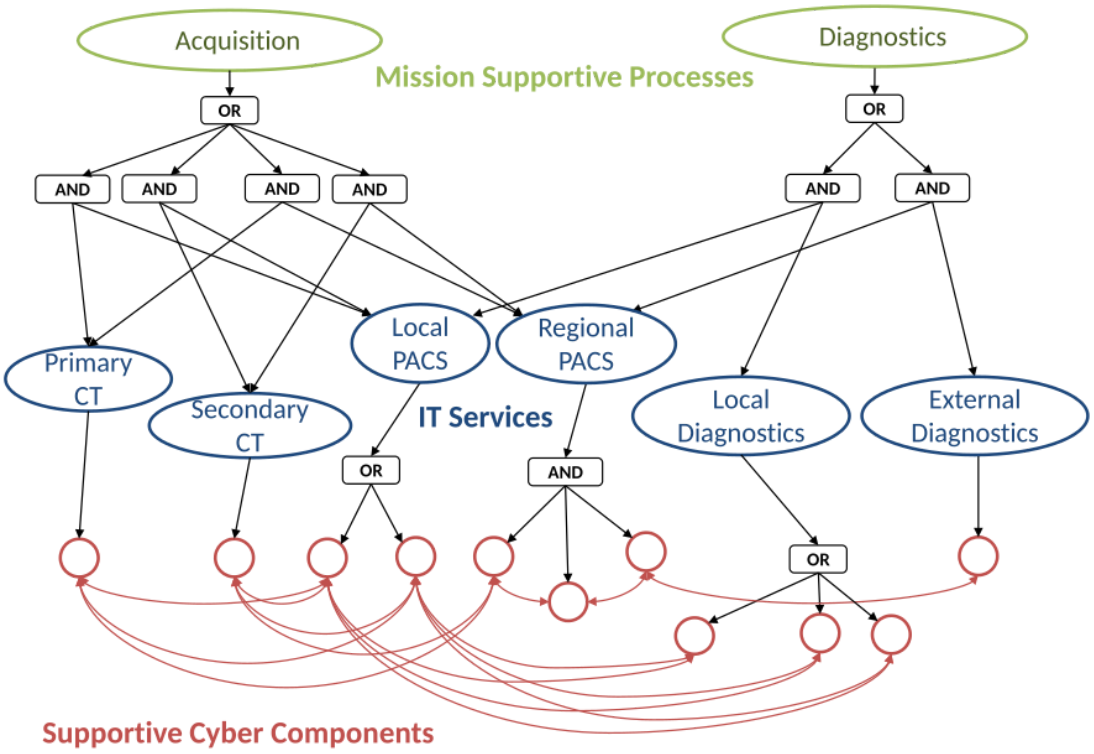
Mezi klíčové IT služby patří zejména podpora akvizice obrazových dat, provoz institucionálního nebo regionálního archivačního a komunikačního systému (PACS), služby pro výměnu a sdílení obrazových dat mezi jednotlivými nemocničními odděleními a dalšími vzdálenými zdravotnickými institucemi a podpora diagnostiky obrazových vyšetření.

Subsystém podpůrných softwarových komponent je tvořen širokým spektrem produktů: konkrétní implementace PACS, softwarové komponenty pro řízení akvizice, software pro speciální diagnostiku apod. Jednotlivé komponenty komunikují prostřednictvím doménově specifických komunikačních protokolů.

Předpokládejme, že modelové pracoviště je vybaveno dvěma přístroji počítačové tomografie (CT). Diagnostika obrazových dat se převážně provádí lokálně, komplikovanější situace jsou navíc konzultovány s odborníky ze vzdálených specializovaných institucí.

Pracoviště zabezpečuje dva kritické procesy. Prvním je proces akvizice (Acquisition), tj. realizace vlastního vyšetření na CT a následné zpřístupnění obrazových dat prostřednictvím archivačního, resp. komunikačního systému PACS pro další zpracování. Druhým procesem je diagnostický proces (Diagnostics), tj. získání obrazového vyšetření z archivačního, resp. komunikačního systému PACS a vlastní realizace požadované diagnostiky.

Procesy využívají následující podpůrné IT služby: akvizice medicínských obrazových dat (PrimaryCT, resp. SecondaryCT), zpřístupnění obrazových dat v rámci pracoviště nebo mezi vzdálenými zdravotnickými institucemi regionu (LocalPACS, resp. RegionalPACS) a služby lokální a externí diagnostiky CT vyšetření (Local-Diagnostics, resp. External-Diagnostics).




Obrázek 1: Příklad modelu mise organizace a jejich závislostí.

## Analytický proces

Analytický proces pro podporu rozhodování vychází z modelu dekompozice mise a využívá výše popsaných matematických metod. Výchozím krokem procesu je stanovení souboru omezujících podmínek definujících přípustná řešení rozhodovacího problému a definice užitkové (penalizační) reálné funkce, která jednotlivá přípustná řešení kvantifikuje. Cílem rozhodovacího procesu je nejen nalezení optimálního řešení takto definovaného problému,

tj. nalezení nejodolnější konfigurace v rámci množiny přípustných konfigurací mise, ale i poskytnutí srozumitelného zdůvodnění všem účastníkům rozhodovacího procesu, tj. zejména osobám zodpovědným za síťovou bezpečnost a osobám zodpovědným za rozhodování na úrovni managementu dané aplikační domény.

Proces podpory rozhodování se skládá ze tří fází:

1. Cílem první fáze je konstrukce statického modelu dekompozice mise. Proces dekompozice je popsán v předchozí kapitole. Tento krok se realizuje pouze jednou. Výstupem této fáze jsou model dekompozice mise a formální zápis omezujících podmínek odvozený z modelu, např. v podobě logického výrazu, který lze následně algoritmicky zpracovat. Formální zápis vyhovujících konfigurací mise použitím výrazových prostředků matematické logiky je základem pro další fáze rozhodovacího procesu.
2. Cílem druhé fáze procesu je kvantifikace bezpečnostní situace. Využívá model dekompozice mise společně s matematicky vyjádřenými omezujícími podmínkami, které definují přípustné konfigurace mise. Tato fáze je postavena na konceptu grafů útoků a Bayesovských sítí, a dále pak na obecných znalostech o kybernetických hrozbách a zejména informacích o aktuální pozici útočníka. Výsledkem je Bayesovský graf útoků, který reprezentuje dynamiku situace, tj. umožňuje odvodit (kvantifikovat) pravděpodobnosti variant jejího dalšího možného vývoje.
3. Třetí fáze integruje statický výstup první fáze a dynamický výstup druhé fáze. Na základě výpočtu identifikuje výslednou optimální konfiguraci mise, a tomu odpovídající protiopatření optimálně reflektující aktuální kybernetickou hrozbu, tj. rekonfiguraci aktivních síťových prvků, případně změny přímo v konfiguracích jednotlivých aplikačních komponent.

Výstupem celého procesu je doporučení nejodolnější konfigurace systému, která dokáže za popsané bezpečnostní situace maximálně snížit efektivitu útoku. Doporučenou konfiguraci je potřeba nastavit ručně, případně automaticky. Pokud následně dojde ke změně pozice útočníka, případně dalších informací vstupujících do algoritmu výpočtu, je potřeba druhou a třetí fázi procesu cyklicky opakovat.

Ke zkonstruování Bayesovského grafu útoku jsou kromě popisu aktivních softwarových komponent mise a jejich povolených interakcí potřeba také aktuální informace o relevantních vektorech útoků a informace o aktuální pozici útočníka, tj. informace, u kterých předpokládáme dynamickou změnu v čase. Zde je nezbytná spolupráce se systémy detekce útoků včetně znalosti posloupnosti aktivit, které útočník již úspěšně provedl.

Obecné informace o zranitelnostech lze získat z různých zdrojů spravovaných výrobci softwaru či zdrojů spravovaných bezpečnostními specialisty. V navržených algoritmech využíváme informace z veřejně dostupných zdrojů. NVD (National Vulnerability Database) reprezentuje de facto standardní knihovnu zranitelností popsaných ve formátu CVE. Zranitelnosti v CVE formátu jsou doplněny o CVSS (Common Vulnerability Scoring Systém) bodovací systém, který ve formě specializovaných metrik charakterizuje například složitost útoku či jeho dopad z hlediska narušení důvěrnosti, integrity a dostupnosti. Databáze CVSS jako veřejně přístupný důvěryhodný zdroj informací nám poskytuje potřebné lokální kvantitativní charakteristiky nezbytné pro zkonstruování Bayesovského grafu útoku.

Bezpečnostní metrika CVSS Attack Complexity, kterou lze po jednoduché transformaci interpretovat jako podmíněnou pravděpodobnost úspěšného zneužití dané zranitelnosti, dodává výchozí informace pro finální výpočet marginální pravděpodobnosti úspěšného vícestupňového útoku cestou zneužití posloupnosti zranitelností kritických pro sledovanou misi.

Bezpečnostní metrika odkazující na potenciální dopad úspěšného zneužití dané zranitelnosti (s ohledem na důvěrnost, integritu a dostupnost dané kybernetické komponenty) specifikuje oprávnění, která útočník získá v důsledku úspěšného zneužití zranitelnosti.

Posloupnost konkrétních kroků procesu kvantifikace bezpečnostní situace je následující:

- identifikace zranitelností a slabin systému využitím databáze NVD a modelu dekompozice mise, který poskytuje seznam všech použitých IT komponent a jejich interakcí,
- získání aktuální pozice útočníka využitím systému detekce narušení nebo využitím podobného nástroje, který hlásí zneužití zranitelnosti nebo podobné události,
- vygenerování grafu útoku použitím některého z dostupných algoritmů, například MulVal,
- transformace grafu útoku na Bayesovský graf útoku doplněním parametrů v podobě lokálních pravděpodobností vhodnou interpretací bezpečnostních metrik CVSS,
- výpočet marginální pravděpodobnosti úspěšného vícestupňového útoku směřujícího k získání oprávnění útočníka, která znamenají narušení důvěrnosti, integrity či dostupnosti systému.

## Podpora rozhodování

Výstupem analytického procesu je seznam vypočtených odolností každé konfigurace zařízení v síti, které plně podporují danou misi. Podpora rozhodování spočívá ve výběru konfigurace IT infrastruktury, která je nejodolnější. Ve fázi podpory rozhodování je nejprve vybrána nejodolnější konfigurace a následně je možné provést finální rozhodnutí.

### Výběr nejodolnější konfigurace

Výběr nejodolnější konfigurace je přímočarý, nicméně v praxi může být výhodné podrobit výběr dalším podmínkám, aby se předešlo doporučením, která by vedla ke zbytečným či příliš častým změnám konfigurace IT infrastruktury. Pro každou konfiguraci, která plně podporuje danou misi, je vypočtena pravděpodobnost ohrožení dané mise. Výpočet je založen na dopadu zranitelností, úsilí, které musí útočník vložit, a závažnosti dopadů na misi. Konfigurace s nejnižší pravděpodobností ohrožení mise je vybrána jako nejodolnější. Pokud není zavedena další kontrola, tento výsledek je přijat.

V praxi může nastat situace, kdy by ke změně konfigurace na základě doporučení docházelo příliš často nebo by odůvodnění doporučení bylo příliš slabé, aby takovou změnu opodstatnilo. Přídavná kontrola může spočívat ve srovnání pravděpodobností narušení mise mezi aktuální a doporučovanou konfigurací. Pokud je rozdíl nepatrný, například méně než jedno procento, není nutné měnit konfiguraci IT infrastruktury. Dopad na bezpečnost mise by byl rovněž nepatrný. Změna konfigurace navíc může být netriviální úkol, který zabere mnoho

času a omezit komfort uživatelů IT infrastruktury. Proto je vhodné přistoupit ke změně konfigurace teprve ve chvíli, kdy by doporučovaná konfigurace výrazněji zvýšila odolnost infrastruktury.

### Finální rozhodnutí

Výběr nejodolnější konfigurace ukončuje proces podpory rozhodování. Jak bylo popsáno výše, doporučení může a nemusí být přijato. Finální rozhodnutí je zodpovědností příslušných správců, manažerů, bezpečnostních expertů či jiných zainteresovaných osob. Vykonání daného rozhodnutí, tedy změna konfigurace IT infrastruktury je pak plně v rukou operátorů infrastruktury.

Při přijetí finálního rozhodnutí je třeba třeba pamatovat na to, že hodnoty odolnosti daných konfigurací jsou pouze relativní a je třeba je brát v kontextu dalších faktorů. Z nejdůležitějších je třeba jmenovat:

- dopady na ekonomickou efektivitu organizace,
- dopady na redundanci kritických komponent,
- dopady na pracovní návyky uživatelů IT infrastruktury.

Konkrétní dopady lze jen těžko popsat bez uvedení konkrétních případů, případně by to vybočovalo z problematiky podpory rozhodování kyberbezpečnostního týmu. Stejně tak automatizace změn konfigurace infrastruktury je mimo toto téma. Účast člověka v procesu rozhodování je v tomto případě nadále vyžadována, minimálně formou dohledu.

# Návod k instalaci

Návod k instalaci software pro podporu rozhodování při řešení bezpečnostního incidentu sestává ze tří částí. Nejprve je popsána instalace prerekvizit, následně je popsána instalace samotného software a na závěr je popsán postup konfigurace před prvním spuštěním.

Návod k instalaci byl ověřen na operačních systémech Ubuntu 18.04 LTS a Debian 10.

## Prerekvizity

Software pro podporu rozhodování při řešení bezpečnostního incidentu vyžaduje, aby byly v systému nainstalovány softwarové nástroje MulVAL a XSB. Nejprve je třeba vybrat, kam budou tyto nástroje nainstalovány. Doporučujeme je umístit do adresáře */opt*.

Poznámka: některé z následujících příkazů mohou vyžadovat oprávnění uživatele *root* nebo spuštění pomocí *sudo*.

Nejprve je třeba stáhnout a rozbalit instalační balík nástroje MulVAL pomocí příkazů:

```
$ wget -P /opt
https://people.cs.ksu.edu/~xou/argus/software/mulval/mulval_1_1.tar.gz
$ cd /opt
$ tar -xzf mulval_1_1.tar.gz
$ rm mulval_1_1.tar.gz
```

Prerekvizitou nástroje MulVAL je nástroj XSB, jehož instalační balík je třeba stáhnout a rozbalit pomocí příkazů:

```
$ wget -P /opt http://xsb.sourceforge.net/downloads/XSB.tar.gz
$ tar -xzf XSB.tar.gz
$ rm XSB.tar.gz
```

Před instalací je třeba nastavit proměnné prostředí udávající umístění nástrojů MulVAL a XSB:

```
$ export MULVALROOT=/opt/mulval
$ export XSBROOT=/opt/XSB
$ export
PATH=$PATH:$MULVALROOT/bin:$MULVALROOT/utils:$XSBROOT/bin:$XSBROOT/build
```

Dále je třeba nainstaloval do systému nástroje pro kompilaci zdrojových kódů v jazycích C++ a Java a nástroje *bison* a *flex* pomocí příkazů:

```
$ apt-get install build-essentials
$ apt-get install bison flex
```

Kompilace nástroje XSB proběhne po zadání příkazů:

```
$ cd /opt/XSB/build
$ ./configure
```

```
$ ./makexsb
```

Nástroj XSB je nyní možné spustit příkazem:

```
$ /opt/XSB/bin/xsb
```

a ukončit zadáním příkazu *halt.* (s tečkou na konci).

Kompilace nástroje MulVAL proběhne po zadání příkazů:

```
$ cd /opt/mulval
$ make
```

Pokud výstup příkazu *make* obsahuje *javac: Command not found*, znamená to, že v systému není nainstalovaná Java nebo se nenachází v proměnné prostředí *PATH*. V takovém případě je třeba nainstalovat do systému JDK, proměnná prostředí *PATH* pak musí obsahovat adresář, ve kterém je umístěn příkaz *javac*.

V případě použití OpenJDK, bude pravděpodobně potřeba nainstalovat i *dom4j.jar* balíček. Nejprve je třeba přejít do adresáře pro externí javovské balíčky, který má obvykle stabilní formu */usr/lib/jvm/<name of java>/jre/lib/ext*, a pak stáhnout samotný balíček pomocí:

```
$ wget -P .
https://github.com/dom4j/dom4j/releases/download/dom4j_1_6_1/dom4j-1.6.1.jar
```

Pokud kompilace proběhla úspěšně, je možné MulVAL otestovat pomocí příkazů (v adresáři *MULVALROOT*, typicky */opt/mulval*):

```
$ cd testcases/3host/
$ graph_gen.sh input.P -l -p
```

## Instalace

Po stáhnutí, respektive dekomprimaci je možné software pro fázi Decide (pod interním názvem *attack graph component*) nainstalovat příkazem *pip install*:

```
$ pip install . -r requirements.txt
```

Příkaz v tomto tvaru sa použije, pokud se sa nacházíme v adresáři, který obsahuje soubor *setup.py* a *requirements.txt*. Tečka uvedená za klíčovým slovem *install* indikuje, že se instaluje z aktuálního adresáře. Aktuální adresář sa dá nahradit za jakoukoliv cestu v systému souborů. Dále v příkazu *pip install* je podobně použita možnost specifikovat soubor s požadavky pomocí přepínače *-r*, který je následovaný cestou k souboru s požadavky.

## Nastavení

Před spuštěním software je třeba provést prvotní nastavení. V adresáři *example_data* se nachází soubor *conf.ini*, ve kterém je třeba nastavit následující proměnné:

- Proměnné *mulval_root* a *xsb_root* je třeba nastavit na stejnou hodnotu jako proměnné prostředí *MULVALROOT* a *XSBROOT* použité během instalace prerekvizit, typicky */opt/mulval* a */opt/xsb*.
- Proměnná *mulval_dir* odkazuje na adresář, ve kterém budou vytvořeny soubory nutné pro generování grafů útoků.
- Proměnná *interaction_rules_file* odkazuje na soubor *crusoe_rules.P* v adresáři *example_data*. Tuto proměnnou je třeba přenastavit pouze v případě, že dojde k přejmenování příslušného souboru.

# Uživatelská dokumentace

Uživatelská dokumentace software pro podporu rozhodování při řešení bezpečnostního incidentu se skládá ze tří částí. Nejprve je popsán technický postup modelování misí a jejich vložení do databáze. Obecné postupy tvorby modelu mise jsou popsány v popisu výsledku, zde se nachází pouze uživatelský návod k zavedení popisu mise do systému. Následně je popsáno použití nástroje pro podporu rozhodování pomocí příkazové řádky. Poslední částí je stručný popis užití nástroje prostřednictvím kontrolního panelu, který je součástí výsledku "Webová aplikace pro vizualizaci bezpečnostní situace v počítačové síti" ze stejného projektu.

## Vložení popisu mise

V rámci *neo4j-rest* je implementovaný endpoint *missions*, POST na tento endpoint vytvoří v databázi nový uzel typu *:Mission* a také všechny uzly ostatních typů (t.j., *:Component*, *:Host* a *:IP*), které se ve vstupním JSON souboru nacházejí, za předpokladu, že předtím neexistovaly v databázi. Tento JSON soubor obsahuje *constrained AND-OR tree* pro danou misi a musí obsahovat data ve formátu specifikovaném v ostatních částech této technické zprávy (sekce *Vstupní data* v kapitole *Programátorská dokumentace*).

Pro samotné vložení mise do databáze je možné použít následující příkaz (případně upravený podle potřeby):

```
$ curl --header "Content-Type: application/json" --data '{"nodes":
{"missions": [{"id": 1, "name": "Web mission", "criticality": 7,
"description": "Mission is responsible for proper functionality of a web
server supported by a database server."}], "services": [{"id": 11, "name":
"Webserver service"}, {"id": 12, "name": "DB Server service"}],
"aggregations": {"or": [ 3 ], "and": [ 2 ]}, "hosts": [{"id": 20, "hostname":
"host2.domain.cz", "ip": "128.228.250.67"}, {"id": 21, "hostname":
"host1.domain.cz", "ip": "128.228.251.133"}, {"id": 22, "hostname":
"host3.domain.cz", "ip": "128.228.123.47"}]}, "relationships": {"one_way": [{
"from": 1, "to": 2 }, { "from": 2, "to": 11 }, { "from": 2, "to": 12 }, {
"from": 11, "to": 20 }, { "from": 12, "to": 3 }, { "from": 3, "to": 21 }, {
"from": 3, "to": 22 }], "two_way": [], "supports": [{"from": "Web mission",
"to": "Webserver service"}, {"from": "Web mission", "to": "DB Server
service"}], "has_identity": [{"from": "Webserver service", "to":
"host2.domain.cz"}, {"from": "DB Server service", "to": "host1.domain.cz"},
{"from": "DB Server service", "to": "host3.domain.cz"}]}}'
http://localhost:8000/rest/missions
```

V tomto *curl* příkazu je nejprve specifikován typ odesílaných dat (*Content-Type*) jako JSON a samotná data jsou předávaná pomocí přepínače *--data*. Uvedený příkaz obsahuje minimalistický příklad předávaného JSON souboru odpovídajícího formátu popsanému v ostatních částech této technické zprávy. Protože JSON soubor obsahuje uvozovky ("), je třeba v sekci *--data* vymezit samotná data například pomocí jednoduchých uvozovek (').

Dále je třeba příkazu předat správnou URL cestu k endpointu na vytváření misí, která odpovídá konkrétnímu nastavení REST API v produkčním prostředí a nemusí sa shodovat s URL cestou uvedenou v předcházejícím příkladě. V případě, že REST API je zabezpečené uživatelským jménem a heslem, je třeba sa autentizovat, např. použitím přepínače *--user*.

Uživatelsky nejpřívětivější je však JSON předat prostřednictvím webového prohlížeče. Django REST framework poskytuje tzv. *Web browsable API*, což zjednodušeně znamená, že při zadání URL adresy endpointu *missions* se zobrazí formulář, ve kterém se do pole *Content* zkopíruje obsah JSON souboru a ten je při uskutečnění operace POST přeposlán do REST API podobným způsobem jako pomocí nástroje curl.

Uzel mise vytvořený v databázi bude potom obsahovat atributy *name*, *criticality*, *description* a obsah souboru se bude nacházet v parametru s názvem *structure*. Dále budou vytvořeny uzly ostatních typů a jejich relace ve tvaru:
*(:Mission)<-[:SUPPORTS]-(:Component)-[:PROVIDED_BY]->(:Host)<-[:IS_A]-(:Node)-[:HAS_ASSIGNED]->(:IP).*

## Použití nástroje pomocí příkazové řádky

Pro správné fungování komponenty je třeba nainstalovat softwarové nástroje MulVAL a XSB na základě návodu, který se nachází v předchozí kapitole nebo v souboru README.md. Pak je možné postupovat podle následujících instrukcí.

Vstupní metodou celého analytického procesu je metoda analytical_process() v souboru *process.py*. Tato funkce očekává na vstupu dva argumenty, heslo k Neo4j databázi a adresu, na které beží databázový konektor *bolt*.

```
>>> from attack_graph_component import process
>>> process.analytical_process("password", "bolt://localhost:7687")
```

Na to, aby komponenta po spuštění korektně fungovala je třeba, aby se data v databázi shodovala s datovým modelom CRUSOE a všechny proměnné v konfiguračním souboru *conf.ini* byly správně nastavené. Kromě toho je potřeba, aby měl proces dostatečná oprávnění pro vytváření souborů ve složce *mulval_dir* nastavené v konfiguračním souboru.

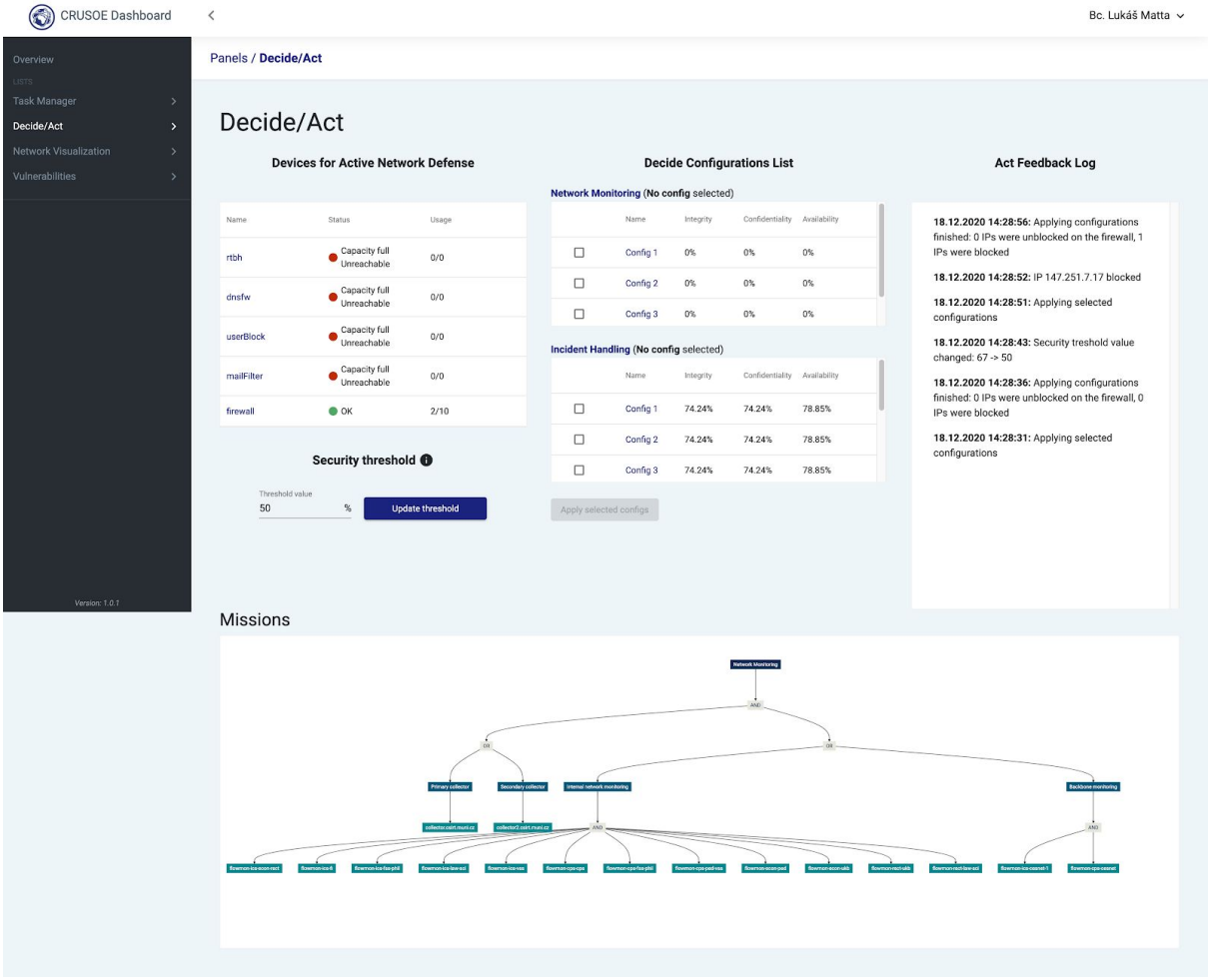
Do funkce analytical process je možné jako třetí parametr předat vlastný logger pomocí klíčového slova *logger*:

```
>>> process.analytical_process("password", "bolt://localhost:7687",
logger=own_logger.get_logger())
```

Komponenta defaultně používá jako logovací nástroj pythonovský balíček *structlog*.

## Ovládání nástroje prostřednictvím kontrolního panelu

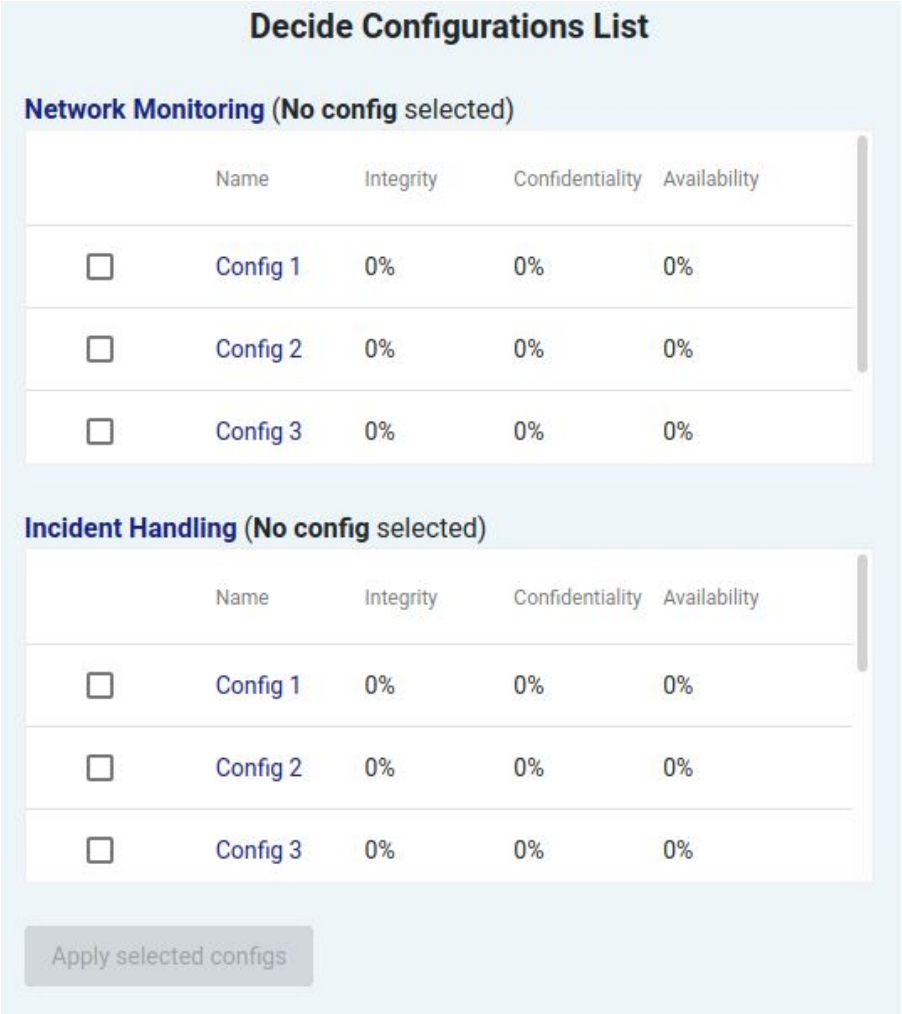
Software pro podporu rozhodování (Decide) byl v rámci projektu CRUSOE propojen s webovou aplikací pro vizualizace bezpečnostní situace (Orient). Díky tomu je možné pohodlně přistupovat ke vstupním datům a výsledkům nástroje Decide. Vzhledem k dalším vazbám na software pro aplikaci reaktivních opatření (Act) je software Orient vybaven panelem, který sdružuje ovládání nástrojů Decide i Act. Z pohledu nástroje Decide jsou zásadní tři části sdruženého ovládacího panelu: seznam misí a konfigurací, nastavení security threshold a vizualizace modelu mise. Tyto části jsou dále popsány podrobněji. Pohled na celý panel je k nahlédnutí na Obrázku 2.




Obrázek 2: Snímek obrazovky s panelem Decide/Act.

## Seznam misí a konfigurací

Seznam misí a konfigurací obsahuje soupis všech misí, se kterými pracuje nástroj Decide. Ke každé misi je dále uveden soupis možných konfigurací. Panel seznamu misí a konfigurací je zobrazen na Obrázku 3.




Obrázek 3: Seznam misí a jejich konfigurací.

Pod názvem mise je vždy zobrazen seznam možných konfigurací spolu s mírou ohrožení pro tři sledované parametry - integritu, důvěrnost, a dostupnost. Kliknutím na název mise se zobrazí popis dané mise a hodnota její kritičnosti (criticality), tj. míry důležitosti pro organizaci. Kliknutím na název konfigurace je pak zobrazena mapa, na níž jsou červeně vyznačeny stroje, jež konfigurace vyžaduje zablokovat. Checkbox u každé konfigurace umožňuje tuto konfiguraci vybrat a tlačítkem Apply selected configs provést blokování a odblokování na prvcích aktivní obrany.

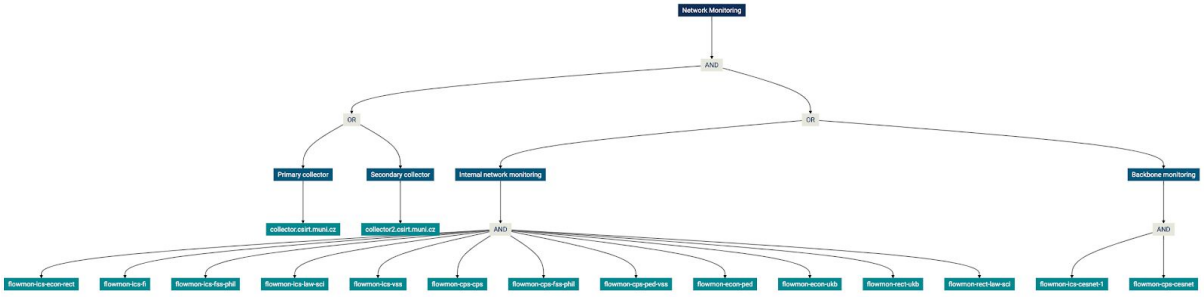
## Security threshold

Každý stroj nacházející se v síti chráněné systémem CRUSOE dostává hodnocení ve třech sledovaných parametrech - míra ohrožení integrity, důvěrnosti, a dostupnosti. Více o těchto parametrech je možné zjistit v popisu výsledku a souvisí s filtrováním výsledků po vybrání nejodolnější konfigurace před jejím doporučením. Software fáze Act pracuje se souhrnnou mírou ohrožení stroje, která je spočtena jako aritmetický průměr míry ohrožení integrity, důvěrnosti, a dostupnosti daného stroje. Dosahuje tedy hodnot z intervalu <0, 100> a její uplatnění je rozepsáno v dokumentaci nástroje Act. Ukázka je uvedena na Obrázku 4.


Obrázek 4: Nastavení hodnoty Security threshold.

### Vizualizace popisu mise

Spodní část panelu je vyhrazena přehledné vizualizaci popisu misí, které jsou zaznamenány v databázi. Mise jsou vykresleny v hierarchické grafové struktuře odpovídající formálnímu popisu modelu mise. V příslušné části panelu je možné vizualizace mise přibližovat a oddalovat, což umožňuje přehledně si prohlédnout celý model mise i jednotlivé její detaily. Příslušná část panelu je k nahlédnutí na Obrázku 5.




Obrázek 5: Vizualizace popisu mise.

# Programátorská dokumentace

V této sekci je popsána programátorská dokumentace. Nejprve je představena struktura balíčku, která objasňuje rozdělení dílčích funkcionalit celého procesu do jednotlivých modulů. Podrobný popis implementace analytického procesu pro podporu rozhodování je doplněn podkapitolami věnovanými vstupním datům, se kterými komponenta pracuje, a výstupům, které komponenta vyprodukuje buď ve formě návratové hodnoty programu nebo jako záznam do grafové databáze. Poslední podkapitola je věnována testům.

## Struktura SW fáze Decide

Obsah komponenty je možné rozdělit do třech kategorií. První tvoří soubory, které implementují analytický proces (*bayes.py, components.py, generator.py, process.py, run_mulval.py* a datový soubor *crusoe_rules.P*). Dále soubory s testy se nacházejí v adresáři *test*. Poslední skupinu tvoří zbývající soubory pro potřebné formální náležitosti - konfigurační soubor (*conf.ini*), pythonovské soubory potřebné pro instalaci (*__init__.py*, *setup.py*, *requirements.txt*) a popisný soubor (*README.md*).

```
attack-graph-component
├── attack_graph_component
    ├── data
        └── conf.ini
        └── crusoe_rules.P
    ├── test
        ├── test_data
        └── attack_graph_test.py
        └── config_decomposition_test.py
        └── neo4j_test_client.py
    └── bayes.py
    └── components.py
    └── generator.py
    └── process.py
    └── run_mulval.py
├── __init__.py
├── requirements.txt
├── setup.py
├── README.md
```

### Modul Bayes

Tento modul obsahuje funkcionalitu týkající sa konverze vytvořeného grafu útoků na Bayesovský graf útoků, který ke každému vrcholu přiřazuje tabulku rozdělení podmíněných pravděpodobností. Na základe vytvořeného Bayesovského grafu útoků je vykonána inference výsledné nepodmíněné pravděpodobnosti pro každý z cílů útoku nacházející se v grafu.

### Modul Generator

V tomto modulu se nacházejí metody, které ze vstupních dat nacházejících sa v grafové databázi a odpovídajících datovému modelu CRUSOE vytvoří vstupní soubor pro nástroj MulVAL[1]. Tento vstupný soubor obsahuje fakty (pojem z terminologie logického programování), které budou zpracované nástrojem MulVAL.

### Modul Process

Modul obsahuje hlavní funkci pro analytický proces a funkcionalitu potřebnou pro zpracování jednotlivých konfigurací misí nacházejících se v databázi. Dále zabezpečuje kombinaci a výběr výsledných pravděpodobností a vytvoření výsledků pro jednotlivé konfigurace v grafové databázi.

### Modul Components

Tento modul obsahuje funkcionalitu pro zjišťování dostupných konfigurací dané mise.

### Modul Run_Mulval

Tento modul zprostředkovává volání příkazového řádku, prostřednictvím kterého je vygenerovaný graf útoků nástrojem MulVAL.

### Adresář test

Adresář test obsahuje soubory s testy, o kterých pojednává jedna z následujících podkapitol.

### Adresář data

Adresář data obsahuje soubory *conf.ini* a *crusoe_rules.P*. Prvý soubor - *conf.ini* - je konfigurační soubor, v kterém je potřeba nastavit několik proměnných:

- umístění nástrojů MulVAL a XSB (*mulval_root / xsb_root*),
- cestu ke složce na vytváření grafu útoků (*mulval_dir*) ,
- výchozí název souboru obsahujícího vlastní pravidla pro generování grafu útoků (*interaction_rules_file*),

Druhý soubor - *crusoe_rules.P* - obsahuje pravidla pro generování grafu útoků. Tyto pravidla budou popsána v podkapitole o implementaci analytického procesu.

## Vstupní data

Všechny vstupy načítá decide komponenta z Neo4j databáze, která je uspořádána podle datového modelu CRUSOE[2]. Jedná se konkrétně o informace týkající se:

- zranitelností ve formátu CVE (stroj, software, vzdálená zneužitelnost, dopad, CVSS),
- mise, kterou organizace zabezpečuje,

---

[1] OU, Xinming; GOVINDAVAJHALA, Sudhakar; APPEL, Andrew W. MulVAL: A Logic-based Network Security Analyzer. In: *USENIX security symposium*. 2005. p. 113-128.

[2] KOMÁRKOVÁ, Jana, Martin HUSÁK, Martin LAŠTOVIČKA a Daniel TOVARŇÁK. CRUSOE: Data Model for Cyber Situation Awareness. In Proceedings of the 13th International Conference on Availability, Reliability and Security. Hamburg: ACM, 2018. s. "36:1"-"36:10".

- stavu sítě - IP adresy strojů v síti, otevřené porty, služby a softwarové produkty běžící na daném stroji a jiné.

Data o zranitelnostech a stavu sítě byla do databáze přidána ostatními komponentami v rámci předcházejících fází OODA cyklu. Specifickým způsobem sa do databáze přidává vstupní JSON popisující požadavky, které mise klade na infrastrukturu. K tomu je třeba využít REST API (balíček *neo4j-rest*), přes které se daný JSON nahrává. Popis obsahuje *constrained AND/OR tree*, ve kterém jsou pomocí logických formulí AND a OR zachycené vztahy mezi hosty / komponentami a službami, které mise vyžaduje. Grafická podoba tohto stromu je znázorněna na obr. 1.

Samotný JSON potom obsahuje vrcholy (*"nodes"*) a vztahy mezi nimi (*"relationships"*). Vrcholy lze rozdělit do čtyř skupin (viz Obr. 1) - jedná se buď o mise (na obrázku zelenou barvou), služby (na obrázku modrou barvou), AND/OR uzly nebo systémové komponenty (na obrázku červenou barvou). Všechny vrcholy v rámci jednoho JSON musí mít unikátní ID. Důležitou sekcí jsou *"aggregations"*, kde jsou shrnuty vztahy AND a OR takovým způsobem, že každému AND / OR vrcholu (viz Obr. 1) je přiřazeno ID a toto ID je následně použito v rámci *one_way* relationships.

Vztahy (*"relationships"*) popisují hrany grafu znázorněné jednosměrnými (*"one_way"*) nebo obousměrnými šipkami (*"two_way"*). Formát definuje sekci *two_way*, ale v rámci implementace se místo těchto manuálně specifikovaných dat používají dynamické informace z monitoringu sítě, které jsou do databáze přidávány komponentami fáze Observe. Sekce *"supports"* a *"has_identity"* jsou pomocné, pro rozlišení toho, jaký vztah sa bude nacházet mezi vrcholy v grafové databázi (*"SUPPORTS"* mezi misí a službami, resp. *"HAS_IDENTITY"* mezi službou a komponentami).

```
{
  "nodes": {
    "missions": [
      {
        "id": 1,
        "name": "Network Monitoring",
        "criticality": 8
      },
      {
        "id": 2,
        "name": "Incident Handling",
        "criticality": 5
      }
    ],
        "services": [
      {
        "id": 11,
        "name": "Internal network monitoring"
      },
      {
        "id": 12,
        "name": "Backbone monitoring"
      },
      {
        "id": 13,
        "name": "Primary collector"
      },
      {
        "id": 14,
        "name": "Secondary collector"
      },
...],
  "aggregations": {
    "or": [ 4, 5, 7, 23, 26 ],
    "and": [ 3, 6, 8, 29, 40, 50 ]
  },
  "hosts": [
    {
      "id": 20,
      "hostname": "collector.csirt.muni.cz",
      "ip": "147.251.14.52"
    },
```

```
      {
        "id": 21,
        "hostname": "collector2.csirt.muni.cz",
        "ip": "147.251.14.53"
      },
      {
        "id": 22,
        "hostname": "rt.csirt.muni.cz",
        "ip": "147.251.14.49"
      },
     ]
  ...],
"relationships": {
    "one_way": [
      { "from": 1, "to": 3 },
      { "from": 3, "to": 4 },
      { "from": 3, "to": 5 },
      { "from": 4, "to": 11 },
      { "from": 4, "to": 12 },
      { "from": 5, "to": 13 },
            ….
],
    "two_way": [
    ],
    "supports": [
      {"from": "Network Monitoring", "to":
"Internal network monitoring"},
      {"from": "Network Monitoring", "to":
"Backbone monitoring"},
      {"from": "Network Monitoring", "to":
"Primary collector"},
            ...
    ],
    "has_identity": [
      {"from": "Internal network monitoring", "to":
"flowmon-ics-econ-rect"},
      {"from": "Internal network monitoring", "to":
"flowmon-ics-fi"},
      {"from": "Internal network monitoring", "to":
"flowmon-ics-fss-phil"},
            ...
    ]
  }
}
```


## Implementace analytického procesu

V této podkapitole je blíže popsán analytický proces, nejprve v hrubých obrysech, následně jsou pak detailně popsány jednotlivé části procesu.

### Obecný popis algoritmu

Hrubý náčrt analytického procesu vypadá následovně. Algoritmus nejprve zjistí, které mise se nacházejí v databázi. Ke každé z těchto misí načte vstupní JSON s popisem konfigurace mise, který sa nachází v uzlu typu *:Mission* v parametru *:structure*. Na základě vstupní JSON struktury jsou pro každou misi zjištěny dostupné konfigurace. Konfigurace je seznam zařízení v síti či softwarových komponent, které poskytují požadovanou funkcionalitu na to, aby síť dokázala zabezpečiť bezproblémový chod mise. Každé z těchto zařízení nebo softwarových komponent je reprezentováno pomocí informací uvedených v sekci *hosts* ve vstupním JSON. Vzhledem k tomu, že každá konfigurace obsahuje více zařízení či komponent, které mohou být cílem útoku, proběhne pro každou z nich analytický proces pro podporu rozhodování, který vytvoří Bayesovský graf útoku a určí výslednou pravděpodobnost pro každý cíl útoku zvlášť. Tyto pravděpodobnosti jsou na závěr zkombinovány a pro každou misi se vybere ta konfigurace, která je nejodolnější. Výběr nejodolnější konfigurace mise je výstupem analytického procesu. Komponenta taktéž uloží všechny částečné výsledky pro jednotlivé konfigurace do databáze.

Samotné zpracování jedné konfigurace (t.j. jedna iterace analytického procesu) sa skládá ze tří kroků:

1. Nejprve je vygenerován vstupní soubor pro nástroj MULVAL.
2. Následně je pomocí nástroje MULVAL vygenerován graf útoků.
3. Graf útoků je zkonvertován na Bayesovský graf útoků a je zjištěna výsledná nepodmíněná pravděpodobnost (tzv. inference výsledné pravděpodobnosti).

### Zjištění konfigurací

Požadavky kladené na misi jsou reprezentovány pomocí tzv. *constrained AND/OR tree*, stromu omezujících podmínek s logickými vazbami. Tento strom je možné zapsat ve formátu JSON, jehož struktura byla popsána v předchozí podkapitole. V modulu *components.py* se nachází funkce *get_mission_components()*, která procházením stromu zjišťuje, které konfigurace jsou dostupné pro danou misi. Postup je následující:

1. Strom, který byl zadán na vstupu, je procházen z kořene stromu k listům ve směru hran.
2. Při průchodu uzlem typu AND jsou do konfigurace v mezivýsledku přidány všechny vrcholy, na které daný uzel odkazuje.
3. Při průchodu uzlem typu OR jsou v mezivýsledku vytvořeny varianty konfigurace pro každý vrchol, na který daný OR uzel odkazuje. Každá varianta konfigurace obsahuje právě jeden z vrcholů.
4. Průchod stromem je zastaven, pokud se v dané konfiguraci nacházejí pouze komponenty (*Supportive Cyber Components,* na vzorovém obrázku 1 znázorněny červenou barvou).

### Vygenerování vstupního souboru

Vzhledem k tomu, že v komponente využíváme nástroj MulVAL, je třeba příkaz na vygenerování grafu útoků zadat způsobem podporovaným tímto nástrojem, tedy pomocí vstupního souboru a případně i souboru s pravidly, na základě kterých je graf útoku vygenerován. Vstupní soubor však musí obsahovat fakta, která zachovávají syntaxi definovanou v souboru s pravidly. Samotná funkcionalita, nacházející se v souboru *generator.py*, bere data z databáze (utvořené podle datového schématu CRUSOE) a zapisuje informace o otevřených portech, zranitelnostech, síťových službách na strojích a stavu sítě pomocí predikátů definovaných v souboru s pravidly *crusoe_rules.P*.

Výchozí pravidla jsou dostupná přímo v nástroji MulVAL, kam byla zanesena tvůrci nástroje na základě zkušeností odborníků v oblasti kyberbezpečnosti. Pravidla pro generování grafu útoku byla přebrána ze souboru výchozích pravidel pro MulVAL, ze kterých byla vybrána a případně přizpůsobena pravidla relevantní pro zkoumanou doménu. Přizpůsobený soubor pravidel se nachází v souboru *crusoe_rules.P*. Dále byly přidány predikáty popisující ztrátu důvěrnosti, integrity a dostupnosti aplikace nebo celého systému.

Konkrétně došlo k následujícím změnám:

- Původní pravidla obsahovala v predikátu *vulExists* (existence zranitelnosti) pouze výsledek typu *privEscalation* (eskalace oprávnění). Namísto privEscalation byla

použita taxonomie dopadů navržená v dříve vydaném článku.[3] V současné podobě tak rozlišujeme, zda má zranitelnost za následek vykonání kódu s oprávněním běžného uživatele či správce nebo eskalaci oprávnění, případně ztrátů důvěrnosti, dostupnosti nebo integrity v systému či v aplikaci.

- Nástroj MulVAL pracuje při generování grafu útoků pouze s jedním možným cílem, kterým je *execCode* (vykonání kódu), což neodpovídalo požadavkům projektu, podle kterých je třeba zohlednit ztrátu důvěrnosti, dostupnosti a integrity. Z toho důvodu byly přidány predikáty *appConfLoss(), appIntegLoss()* a *appAvailLoss()* pro ohrožení aplikací a *sysConfLoss(), sysIntegLoss()* a *sysAvailLoss()* pro ohrožení systémů.
- Další změny zahrnují vytvoření predikátu *inSubset* (příslušnost k podsíti) a nových pravidel pro predikáty související s ohrožením důvěrnosti, dostupnosti a integrity.

### Vygenerování grafu útoků

Aby bylo možné vygenerovat graf, je třeba nastavit systémové proměnné MULVALROOT a XSBROOT, které popisují umístění nástrojů MulVAL a XSB, a přidat je do systémové proměnné PATH. Na základě vytvořeného vstupního souboru a souboru s pravidly dodaného nástrojem MulVAL je vygenerován graf útoku. V nástroji nepoužíváme možnost vygenerovat PDF soubor s grafem, ale pouze generování grafu do souboru typu TXT. Vrcholy grafu jsou uloženy do souboru VERTICES.CSV a hrany grafu do souboru ARCS.CSV. Soubory jsou uloženy v adresáři *mulval_dir*, který je definován v konfiguračním souboru.

### Sestrojení Bayesovského grafu útoků a inference pravděpodobnosti

Ze souborů VERTICES.CSV a ARCS.CSV, které byly vytvořeny v předchádzejícím kroku, jsou načteny vrcholy a hrany grafu útoků, ze kterého je vytvořena Bayesovská síť. Při vytváření Bayesovské sítě je však třeba pro každý vrchol zadat tzv. rozdělení podmíněné pravděpodobnosti ve formě tabulky (tzv. *Tabular Conditional Probability Distribution*). Při řešení problému byly identifikovány čtyři obecné případy toho, jak může tabulka vypadat. Tyto případy vycházejí z logiky nástroje MulVAL, který uzly grafu rozděluje na uzly typu AND, OR a LEAF. Vrcholy typu LEAF jsou vrcholy, které sa v grafu nacházejí na pozici listů, t.j. nevchází do nich žádná hrana, pouze z nich hrana vychází. Vrcholy typu AND jsou vrcholy, které odpovídají realizaci některého pravidla. Ve výstupu nástroje MulVAL je možné je rozpoznat podle toho, že jejich název začíná klíčovým slovem “RULE”, např. “RULE 33” indikuje, že bylo použito pravidlo č. 33. Vrcholy typu OR jsou jednoduše řečeno ty vrcholy, které nejsou ani listy ani vrcholy typu AND.

Výše uvedené rozdělení podmíněné pravděpodobnosti ve formě tabulky je v jednotlivých případech následující:

- Pro vrcholy typu LEAF platí
    - Vrchol typu LEAF, který neodpovídá faktu o existenci zranitelnosti dostane přiřazenou pravděpodobnost, že útočník dosáhne exploitu, pokud jev existuje, rovnou 1.0.

[3] KOMÁRKOVÁ, Jana, Lukáš SADLEK a Martin LAŠTOVIČKA. Community Based Platform for Vulnerability Categorization. In NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium. Taipei, Taiwan: IEEE, 2018., 2 s. ISBN 978-1-5386-3416-5.

    - Vrchol typu LEAF, který odpovídá faktu o existenci zranitelnosti s přiřazeným CVE dostane pravděpodobnost, že útočník dosáhne exploitu, pokud jev existuje, rovnou tzv. *exploitability score* z CVSS verze 3[4], kterou je možné získat pro jednotlivé CVE z dat nacházejících se v databázi NVD[5].
- Pro vrchol typu AND s *n* rodiči platí:
    - Pravděpodobnost, že se útočníkovi podaří vykonat další krok je ve všech (2^n)-1 případech, kde aspoň jeden z rodičů má ohodnocení FALSE, nulová.
    - V případě, že všichni rodiče mají ohodnocení TRUE, je pravděpodobnost úspěchu rovná 0.8. Tato pravděpodobnost byla převzata z výchozího nastavení nástroje MulVAL, kde byla stanovena na základě zkušeností odborníků.
- Pro vrchol typu OR s *n* rodiči platí:
    - Pravděpodobnost, že se útočníkovi podaří vykonat další krok, je v případě, kdy všichni rodiče mají ohodnocení FALSE, nulová.
    - Ve všech (2^n)-1 případech, kdy alespoň jeden z rodičů má ohodnocení TRUE, je pravděpodobnost úspěchu rovná 1.0, což je to způsobeno tím, že v tomto vrcholu nedochází k realizaci pravidla, pravděpodobnost se pouze propaguje směrem k cíli útoku.

K výpočtu výsledné inference pravděpodobnosti se používá balíček pgmpy[6], který poskytuje efektivní implementaci operací pro práci s Bayesovskými sítěmi. Inference je vypočtena samostatně pro každou složku z triády důvěrnost, dostupnost, integrita a výsledkem je tak trojice pravděpodobností.

### Kombinace pravděpodobností a výstup pro vizualizaci

Postupem popsaným v předcházející podkapitole získáme pro jednu cílovou komponentu trojici pravděpodobností pro ohrožení důvěrnosti, integrity a dostupnosti. Jako výslednou trojici pravděpodobností pro konfigurace mise je vybrána nejhorší varianta mezi mezivýsledky, které byly napočítány pro jednotlivé komponenty v konfiguraci.

Pokud je získána pravděpodobnost pro jednotlivé konfigurace mise, je přikročeno k výběru nejodolnější konfigurace. Za nejodolnější konfiguraci je vybrána ta konfigurace, která má nejlepší ohodnocení pravděpodobnosti a seznam komponent této konfigurace je vrácen v návratové hodnotě.

Jak vyplývá z předcházejících dvou odstavců, porovnávání trojic pravděpodobností může být nejasné. Například nelze obecně určit horší variantu při porovnání trojic jako jsou (0.7, 0.6, 0.5) a (0.9, 0.2, 0.2). V takovém případě má první trojice vyšší hodnotu pravděpodobnosti narušení integrity a dostupnosti, zatímco druhá má vyšší hodnotu pravděpodobnosti narušení důvěrnosti. Tento problém řeší tzv. *utility function*, která umožňuje převést trojice hodnot na jednu porovnatelnou hodnotu. Jednoduchými implementacemi *utility function* mohou být výběr maximální či průměrné hodnoty. V rámci implementace komponenty je jako utility function použit součet hodnot v trojici, což fakticky odpovídá neváženému průměru.

---

[4] https://www.first.org/cvss/specification-document
[5] https://nvd.nist.gov/general
[6] https://pgmpy.org/

Často však může být výhodnější přiřadit jednotlivým složkám (důvěrnost, integrita, dostupnost) váhy a počítat vážený průměr odvíjející se od konkrétních požadavků sledované infrastruktury a prioritám jejich správců.

Abychom umožnili doménovým expertům, například správcům kritické informační infrastruktury, vnést do procesu rozhodování svůj vlastní názor a zkušenosti, jsou všechny mezivýsledky (trojice pravděpodobností), ke kterým se komponenta dopracovala, zapsané k jednotlivým konfiguracím do databáze. Uživatel komponenty tak má šanci nahlédnout do mezivýsledků a finální rozhodnutí upravit dle svých zkušeností a priorit.

## Výstup komponenty

Výstup komponenty fáze Decide má dvě podoby, návratovou hodnotu programu a data vytvořená v grafové databázi, která odpovídají mezivýsledkům pro jednotlivé konfigurace.

### Návratová hodnota

Návratová hodnota může vypadat například následovně:

{'Network Monitoring': {'configuration': {58, 51, 20, 53, 54, 52, 55, 57, 56, 59, 60, 61, 62}, 'probability': (0, 0, 0)}, 'Incident Handling': {'configuration': {32, 20, 22, 24, 27, 30, 31}, 'probability': (0.7474, 0.7474, 0.7781)}}

V této návratové hodnotě se nachází datová struktura s popisem dvou misí, *Network Monitoring* a *Incident Handling*. Pro první misi, *Network Monitoring,* existuje konfigurace s pravděpodobností kompromitace (0, 0, 0), což znamená, že na základě získaných dát o zranitelnostech, o službách běžících na daných zařízeních a o otevřených portech nebyl k dané konfiguraci vygenerován žádný graf útoků, který by popisoval cestu útočníka vedoucí ke kompromitaci některého zařízení. Je třeba pamatovat na to, že daný výstup neznamená, že daná zařízení nemohou být kompromitována, ale znamená pouze to, že na základě získaných dat sa nepodařilo najít žádný postup, kterým by mohl útočník dané zařízení kompromitovat.

Pro druhou misi, *Incident Handling,* byla jako nejodolnější konfigurace vybrána konfigurace určená kombinací ID jednotlivých zařízení, která danou konfiguraci tvoří. Kompletní informace je možné podle ID dohledat). Na základě dat nasbíraných do databáze nástroji fáze Observe jsme této konfiguraci přiřadili pravděpodobnost zneužití 0.7474 pro důvěrnost (confidentiality), 0.7474 pro integritu (integrity) a 0.7781 pro dostupnost (availability). Všechny pravděpodobnosti se pohybují na škále od 0 do 1.

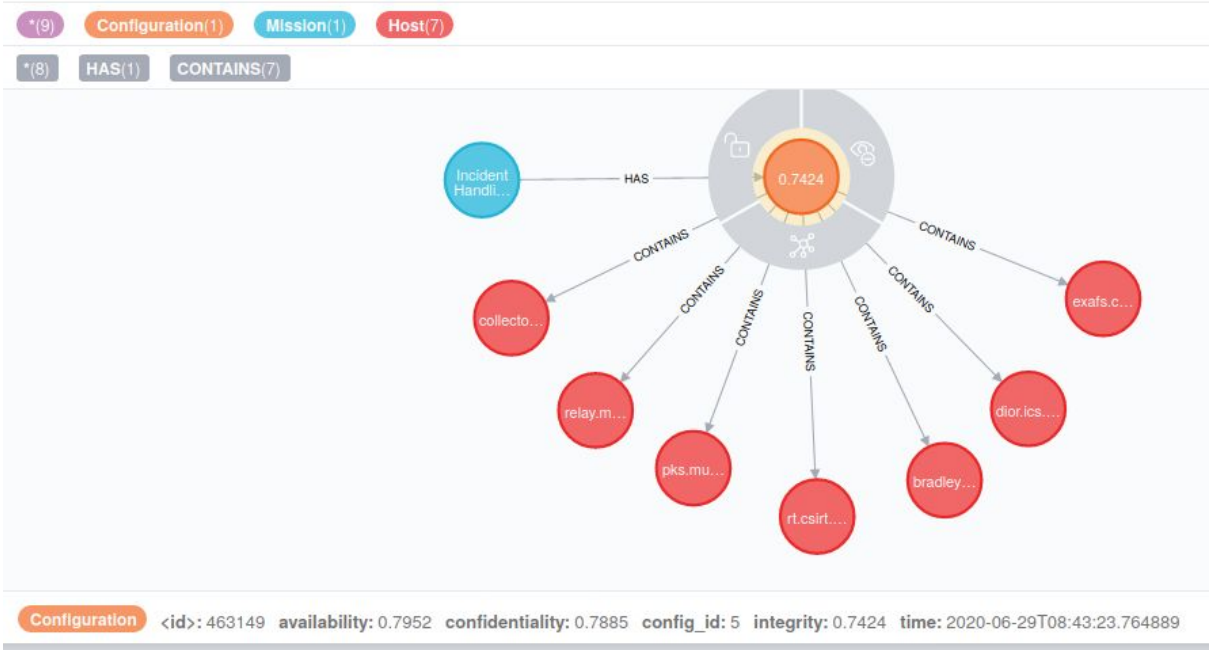
### Formát výsledků v databázi

V rámci analytického procesu počítá software pro podporu rozhodování částečné výsledky pro každou možnou konfiguraci každé mise. Výsledky výpočtů pro každou konfiguraci jsou založeny na výsledcích výpočtů pro každou komponentu v dané konfiguraci, například pro každé zařízení nebo síťovou službu. Výsledky ve všech případech obsahují

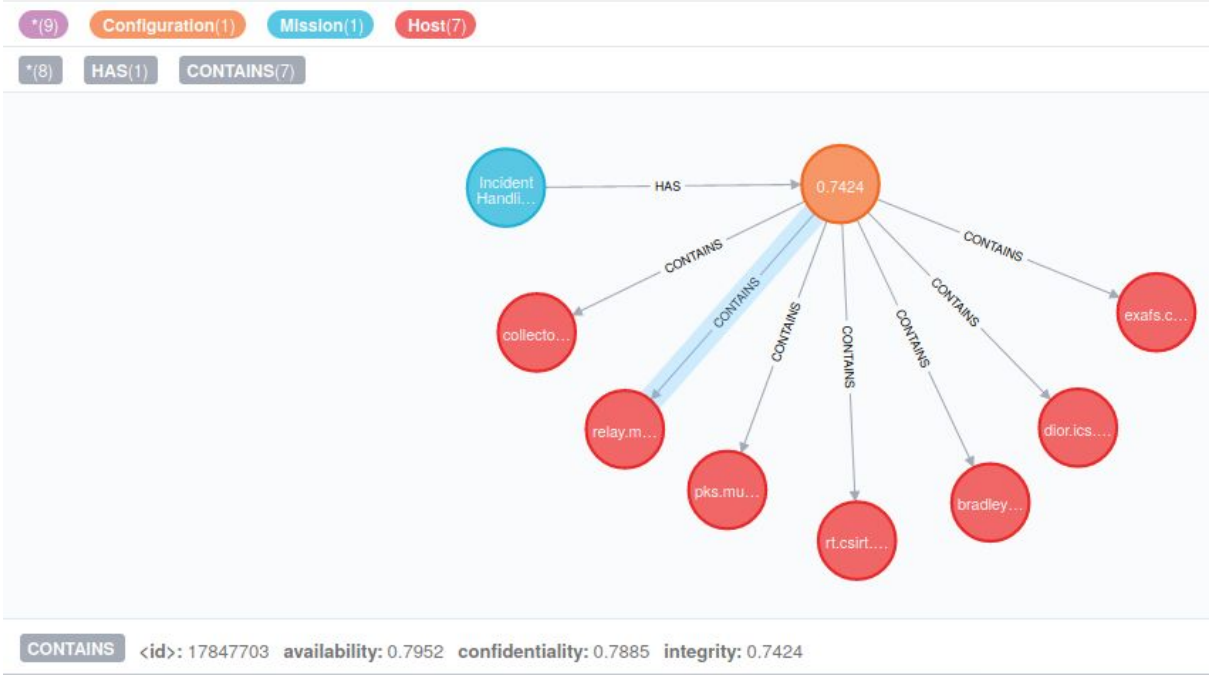
pravděpodobnost kompromitace dané komponenty z pohledu důvěrnosti, dostupnosti a integrity.

Příslušný uzel *:Mission* je v databázi propojen relací *:HAS* s uzlem *:Configuration*. Uzel *:Configuration* je propojen relací *:CONTAINS* se zařízením, na kterém je hostován. Výsledná cesta je *(:Mission)-[:HAS]->(:Configuration)-[:CONTAINS]->(:Host)*. Uzel *(:Configuration)* obsahuje výsledky (pravděpodobnost kompromitace z pohledu důvěrnosti, dostupnosti a integrity), ID konfigurace a časovou známku poslední evaluace. Výsledky pro každé zařízení jsou uloženy na hraně *:CONTAINS*.

Na obrázcích 6 a 7 je zobrazena konkrétní podoba dat v databázi pro jednu konfiguraci vzorové mise s názvem *Incident Handling*. Na obrázku 6 je vidět ohodnocení konfigurace mise, na obrázku 7 pak ohodnocení zařízení v síti, které danou misi podporuje.




Obrázek 6: Formát dat v grafové databázi (ohodnocení konfigurace).




Obrázek 7: Formát dat v grafové databázi (ohodnocení zařízení).

## Testy

Testy uvedené v souboru *attack_graph_test.py* testují vyhodnocování bezpečnostní situace některých základních namodelovaných případů. K namodelovaní těchto případů slouží funkce *create_real_vulnerabilities()* a *create_fake_vulnerabilities()*, které do databáze uloží reálné nebo fiktivní zranitelnosti s daným impaktem. Do databáze jsou dále vloženy prostřednictvím funkce *create_data()* ostatní data potřebná pro vygenerování AG, konkrétně IP adresy, služby běžící na daných strojích a otevřené porty. Pro každý z testovacích modelových příkladů je v rámci testů zkontrolován výběr korektní konfigurace s korektní pravděpodobností.

Test v souboru *config_decomposition_test.py* kontroluje, zda funkcionalita zjišťování dostupných konfigurací pro danou misi a dané vzorové soubory (*missions.json*, resp. *constraint.json*) najde všechny konfigurace ve správném složení.

Soubor *neo4j_test_client.py* obsahuje funkce, které jsou využívané při vytváření dat pro modelové testovací příklady podle datového modelu CRUSOE v grafové databázi Neo4j.

# Poděkování

Práce na software byla podpořena z projektu *Výzkum nástrojů pro hodnocení kybernetické situace a podporu rozhodování CSIRT týmů při ochraně kritické infrastruktury* (VI20172020070) řešeného v programu *Bezpečnostní výzkum České republiky* v letech 2017-2020 na Masarykově univerzitě. Autory software jsou Lukáš Sadlek, Michal Javorník a Martin Husák.