Webová aplikace pro vizualizaci bezpečnostní situace v počítačové síti

Lukáš Matta, 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 *"Webová aplikace pro vizualizaci bezpečnostní situace v počítačové síti"* (dále Orient), 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. *"Webová aplikace pro vizualizaci bezpečnostní situace v počítačové síti"* implementuje fázi Orient, tedy fázi zorientování se v nasbíraných informacích a datech.

Software Orient sestává z webové aplikace, která vhodným způsobem prezentuje a vizualizuje informace sesbírané k tomu určenými nástroji, například nástroji vytvořenými v rámci projektu CRUSOE pro fázi Observe. Informace o kyberbezpečnostní situaci v počítačové síti sestávají především z výčtu zařízení a služeb v dané síti, seznamu nalezených zranitelností a historie bezpečnostních incidentů, vyznačení prvků kritické infrastruktury, seznamu kontaktních údajů, topologie sítě, seznamu uživatelů a jejich oprávnění, vazeb mezi jednotlivými entitami a podobně. Software Orient načítá tato data z databáze a umožňuje uživateli těmito daty procházet a vyhledávat v nich. Různé pohledy na data zrcadlí požadavky kyberbezpečnostích týmů a zkušenosti z praxe. Při řešení kyberbezpečnostích incidentů je často třeba znát kontext. Proto například software Orient obsahuje pohled na entitu v síti (zařízení, službu, uživatele) a jeho nejbližší okolí a vazby, což umožňuje rychle kontext nabrat. Mezi další významné pohledy, které software umožňuje, patří pohled na počty zranitelných zařízení pro každou zranitelnost, což umožňuje plánovat a prioritizovat odstraňování zranitelností. Software dále integruje i vizuální nástroje a kontrolní panely pro snazší ovládání a kontrolu dalších nástrojů vytvořených v rámci projektu CRUSOE.

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 ve formě webové aplikace prezentuje a vizualizuje data o aktuální kyberbezpečnostní situaci v počítačové síti. Webová aplikace nabízí různé pohledy na data sesbíraná jinými nástroji. Mezi významné pohledy na data patří vizualizace kontextu k dané entitě v síti a globální přehled zranitelností a zranitelných zařízení v síti. Webová aplikace dále implementuje ovládací prvky a přehledy o fungování dalším softwarových výsledků stejného projektu. Tyto prvky a přehledy umožňují vizuálně ovládat a kontrolovat sběr dat a jejich další zpracování, čímž vytvářejí alternativu k výchozímu ovládání těchto nástrojů.

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 organizace, které provozují či budují svůj CERT/CSIRT, Security Operation Centre, případně zprostředkovávají kybernetickou bezpečnost jako službu. Výsledek umožňuje uživatelům a provozovatelům sítí a služeb přehledně vizualizovat data o počítačové síti a výstupy nástrojů pro podporu činnosti kyberbezpečnostního týmu. Software umožňuje uživatelům rychleji nahlédnout do data o chráněné síti, zorientovat se v situace a rozhodnout o dalším postupu při řešení bezpečnostního incidentu. Výsledek tak přináší úsporu v čase řešení kyberbezpečnostních problémů, což snižuje nároky na lidské zdroje, případně umožňuje uživatelům zvládat více úkolů za stejný čas. Výsledek tak zkracuje dobu reakce na kyberbezpečnostní incident, čímž přispívá k obecnému zvýšení úrovně kyberbezpečnosti v organizaci. Využití výsledku je licencováno bez poplatku.

Popis výsledku

Webová aplikace pro vizualizaci bezpečnostní situace v počítačové síti (dále označovaný jako software Orient) je webové aplikace, která vhodným způsobem prezentuje a vizualizuje informace sesbírané k tomu určenými nástroji, například nástroji vytvořenými v rámci projektu CRUSOE pro fázi Observe. K tomu zahrnuje společnou funkcionalitu pro všechny vizualizace, například řízení přístupu, přehled panelů a podobně. Informace o kyberbezpečnostní situaci v počítačové síti sestávají především z výčtu zařízení a služeb v dané síti, seznamu nalezených zranitelností a historie bezpečnostních incidentů, vyznačení prvků kritické infrastruktury, seznamu kontaktních údajů, topologie sítě, seznamu uživatelů a jejich oprávnění, vazeb mezi jednotlivými entitami a podobně. Software Orient načítá tato data z databáze a umožňuje uživateli těmito daty procházet a vyhledávat v nich. Různé pohledy na data zrcadlí požadavky kyberbezpečnostích týmů a zkušenosti z praxe. Při řešení kyberbezpečnostích incidentů je často třeba znát kontext. Proto například software Orient obsahuje pohled na entitu v síti (zařízení, službu, uživatele) a jeho nejbližší okolí a vazby, což umožňuje rychle kontext nabrat. Mezi další významné pohledy, které software umožňuje, patří pohled na počty zranitelných zařízení pro každou zranitelnost, což umožňuje plánovat a prioritizovat odstraňování zranitelností. Software dále integruje i vizuální nástroje a kontrolní panely pro snazší ovládání a kontrolu dalších nástrojů vytvořených v rámci projektu CRUSOE.

Software Orient pokrývá dvě základní oblasti - vizualizaci dat a kontrolní panel. Oblast vizualizace dat cílí na způsoby prezentace výstupů projektu CRUSOE uživateli tak, aby rychle získal celkový přehled o stavu sítě a zároveň mohl snadno získat detailní informace o jednotlivých částech sítě. Vizualizační komponenty agregují výstupy ostatních nástrojů vytvořených v rámci projektu. Komponenty vizualizace a prezentace dat jsou propojeny v souhrnné webové aplikaci tak, aby uživateli umožnily získat globální přehled o stavu sítě i detailní pohled na konkrétní stroje. Komponenty kontrolního panelu pak umožňují ovládat nástroje pro sběr a zpracování dat a jiné nástroje, například komponenty podpory rozhodování a aplikace reaktivních opatření pro mitigaci kyberútoků. Oba typy komponent, vizualizace i kontrolní prvky, je také třeba integrovat do jednotného systému. Za tímto účelem byl vytvořen komplexní kontrolní panel, který řeší společnou funkcionalitu, například správu přihlášení a uživatelů, a pomocí přepínatelných panelů umožňuje přístup k jednotlivým komponentám.

Případy užití

Software Orient implementuje řadu různých případů užití, které je vhodné nejprve popsat. Případy užití vycházejí z různých situací a činností, na které narážejí členové kyberbezpečnostních týmů při řešení incidentů. Zde jsou uvedeny požadavky na funkcionalitu software a k nim stručný popis případu užití pro každý z nich. Ne všechny případy užití vyžadují specializovanou komponentu, jak je dále ukázáno v analýze požadavků a návrh komponent, které jsou obsahem dalších částí tohoto dokumentu.

Mapa sítě

První očekávanou funkcionalitou je vizualizace grafu sítě s možností interakce, která umožňuje uživateli zorientovat se v síti a vyhledat v ní příslušné entity. Namísto topologické mapy, jejíž implementace jsou běžně dostupné, by se mělo jednat spíše o mapu sítě z logického pohledu, například se zohledněním podsítí a jiných vazeb mezi zařízeními v síti, než je jejich propojení síťovou linkou. Nabízí se propojení přes stejné správce či uživatele, podobnou charakteristiku zařízení a softwarové vybavení a podobné. Komponenta bude prezentovat síťovou topologii v podobě grafu sítě se zaměřením na rychlé získání přehledu o bezpečnostní situaci s možností interaktivního vyhledávání. Součástí zobrazení každého uzlu grafu sítě bude kompletní menu nabízených operací či informací vztahujících se k danému zařízení s možností vyhledávání. Jednotlivé položky menu budou napojeny na ostatní komponenty vizualizačního dashboardu pro usnadnění získání detailních informací.

Detailní informace o stroji

Cílem vizualizace detailních informací o stroji je uživateli přehledným způsobem prezentovat informace o konkrétním stroji v síti. Komponenta plní případ užití, kterým je rychlé zjištění informací relevantních pro řešení bezpečnostního incidentu. Takové informace obsahují data o operačním systému stroje, aplikací běžících na stroji, služeb poskytovaných strojem a jeho případné zapojení do výpočetního clusteru.

Přehled aktuálních hrozeb a zranitelností

Dalším cílem nástroje je na jednom místě přinést ucelený přehled o aktuálních hrozbách a zranitelnostech v síti, čemuž odpovídá i případ užití, ve kterém komponenta poskytne uživateli přehled zranitelností detekovaných různými systémy a umožní problémy v síti seřadit nebo jinak strukturovat podle závažnosti. Základní dělení bude podle závažnosti samotné zranitelnosti podle CVSS metriky, které dále umožní zpřesnit pomocí důležitosti zranitelného stroje a požadavků na tento stroj.

Přehled síťových útoků

Cílem je také zobrazit aktuální přehled útoků detekovaných v rámci organizace. Tento přehled bude naplňovat následující případy užití:

1. Vizualizace útoků na síť organizace
   Komponenta vytvoří Real-Time Threat Map zachycující aktuální bezpečnostní situaci organizace z pohledu globální sítě Internet. Tím uživateli umožní získat přehled o aktuální bezpečnostní situaci organizace z globálního pohledu, o geografickém původu jednotlivých útočníků a vztazích mezi jednotlivými útoky.
2. Vizualizace útoků na jednotlivá zařízení v síti organizace
   Komponenta zobrazí aktuální bezpečnostní situaci organizace z pohledu jednotlivých koncových zařízení. Tím uživatel získá přehled o typech a množství útoků vedených na jednotlivá zařízení a umožní mu tak odhalit koordinovaný útok proti určitému typu zařízení nebo část organizace. Mapa bude dále také zahrnovat informaci o stejných typech útoků vedených na jednotlivá zařízení a kontakt na správce daného adresního rozsahu.

Mapování kritických závislostí strojů a služeb

Cílem nástroje je rozšířit povědomí uživatele o stavu sítě z hlediska významnosti jednotlivých zařízení a jejich vzájemných závislostí. Z toho vycházejí dva případy užití:

1. Vizualizace závislostí v síti - Komponenta poskytne uživateli přehled o všech detekovaných závislostech konkrétního stroje na dalších strojích nebo službách v síti.
2. Vizualizace kritických serverů organizace - Komponenta umožní zvýraznit stroje v síti na základě jejich vypočítané důležitosti. Toto zvýraznění bude možné dále upravovat pomocí jednotlivých metrik důležitosti, jako je počet poskytovaných služeb, počet uživatelů, množství závislostí a podobně. Dále bude komponenta zobrazovat požadavky organizace na stroj (dostupnost, důvěrnost, integrita).

Zobrazení možných dopadů konkrétního útoku nebo hrozby

Jedním z cílů nástroje je přinést přehled o aktuálních hrozbách a útocích včetně informací o jejich možného dopadu na síť organizace. Tím plní následující případy užití:

1. Vyhodnocení okamžitého stavu a možného postupu útoku - komponenta uživateli zobrazí aktuálně detekované útoky a hrozby v síti, které namapuje na jednotlivé prvky sítě. Tato stavová informace bude dále doplněna výpočtem možného dalšího postupu útočníka nebo další šíření hrozby pomocí promítnutí útočných grafů do topologie sítě.
2. Vyhodnocení rizika útoku - komponenta umožní zvýraznit hrozby, jejichž relativní dopad výrazně převyšuje ostatní aktuální hrozby v síti. Tím uživateli usnadní rozhodování při prioritizaci řešení problémů v síti na základě kvantifikace škod, které může hrozba způsobit.

Vizualizace obranné akce

Cílem nástroje je přinést přehled možných obranných akcí a jejich dopadů na síť organizace. Tím řeší případ užití, kterým je návrh obranné akce a zjištění její vhodnosti - komponenta pro každý probíhající útok zobrazí seznam dostupných obranných akcí. Pro zobrazované akce budou navíc dostupné detaily o jejich dopadu jak na probíhající útok, tak na stav sítě organizace, čímž uživateli usnadní výběr nejvhodnější z nich.

Vizualizace prvků aktivní obrany

Cílem nástroje je zobrazit dostupné prvky aktivní obrany, jejich aktuální konfiguraci a dopady aktivace jejich částí. Tím naplní případy užití:

1. Vizualizace nastavení prvků aktivní obrany - komponenta uživateli zobrazí veškeré prvky aktivní obrany v síti včetně jejich umístění v síti nebo na konkrétních strojích. Po rozkliknutí obranného prvku uživateli zobrazí jeho aktuální nastavení a možnosti konfigurace.
2. Simulace dopadů obranného opatření - komponenta zobrazí možné dopady provedené akce z hlediska dostupnosti jednotlivých služeb nebo prvků sítě. Tím poskytne přehled nejen z hlediska síťování, ale i závislostí mezi jednotlivými stroji v síti a umožní tak komplexně posoudit dopady akce ještě před jejím vykonáním.

Vizualizace měření sítě

Cíle nástroje je vhodně vizualizovat stav měřící infrastruktury, nástrojů a procesů, které jsou používány ke sběru dat pro další analýzu. Toto plní případ užití kontroly aktuálnosti a správnosti informací. Před analýzou dat je často vhodné si ověřit, zda jsou data dostupná, aktuální a správná. K analýze nelze přistoupit pokud data chybí nebo jsou neúplná, což může být způsobeno výpadkem měřících nástrojů. Pokud jsou data dostupná, ale neaktuální, opět například díky nedostupnosti měřících nástrojů, lze při analýze bezpečnostních incidentů udělat chybu. Změny konfigurace či chyby v systémech mohou také vyvolat reakce jako zvýšení objemu poskytovaných dat. V takovém případě může být analytik zahlcen daty a jejich analýza se tak stává obtížnější.

Analýza požadavků

V této sekci jsou uvedeny požadavky na webovou aplikace a jejich stručná analýza. Nejprve jsou uvedeny funkční a nefunkční požadavky. Funkční požadavky vyjadřují to, co by měl být systém schopný vykonat. Nefunkční požadavky popisují omezení kladená na systém, například ohledně výkonu, zabezpečení, či technických parametrů. Dále je popsán datový model, se kterým webová aplikace pracuje.

Funkční požadavky

Základem při tvorbě webového rozhraní jsou následující funkční požadavky, které byly identifikovány při analýze a návrhu systému a které vycházejí z případů užití popsaných výše. Jedná se o:

- správu úloh a workerů,
- získání přehledu o misích organizace,
- aplikace konfigurace pro jednotlivé mise,
- vyhledání síťového uzlu podle IP adresy,
- ověření výskytu zranitelnosti podle CVE identifikátoru.

Na základě těchto požadavků byly navrženy čtyři panely, které jsou blíže představeny dále v textu. Jednotlivé panely obsahují jednu či více komponent, které jednak naplňují funkční požadavky a jednak umožňují realizovat případy užití. Jeden panel či komponenta může realizovat více případů užití.

Nefunkční požadavky

Nefunkční požadavky jsou takové požadavky, které sa netýkají přímo toho, co by měl být systém schopný dělat, ale jde zejména o požadavky na výkon, bezpečnost, dostupnost a různé další technické parametry.

Výsledné webové rozhranie by má podporovat autentizaci pomocí OpenID Connect (OIDC), což je rozšíření protokolu OAuth 2 o autentizaci a API pro získávání informací o uživateli[1]. Využití OIDC umožňuje připojit webovou aplikace k systému jednotného přihlášení organizace, ve které je aplikace nasazena.

[1] https://openid.net/connect/

Keďže webové rozhranie obsahuje viacero grafových vizualizácii, je dôležité zvoliť také vizualizačné knižnice, ktoré budú fungovať plynulo a bez zbytočne dlhého načítavania.

Vzhledem k četnosti vizualizačních komponent a druhů vizualizací je rovněž třeba zvolit vhodné vizualizační knihovny, které budou fungovat plynule a bez zbytečně dlouhého načítání. Vhodné je také zohlednit dostupnost dokumentace a živost projektu, tedy zda je daná knihovna aktivně vyvíjena nebo udržována.

Webové rozhraní má být ve formě SPA (Single Page Application). Pokud to použité technologie umožní, součástí výsledného kódu by měli být i jednotkové testy a dokumentace. Též by měla existovat jednoduchá možnost lokálního spuštění webového rozhraní pomocí technologie Vagrant[2] a Ansible[3], případně spuštění webového rozhraní ve formě Docker kontejneru[4].

Na rozměry obrazovky, na které by mělo být webové rozhraní dostupné, nejsou kladeny žádné specifické požadavky, aplikacia by teda měla být dostupná pro standardní rozlišení používaná na desktopech či laptopech. Podpora mobilních zařízení není vyžadována, naopak je vhodné kalkulovat s možností provozování webové aplikace na větších obrazovkách s velkým rozlišením, například v případě nasazení v bezpečnostních operačních centrech (SOC).

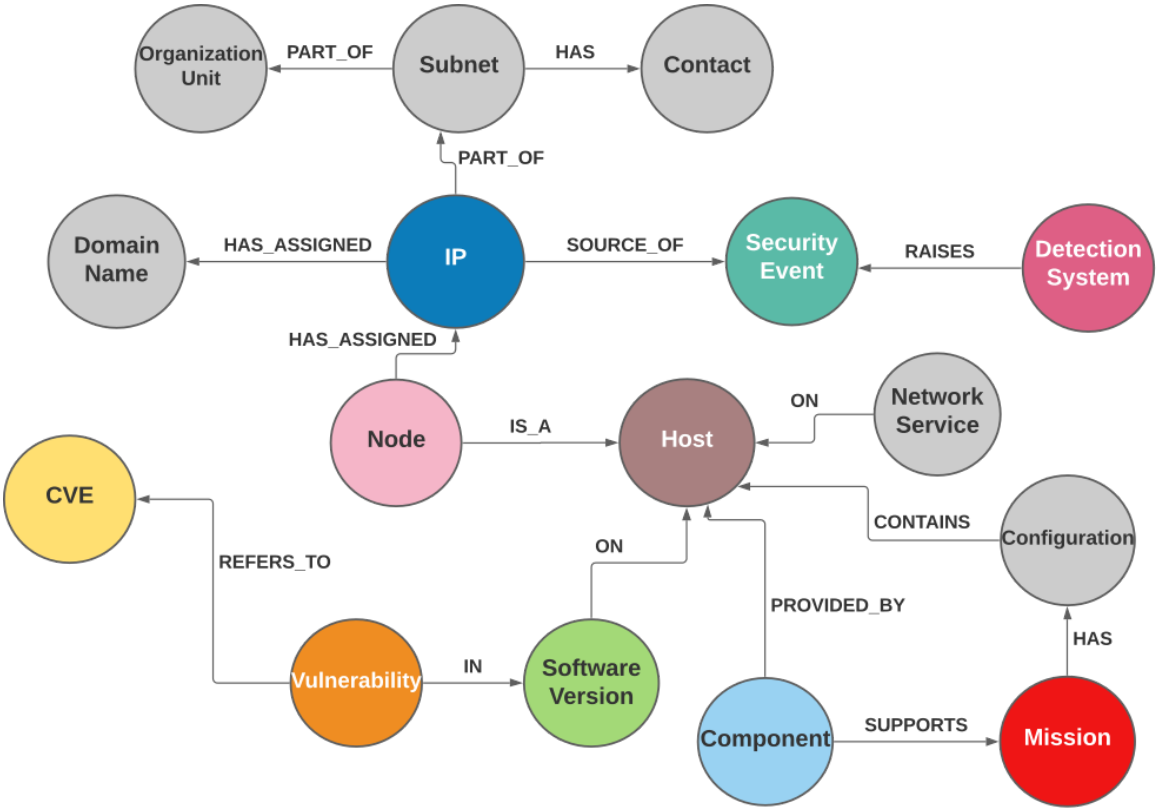
Na použité technologie a programovací jazyky nejsou kladeny další požadavky, pouze je doporučeno držet se rozšířených jazyků a nástrojů a pokud je to možné, využívat podobné technologie jako ostatní softwarové nástroje vytvořené v rámci projektu CRUSOE.

Datový model

Webové rozhraní vizualizuje převážně data, která poskytují další nástroje, které vznikly v rámci projektu CRUSOE. Pro usnadnění spolupráce mezi nástroji a sjednocení formátu pro ukládání dat a manipulaci s nimi byl navržen datový model, který popisuje entity a relace, které se vyskytují v dané doméně, tedy síťové bezpečnosti s ohledem na budování situačního povědomí. Datový model, prezentovaný v odborném článku[5], se drží grafové reprezentace entit a relací, kde entity vystupují jako uzly v grafu a relace jako hrany mezi nimi. Hrany i uzly pak mohou obsahovat řadu dalších informací ve svých parametrech. Pro účely práce s těmito daty je vhodné vypsat seznam entit, se kterými webová aplikace pracuje. Datový model je rozšiřitelný, proto je možné, že bude v budoucnu rozšířen o další entity a relace a bude možné implementovat další vizualizace. Ke dni dokončení vývoje software Orient byl znám následující seznam entit, která bylo možné vizualizovat. Seznam je

[2] https://www.vagrantup.com/
[3] https://www.ansible.com/
[4] https://www.docker.com/
[5] KOMÁRKOVÁ, Jana; HUSÁK, Martin; LAŠTOVIČKA, Martin; TOVARŇÁK, Daniel. CRUSOE: Data Model for Cyber Situational Awareness. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. Association for Computing Machinery, 2018. ARES 2018. ISBN 9781450364485.

rozdělen podle kategorií entit tak, jak jsou popsány v příslušném článku a kompletně je zobrazen též na Obrázku 1.



Obrázek 1: Schéma datového modelu.

Component
Uzel Component představuje službu, která podporuje činnost určité mise.
Atributy:

- name - název komponenty, např. Email blacklist.

Configuration
Uzel představuje konfiguraci dostupnou pro danou misi.
Atributy:

- config_id - identifikátor konfigurace,
- availability - míra dostupnosti mise po aplikování konfigurace,
- confidentiality - míra důvěrnosti mise po aplikování konfigurace,
- integrity - míra integrity mise po aplikování konfigurace,
- time - čas vytvoření konfigurace ve formátu YYYY-MM-DDTHH:MM:SS.

Contact
Uzel představuje kontaktní údaj na osobu resp. tým zodpovědný za danou podsíť. Většinou jde o emailovou adresu.

Atributy:

- name - Emailová adresa resp. jiný dostupný kontakt.

CVE

Uzel představuje CVE kód pro danou zranitelnost a obsahuje atributy zranitelnosti dané metrikami CVSSv2 nebo CVSSv3.

Atributy:

- CVE_id - CVE kód dané zranitelnosti, např. CVE-2014-0160,
- access_vector - uvádí způsob, jakým může být zranitelnost zneužita, např. NETWORK,
- access_complexity - uvádí náročnost zneužití zranitelnosti, např. LOW,
- attack_vector - atribut odpovídá atributu access_vector v CVSSv2,
- attack_complexity - atribut odpovídá atributu access_complexity v CVSSv2,
- authentication - uvádí počet autentizací, které je nutné při útoku vykonat,
- availability_impact_v2 - atribut metriky CVSSv2 uvádí jakým způsobem může být narušena dostupnost služby při útoku,
- availability_impact_v3 - atribut metriky CVSSv3 uvádí jakým způsobem může být narušená dostupnost služby při útoku,
- base_score_v2 - celkové skóre zranitelnosti metriky CVSSv2,
- base_score_v3 - celkové skóre zranitelnosti metriky CVSSv3,
- confidentiality_impact_v2 - atribut metriky CVSSv2 uvádí jakým způsobem může být narušena důvernost služby při útoku,
- confidentiality_impact_v3 - atribut metriky CVSSv3 uvádí jakým způsobem může být narušena důvěrnost služby při útoku,
- description - slovní popis zranitelnosti,
- impact - slovní popis dopadu zranitelnosti,
- integrity_impact_v2 - atribut metriky CVSSv2 uvádí jakým způsobem může být narušena integrita služby při útoku,
- integrity_impact_v3 - atribut metriky CVSSv3 uvádí jakým způsobem může být narušena integrita služby při útoku,
- obtain_all_privilege - atribut uvádí, zda se využitím zranitelnosti dá dopracovat k právům správce systému,
- obtain_user_privilege - atribut uvádí, zda se využitím zranitelnosti dá dopracovat k právům užívatele systému,
- privileges_required - atribut uvádí, jaká práva jsou potřebná pro vykonání útoku,
- published_date - datum zveřejnění zranitelnosti ve formátu YYYY-MM-DDTHH:MMZ.

DetectionSystem

Uzel představuje detekční systém, který vytváří bezpečnostní události dostupné pod uzly SecurityEvent.

Atributy:

- name - název detekčního systému.

DomainName

Uzel představuje doménové jméno priřazené k IP adrese.

Atributy:

- domain_name - název domény,
- tag - typ DNS záznamu, např. A/AAAA.

Host
Uzel představuje zařízení v síti. Uzel nemá žádné atributy.

IP
Uzel představuje IP adresu přiřazenou uzlu v síti. Atributy:
- address - IP adresa zařízení.

Mission
Uzel představuje misi organizace.
Atributy:
- name - název mise,
- description - textový popis mise,
- structure - struktura závislostí mise na komponentách a zařízeních uváděná ve formátu JSON.

NetworkService
Uzel představuje síťovou službu provozovanou na daném zařízení.
Atributy:
- port - port, na kterém je provozována síťová služba,
- protocol - protokol síťové služby, typicky TCP nebo UDP,
- service - název síťové služby.

Node
Uzel představuje uzel v počítačové síti.
Atributy:
- dependency_degree - míra závislosti na jiných síťových zařízeních,
- topology_betweenness - míra udávající polohu zařízení v rámci síťové topologie.

OrganizationUnit
Uzel představuje organizační jednotku, pod kterou spadají určité podsítě.
Atributy:
- name - název organizační jednotky.

SecurityEvent
Uzel představuje bezpečnostní událost, která byla detekovaná detekčním systémem.
Atributy:
- confirmed - atribut udává, zda došlo k potvrzení bezpečnostní události,
- description - textový popis bezpečnostní události,
- detection_time - čas detekce bezpečnostní události,
- type - typ bezpečnostní události.

SoftwareVersion
Uzel představuje software nainstalovaný na daném zařízení.

Atributy:

- tag - označuje program pomocí kterého došlo k detekci softwaru, např. nmap_client,
- version - název a verze detekovaného software.

Subnet
Uzel představuje podsíť.
Atributy:

- note - poznámka, která blíže specifikuje podsíť,
- range - rozsah podsítě zadaný v CIDR notaci.

Vulnerability
Uzel představuje zranitelnost, která je blíže specifikována uzlem CVE.
Atributy:

- description - popis zranitelnosti.

Přehled komponent

Na základě všeobecných funkčních požadavků jsme se rozhodli rozdělit webové rozhraní na čtyři panely, přičemž pro každý panel byl vytvořený návrh, který byl následně konzultovaný s celým řešitelským týmem. V následujících podkapitolách blíže představujeme jednotlivé panely.

Jednotlivé panely budou obsahovat dílčí vizualizace tak, jak byly navrženy v odkazovaných technických zprávách, případně v této technické zprávě na Obrázku 1. Další panely budou obsahovat ovládací prvky dílčích komponent softwarových výstupů projektu. Seznam rozpracovaných panelů je následující:

- Kontrolní panel software pro fázi Orient,
- Panel pro procházení a vyhledávání ve společné databázi včetně detailních informací o hostech v síti,
- Vizualizace síťové topologie včetně výskytu prvků KII a jejich závislostmi,
- Vizualizace výskytu zranitelností,
- Vizualizace historie bezpečnostních incidentů,
- Kontrolní panel software pro fázi Decide včetně vizualizací dopadu útoku a podpory rozhodování,
- Kontrolní panel software pro fázi Act a komponent pro ovládání prvků aktivní obrany.

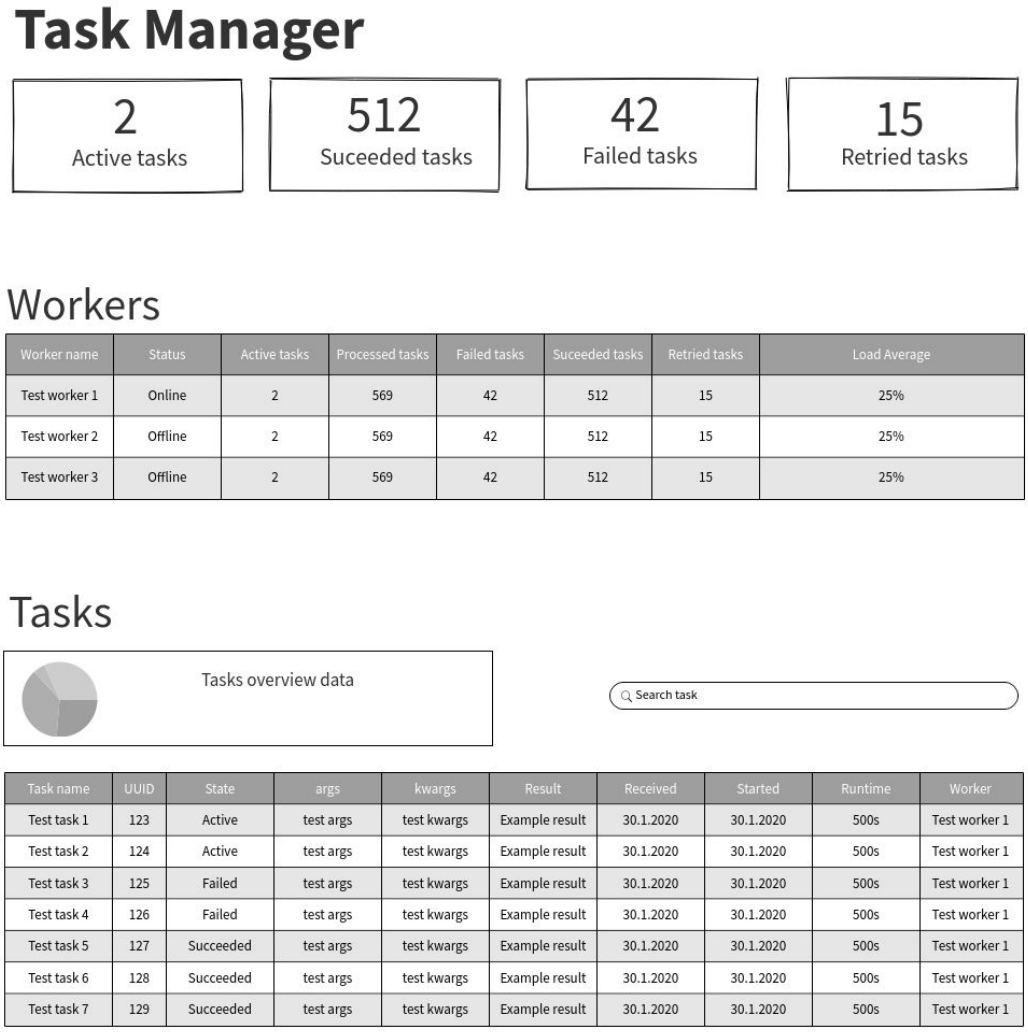
Seznam není konečný, v průběhu finalizace vývoje software a uživatelského testování může dojít k přesunům obsahu mezi různými panely tak, aby vyhovoval běžným scénářům užití software a ulehčil proces rozhodování o bezpečnostních hrozbách a incidentech z pohledu operátora.

Panel Task Manager

Pro získávání dat potřebných pro navyšování situačního povědomí v síti je nutné spouštět poměrně vysoké množství různých úloh. Správu těchto úloh zabezpečuje Orchestrační

služba Výsledku č. 1 Software pro evidenci zranitelností v počítačové síti, konkrétně je pak realizována systémem Celery[6].

Cílem panelu je poskytnout uživateli přehled o těchto úlohách a možnost je spravovat. Uživatel by měl mít možnost vyhledávat úlohu na základě jejích parametrů. Panel by měl rovněž obsahovat přehled o stavu Celery workerů (především jejich vytíženost a počty úloh) a možnost jejich konfigurace. Přínosem by byl také koláčový diagram, který by rozděloval úlohy na základě úspěšnosti jejich dokončení, aby tak bylo uživateli na první pohled jasné, zda nedochází k selhání velkého množství úloh. Diagram by měl být průběžně aktualizovaný bez nutnosti manuálního obnovení stránky v prohlížeči.



Obrázek 2: Wireframe panelu Task Manager.

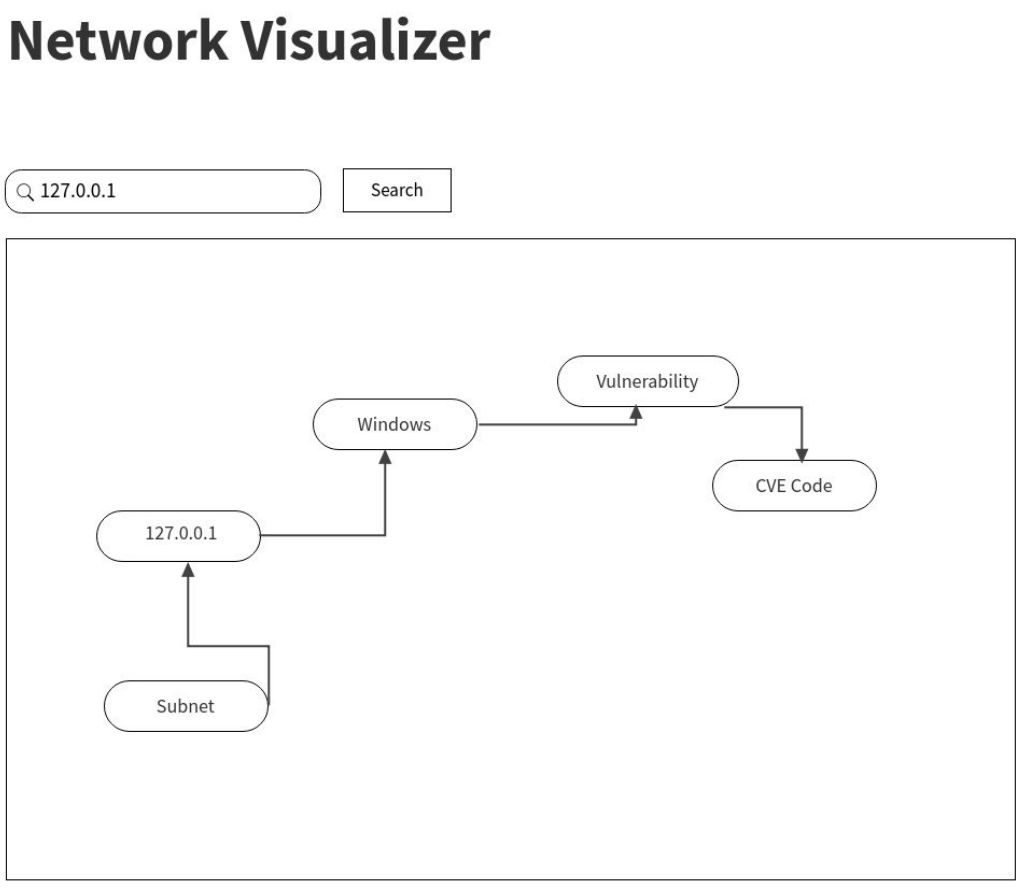
[6] http://celeryproject.org/

Na Obrázku 2 je zobrazený návrh panelu Task Manager. Panel by se měl skládat z následujících částí:

- informace o počtu úloh na základě jejich stavu (aktivní, úspěšně ukončené, neúspěšně ukončené a opakovan spuštěné),
- přehled Celery workerů, jejich stav, počet přidělených úloh a vytíženost,
- koláčový diagram zobrazující úlohy dle jejich stavu,
- přehled úloh seřazený od nejaktuálnější s možností vyhledávání.

Panel Network Visualization

Panel *Network Visualization* by měla poskytovat možnost vyhledat uzel v počítačové síti na základě IP adresy. Následně by měla pro tento uzel zobrazit graf, který bude zobrazovat vlastnosti a entity spojené s tímto uzlem. Uživatel si tak bude moci v rychlosti utvořit přehled o daném zařízení v síti, o software, který je na zařízení naistalovaný, o příslusnosti k podsíti, či zranitelnostech spojených s nainstalovaným software. Informace jsou získávány z *Neo4j* databáze pomocí podpůrných nástrojů Výsledku č. 1 projektu CRUSOE.



Obrázek 3: Wireframe panelu Network Visualizer.

Pro daný uzel v síti by měly být dostupné následující informace:

- příslušnost k podsíti,
- přiřazené doménová jména,

- nainstalovaný software,
- zranitelnosti nacházející se v nainstalovaném software.

Další přínosem je možnost uživatele rozkliknout daný uzel, přičemž se po rozkliknutí zobrazí jeho další relevantní okolí. Návrh panelu Network Visualization je zobrazen na Obrázku 3.

Panel Vulnerability

Panel *Vulnerability* by měla podporovat možnost vyhledat zranitelnost na základě jejího CVE (Common Vulnerabilities and Exposures) identifikátoru. Data potřebná pro tento panel jsou dostupná v centrální Neo4j databázi podobně jako v případě panelu *Network Visualization*. Panel by měl zobrazovat popis zranitelnosti, koláčový diagram zobrazující její výskyt v podsítích a tabulku obsahující počítače v síti, na nichž se nachází software obsahující tuto zranitelnost. Kliknutím na některý z počítačů v tabulce dojde k přesměrování uživatele na panel *Network Visualization*, na které se ve formě grafu zobrazí vybraný počítač a jeho okolí.

Hlavním cílem panelu je tedy umožnit uživateli ověřit přítomnost dané zranitelnosti ve spravované síti a v případě jejího výskytu by si měl být uživatel schopen rychle utvořit obraz o rozsahu rozšíření této zranitelnosti a její závažnosti.

Na Obrázku 4 je zobrazený návrh panelu *Vulnerability*.

Vulnerability

CVE-123-123 Search

Vulnerability description

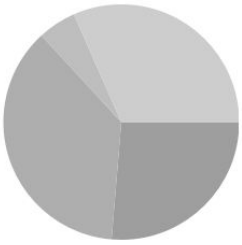
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat

Sample parameter: Sample value
Sample parameter: Sample value
Sample parameter: Sample value

Sample parameter: Sample value
Sample parameter: Sample value
Sample parameter: Sample value

Show more info

Vulnerability occurences in subnets



List of affected hosts

IP Address	Domain Name	Subnet	Software
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10
127.0.0.1	localhost	127.0.0.1	Windows 10

Obrázek 4: Wireframe panelu Vulnerability.

Panel Decide/Act

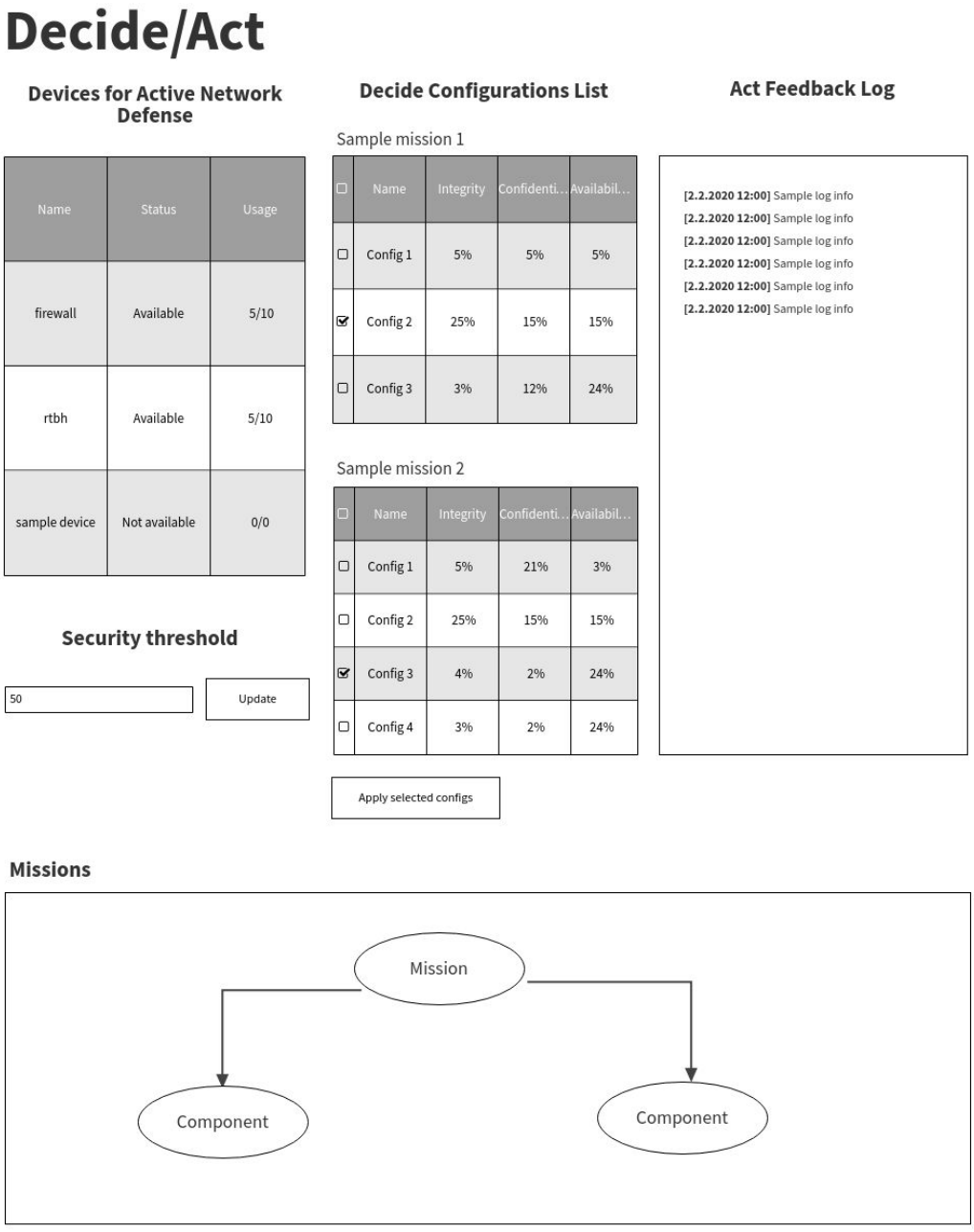
Panel *Decide/Act* by měla zobrazovat informace získané z webových API nástrojů fází Decide a Act, tedy Výsledků č. 3 a 4 projektu. Jde především o informace o misích a prvcích aktivní obrany (dále PAO). Uživatel by měl mít možnost zjistit stav PAO, jejich vytíženost, IP adresu a port, na kterých jsou dostupné.

Pro jednotlivé mise by měl být dostupný jejich popis a výčet konfigurací. Taktéž by měla existovat možnost pro každou misi aplikovat některou z jejích konfigurací. Každá konfigurace je definována třemi atributy - důvěrnost, integrita a dostupnost. Na panelu by se měl nacházet log, který bude uživateli poskytovat zpětnou vazbu o tom, zda došlo k úspěšné aplikaci konfigurace, případně jej informovat o možných selháních. Aplikace konfigurace a získání zpětné vazby bude probíhat pomocí webových API PAO.

Přínosem pro panel by bylo zobrazení grafu, který by vizualizoval dostupnost jednotlivých misí. Jednotlivé mise sou zabezpečované informačními službami, které je možné dále dekomponovat na podpůrné komponenty. Propojení těchto uzlů je vyjádřeno prostřednictvím *AND/OR* uzlů, které určují konjunktivní resp. disjunktní závislost na potomcích.

Vztahy mezi těmito entitami a dostupnost podpůrných komponent bude poskytovat webové API Výsledku č. 3. Na základě dostupnosti podpůrných komponent a vztahů panelu *Decide/Act* vygeneruje graf, který červenou barvou zvýrazní všechny nedostupné mise, informační služby a podpůrné komponenty. Uživatel tak bude moct rychle zjistit, které mise jsou nedostupné a co tuto nedostupnost způsobuje.

Na Obrázku 5 je zobrazený návrh panelu *Decide/Act*.



Obrázek 5: Wireframe panelu Decide/Act.

Návod k instalaci

Webovou aplikaci pro vizualizaci bezpečnostní situace v počítačové síti lze spustit více různými způsoby. Pro testování či vývoj doporučujeme použít spuštění prostřednictvím nástrojů Docker nebo Vagrant. Postupy spuštění jsou obsahem prvních dvou sekcí návodu k instalaci. Návod k instalaci na lokální systém je popsán v následující sekci a je doporučen pro uživatele, kteří si přejí aplikaci nainstalovat a provozovat permanentně, například v produkčním prostředí.

Instalace a spuštění pomocí nástroje Docker

Instalace a spuštění webové aplikace pomocí nástroje Docker spočívá ve vytvoření Docker image a jeho spuštění v kontejneru.

Nejprve je třeba nainstalovat Docker, doporučujeme postupovat podle návodu na stránce:

https://docs.docker.com/get-docker/

Docker je dostupný pro všechny hlavní operační systémy.

Před vytvořením Docker image je třeba se ujistit, že soubor *src/environments/environment.prod.ts* obsahuje správná URL k API.

Obsah souboru *src/environments/environment.prod.ts* vypadá následovně:

```
1. import * as packageData from '../../package.json';
2.
3. export const environment = {
4.   applicationName: 'CRUSOE Dashboard',
5.   production: true,
6.   version: packageData.version,
7.   /* Redirect API URL */
8.   apiUrl: 'https://crusoe.muni.cz/redirect-api/redirect/',
9.   /* Act API URL */
10.  tmpActApi: 'https://crusoe.muni.cz/act/',
11.  /* GraphQL API URL */
12.  graphqlApi: 'https://crusoe.muni.cz/graphql-api/graphql',
13.  /* Firewall API URL */
14.  firewallApi: 'https://crusoe.muni.cz/firewall',
15.};
```

Výše uvedená ukázka předpokládá, že data jsou dostupná prostřednictvím více různých API na stroji *crusoe.muni.cz*. Jednotlivá API jsou nastavena na řádcích 8, 10, 12 a 14.

Docker image je vytvořen příkazem:

```
$ docker build --no-cache -t crusoe_dashboard .
```

Dále je třeba vytvořený Docker image spustit v kontejneru příkazem:

```
$ docker run --name crusoe_dashboard -d -p 4200:80 crusoe_dashboard
```

K webové aplikaci je možné přistupovat na adrese:

http://localhost:4200/

Instalace a spuštění pomocí nástroje Vagrant

Instalace a spuštění webové aplikace pomocí nástroje Vagrant spočívá ve vytvoření virtuálního stroje a nainstalování webové aplikace na daný virtuální stroj.

Nejprve je třeba stáhnout a nainstalovat nástroj Vagrant, pokud již na stroji není nainstalován. Doporučujeme postupovat podle návodů uvedených webu projektu. Instalační soubory pro všechny hlavní operační systémy jsou dostupné na adrese:

https://www.vagrantup.com/downloads

Dále je třeba nainstalovat nástroj Ansible, pokud již není nainstalovaný. Opět doporučujeme postupovat podle návodu na webu vývojářů dostupného na adrese:

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-the-control-machine

Před spuštěním je třeba zkontrolovat, zda jsou endpointy API nastaveny správně. Je třeba zkontrolovat obsah souboru *ansible/playbook.yml*, zda obsahuje správná URL. Obsah souboru *ansible/playbook.yml* může vypadat následovně:

```
1. ---
2. - hosts: all
3.   vars:
4.     # Dashboard folder
5.     dashboard_folder: /home/vagrant/dashboard
6.
7.     # Redirect API URL
8.    redirectApi: 'https://crusoe.csirt.muni.cz/redirect-api/redirect/'
9.     # Act API URL
10.    actApi: 'https://crusoe.csirt.muni.cz/act/'
11.    # GraphQL API URL
12.    graphqlApi: 'https://crusoe.csirt.muni.cz/graphql-api/graphql'
13.    # Firewall API URL
14.    firewallApi: 'https://crusoe-worker.csirt.muni.cz/firewall'
15.  # become: true
16.  roles:
17.    - dashboard
```

Nástroj Vagrant může následně spustit a nastavit virtuální stroj s webovou aplikací příkazem:

```
$ vagrant up
```

Po sestavení a konfigurace virtuálního stroje, která může trvat i několik minut, je webová aplikace dostupná na adrese:

http://localhost:4200/

Lokální instalace

Prerekvizitou je nainstalování balíku Node.js minimálně ve verzi 10. Doporučujeme získat Node.js ze stránek vývojářů a postupovat podle návodu na adrese:

https://nodejs.org/en/download/

Ověření nainstalované verze Node.js lze provést zadáním příkazu:

```
$ node -v
```

Dalším krokem je instalace balíčků *npm*, které obsahují závislosti webové aplikace. Správce balíčků *npm* je zpravidla nainstalován spolu s *Node.js*. Seznam závislostí webové aplikace je uveden v souboru package.json v kořenovém adresáři projektu. K jejich instalaci stačí v kořenovém adresáři spustit příkaz:

```
$ npm install
```

Podobně jako u ostatních způsobů instalace je potřeba ověřit správnost nastavení endpointů pro API. Nastavení se nachází v souboru *src/environments/environment.prod.ts*, případně v souboru *src/environments/environment.ts*, pokud je projekt spravován nástrojem *ng serve*. Typicky však platí první možnost, která je určená pro produkční nasazení, druhá je relevantní pro vývoj a testovací nasazení. V tomto souboru je třeba nastavit URL různých API. Ukázka souboru je uvedena v návodu pro spuštění pomocí nástroje Docker, jedná se o stejný soubor.

Vývojový server, který slouží k vývoji a testování, je možné spustit pomocí příkazu:

```
$ ng serve
```

Poté stačí přistoupit na adresu:

http://localhost:4200/

Na adrese bude dostupná webová aplikace, která automaticky reaguje na změny ve zdrojových kódech a znovu se načítá po každé změně.

Pro sestavení projektu je třeba použít příkaz:

```
$ ng build
```

Spuštěním příkazu bude projekt sestaven a uložen v adresáři dist.

Produkční verzi webové aplikace lze sestavit příkazem:

```
$ ng build -- prod
```

Sestavenou produkční verzi ze následně z adresáře dist přesunout do adresáře, ze kterého má být aplikace načítána pro produkční prostředí, například adresáře webového serveru. Nasazení produkční verze tak může vypadat následovně:

```
$ rm -r /var/www/dashboard/*
$ cp -r ./dist/ /var/www/dashboard/
$ sudo /etc/init.d/apache2 restart
```

Nejprve je smazán aktuální obsah adresáře webového serveru (například pokud tam je dřívější verze webové aplikace), následně jsou soubory zkopírovány do adresáře a nakonec je webový server (v tomto případě Apache 2) restartován. Webová aplikace je potom dostupná na adrese z nastavení webového serveru.

Uživatelská dokumentace

Uživatelská dokumentace webové aplikace pro vizualizaci bezpečnostní situace v počítačové síti sestává z pěti částí. Nejprve je popsán přístup k aplikaci a navigace v ní, tedy proces přihlášení a správy uživatelů a navigace mezi panely v aplikace. V dalších čtyřech částech jsou popsány návody pro používání jednotlivých panelů s vizualizacemi. Prvním panelem je Task Manager sloužící ke správě zdrojů dat pro vizualizace. Druhým panelem je Network Visualizer, ve kterém je možné procházet nasbíraná data, zejména dohledávat kontextové informace k entitám v síti. Třetím je panel Vulnerabilities, ve kterém je vizualizován výskyt zranitelností v síti včetně detailních informací. Posledním panelem je panel Decide/Act, který obsahuje vizualizace a kontrolní prvky nástrojů pro podporu rozhodování při řešení bezpečnostního incidentu a aplikaci reaktivních opatření na prvcích aktivní obrany počítačové sítě.

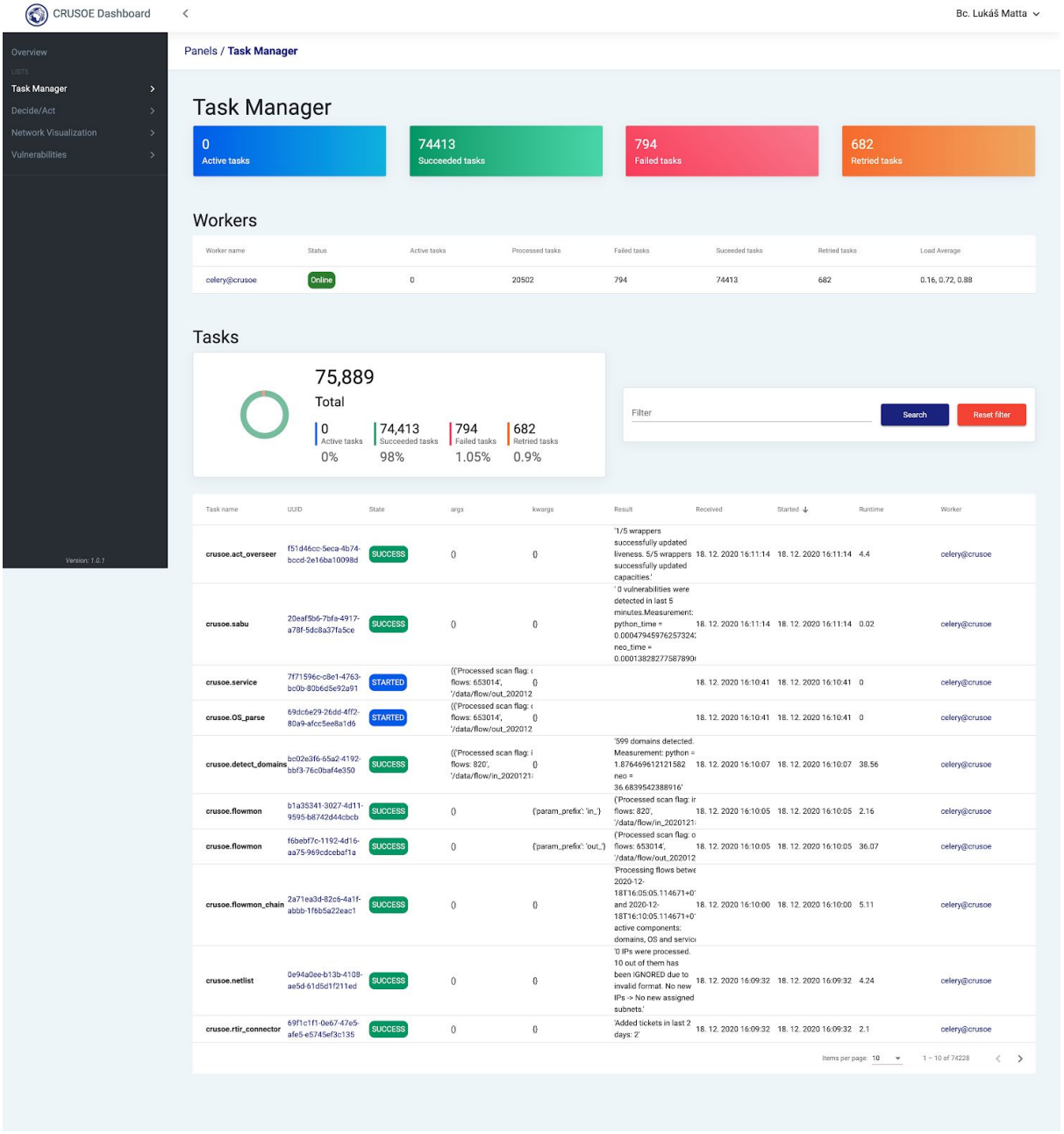
Přihlášení a navigace

Po otevření adresy webové aplikace se zobrazí úvodní obrazovka zachycená na snímku na Obrázku 6. Obrazovka obsahuje tlačítko Login, které přesměruje uživatele na přihlašovací obrazovku. V případě využití OIDC autentizace je uživatel přesměrován na přihlašovací obrazovku příslušné správy identit. Po přihlášení je uživatel přesměrován zpět na webovou aplikaci, kde se mu zobrazí úvodní obrazovka se seznamem panelů v menu na levé části panelu a plocha s dlaždicemi odkazujícími na jednotlivé panely ve zbytku obrazovky. Po kliknutí na odkaz na panel v menu nebo na dlaždici se namísto dlaždic zobrazí daný panel. Menu vlevo je zachována pro navigaci do dalších panelů. V pravém horním rohu se nachází jméno právě přihlášeného uživatele. Po kliknutí na jméno se zobrazí informace o uživateli. V levém horním rohu je tlačítko s logem, které nasměruje uživatele na úvodní obrazovku.

Obrázek 6: Snímek přihlašovací obrazovky.

Panel Task Manager

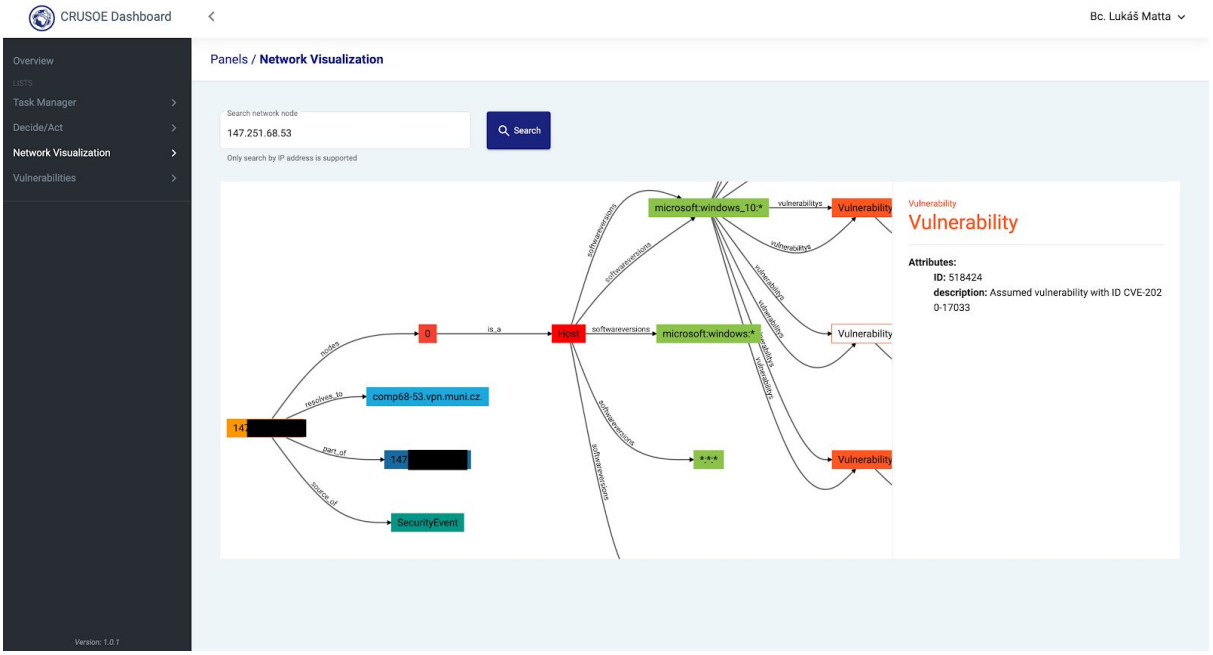
Panel Task manager je zobrazen na Obrázku 7 a slouží ke správě úloh sběru dat. Panel je napojen na systémy dohledu nad komponentami a nástroji pro sběr dat o síti. Panel se skládá ze čtyř částí umístěných pod sebou. V horní části je v barevných obdélnících vizualizován počet aktivních, úspěšně dokončených, neúspěšně dokončených a opakovaných úloh. Díky tomu má uživatel rychlý přehled o tom, jaká je aktuální zátěž systému a zda systémy pracují správně. Níže se nachází tabulka Workers se seznamem workerů, nástrojů, které tyto úlohy provádějí. Pod tabulkou je v sekci Tasks uveden graf počtu úloh (detailnější vizualizace dat z horní části obrazovky). Zbytek panelu tvoří tabulka se seznamem úloh, ve které je možné procházet záznamy o úlohách, filtrovat je a vyhledávat v nich. Po kliknutí na některý záznam se otevře stránka s detailními informacemi.



Obrázek 7: Snímek obrazovky s panelem Task Manager.

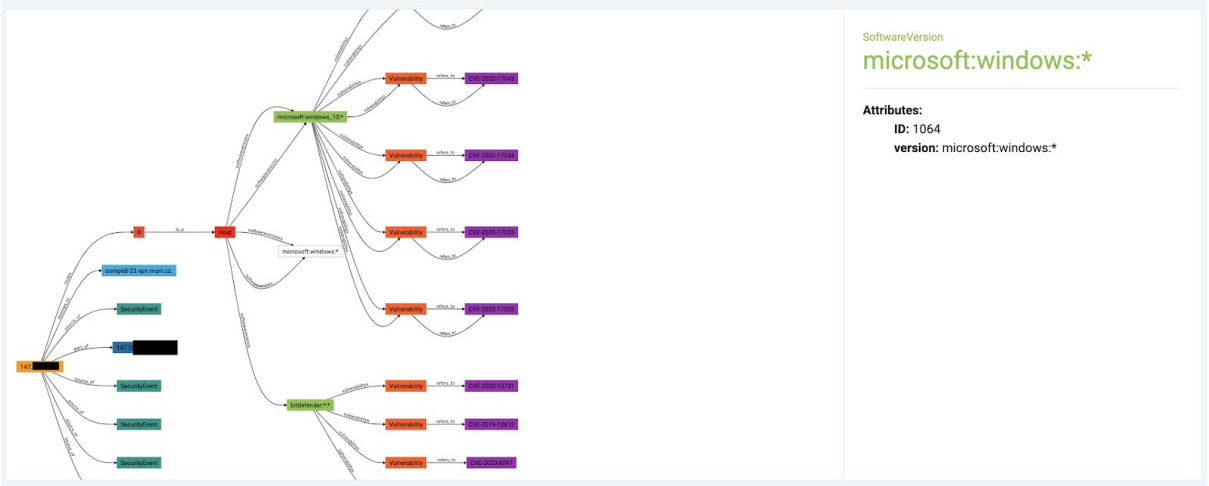
Panel Network Visualization

Panel Network VIsualization, který je k nahlédnutí na Obrázku 8, slouží k vizualizaci obsahu databáze a procházení grafem, kterým jsou data modelována. Entity jsou modelovány jako vrcholy grafu a relace jako hrany mezi nimi. Vzhledem k velkému množství dat, která jsou v databázi často umístěna, není vhodné zobrazovat je všechny naráz. Nejčastějším případem užití tohoto panelu je vyhledání kontextových informací k entitě v síti, nejčastěji IP adrese. Proto je v horní části panelu vyhledávací pole pro zadání IP adresy. Po zadání IP adresy a stisknutí tlačítka Search nebo potvrzení klávesou Enter je v databázi vyhledána patřičná IP adresa a pokud je nalezena, poslouží jako výchozí bod pro vizualizace.



Obrázek 8: Snímek obrazovky s panelem Network Visualization.

Vizualizace zabírá zbytek panelu. V bílém poli je zobrazen graf, jehož centrálním prvkem je vyhledávaná IP adresa. Vrcholy grafu jsou zobrazeny jako barevné obdélníky vyplněné barvou odpovídající vrstvě datového modelu, ve které se entita nachází, a názvem entity. Hrany mezi vrcholy jsou vizualizovány černými šipkami s popiskem odpovídajícím názvu hrany. Vyhledaná IP adresa je zobrazena vlevo uprostřed pole pro vizualizaci. Dále jsou z databáze vybrány všechny entity, které mají hranu spojující je s danou IP adresou. Ty jsou vykresleny napravo od IP adresy. Po kliknutí na některý vrchol grafu se v pravé části vizualizační oblasti otevře panel, na kterém jsou vypsány všechny informace, které jsou v databázi uloženy jako parametry daného vrcholu. Grafem je možné dále procházet a přistupovat k dalším sousedícím objektům. Dvojklikem na některý vrchol grafu se do vizualizovaného grafu přidají všechny vrcholy, které mají společnou hranu s rozkliknutým vrcholem, a hrany mezi nimi. Opakováním tohoto úkonu je možné procházet grafem a dohledávat další a další související informace. Detail takového procházení grafem je zobrazen na Obrázku 9. Díky procházení a postupnému načítání je omezena velikost zobrazovaných dat, která se soustřeďují jen na potenciálně relevantní informace. Další data si uživatel načte podle potřeby.

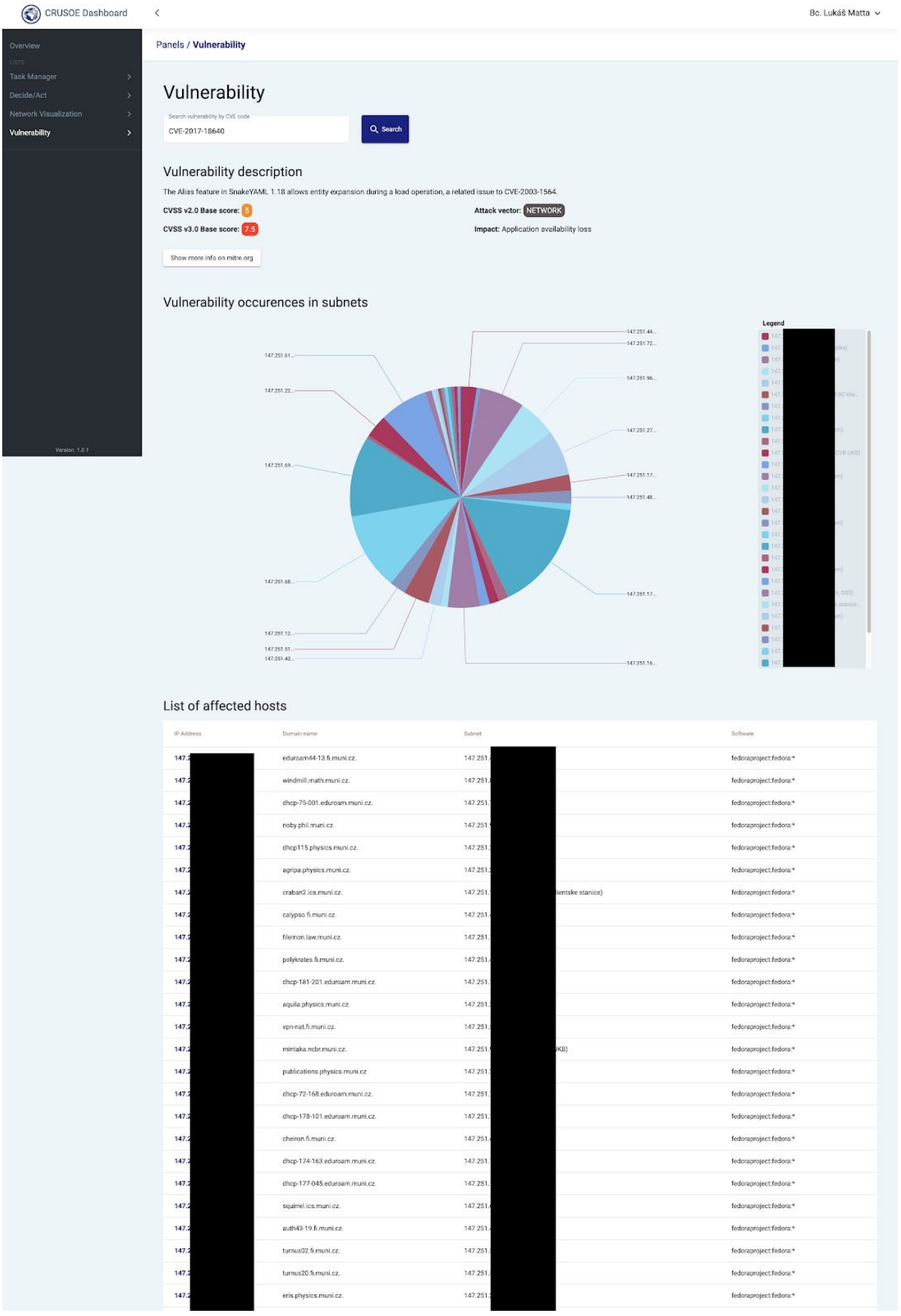


Obrázek 9: Detail procházení grafu v panelu Network Visualization.

Typicky uživatel vyhledá IP adresu, která je momentálně ohrožena v rámci probíhajícího bezpečnostního incidentu. Díky hlášení detekčních nástrojů například víme, že útočník dané zařízení intenzivně skenuje. Vyhledáním IP adresy uživatel zjistí, o jaké zařízení se jedná, například že jde o součást kritické infrastruktury v některé z podsítí. Procházením grafu si uživatel otevře informace o tom, jaký software na daném zařízení běží, například verzi operačního systému. Rozkliknutím verze operačního systému se pak uživatel dozví, jaké jsou pro tuto verzi známé zranitelnosti. Podle závažnosti zranitelností a důležitosti zařízení se pak uživatel může rozhodnout pro patřičnou akci v reakci na probíhající incident.

Panel Vulnerabilities

Panel Vulnerabilities slouží k vyhledávání a a vizualizaci informací o výskytu zranitelností v chráněné síti. Panel umožňuje uživateli vyhledat konkrétní zranitelnost podle jejího identifikátoru CVE a pokud je taková zranitelnost nalezena, jsou zobrazeny informace, které jsou k této zranitelnosti a k jejímu výskytu v síti dostupné v databázi. Panel se skládá z několika částí, které jsou popsány dále. Celý panel je zobrazen na Obrázku 10.



Obrázek 10: Snímek obrazovky s panelem Vulnerabilities.

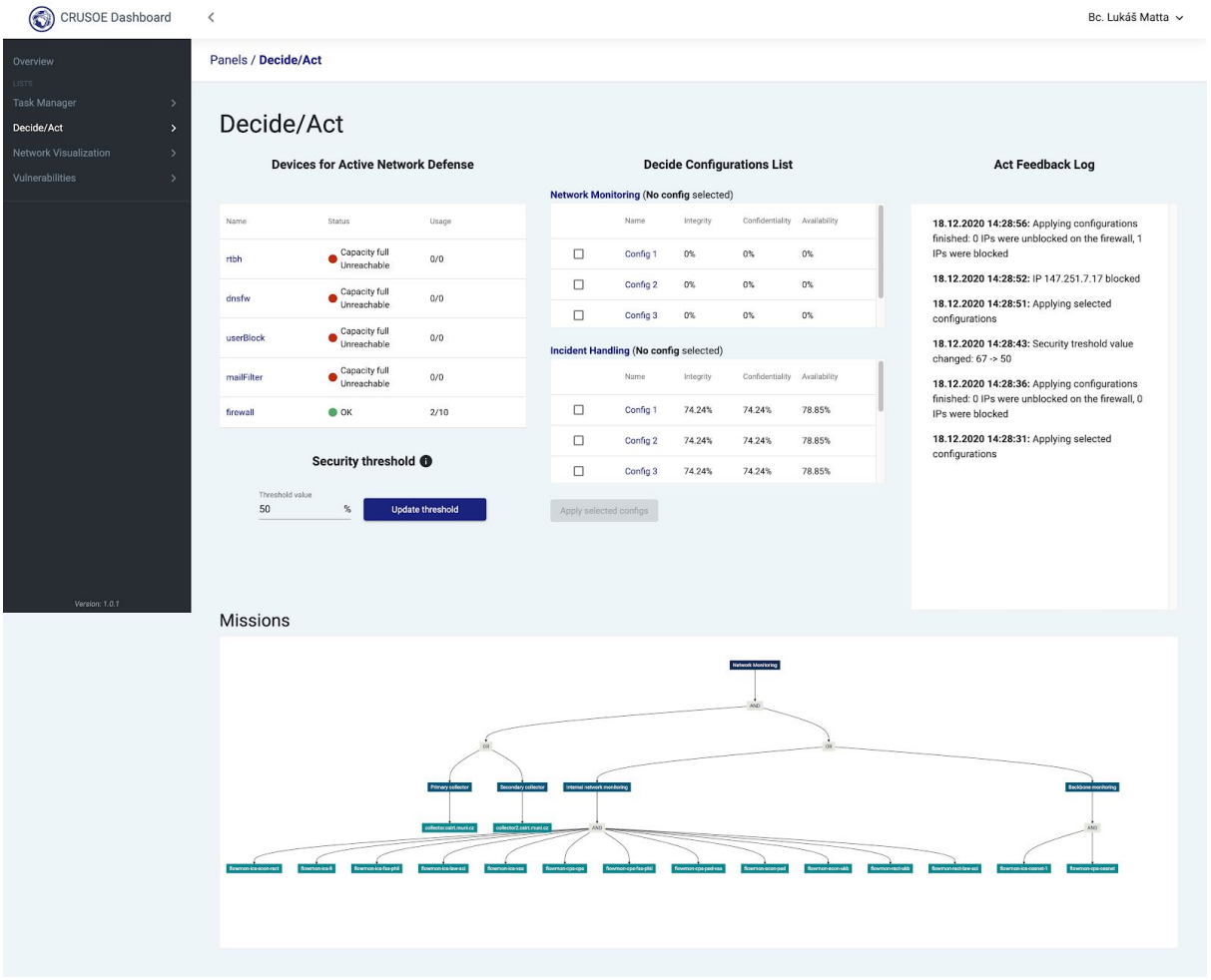
Výchozím bodem pro uživatele je pole pro vyhledávání zranitelnost umístěné v horní části panelu. Bílé pole je opatřeno nápovědou s očekávaným formátem vstupu, tedy CVE identifikátoru zranitelnosti. Uživatel zadá CVE identifikátor zranitelnosti a stisknutím tlačítka Search nechá panel vykreslit dostupné informace k dané zranitelnosti.

Hned pod vyhledávacím polem jsou zobrazeny vybrané informace o nalezené zranitelnosti, zejména její CVSS skóre a vektor útoku, který zranitelnost využívá. Tyto informace jsou získané z databáze NVD. Pod informace je umístěno tlačítko sloužící k otevření stránky věnované dané zranitelnosti na webu cve.mitre.org, kde jsou uvedeny všechny podrobné informace o dané zranitelnosti. Pokud má uživatel zájem o tyto informace, kliknutím na tlačítko se mu otevře patřičná webová stránka. Informace využívané při práci s nástroji projektu CRUSOE jsou však uvedeny rovnou v panelu.

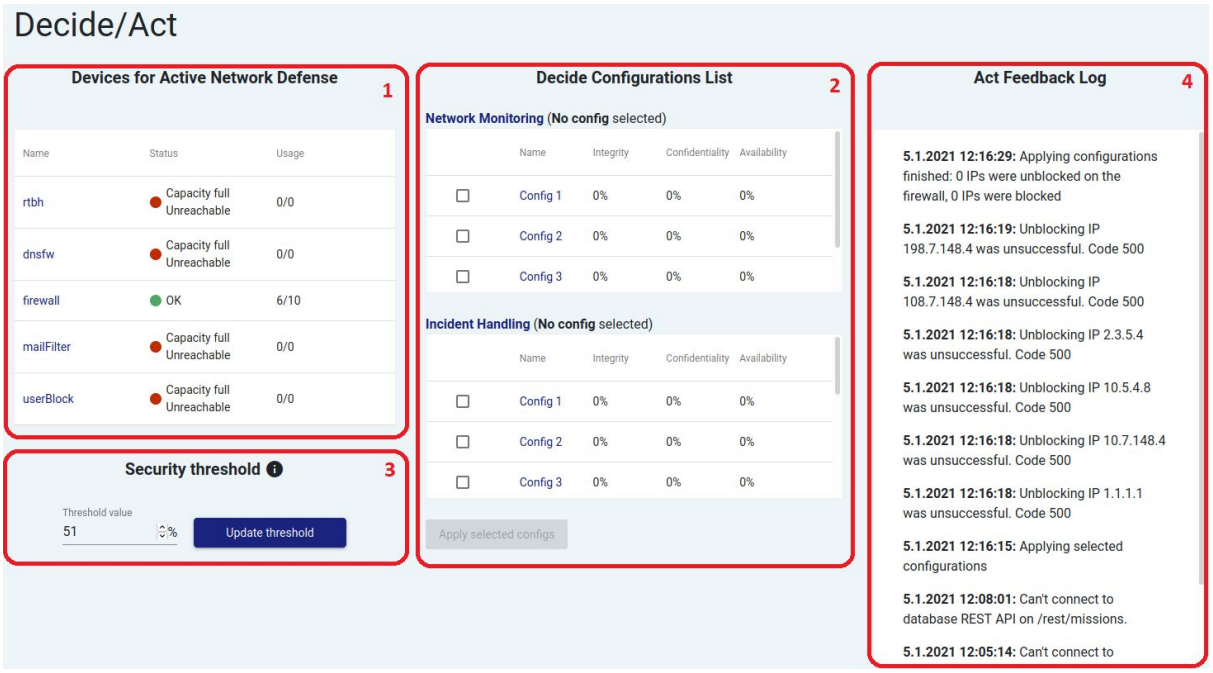
Pod informacemi o zranitelnosti se nachází první vizualizace, koláčový graf výskytu zranitelnosti v jednotlivých podsítích chráněné sítě. Koláčový graf zobrazuje počty zařízení, na kterých se vyhledaná zranitelnost nachází nebo pravděpodobně může nacházet. Každá výseč grafu reprezentuje jednu podsíť chráněné sítě. Koláčový graf je vybaven popisky pro snadnější orientaci. Legenda s celými názvy podsítí a uvedeným počtem zranitelných strojů se nachází vpravo od koláčového grafu.
Posledním prvkem panelu je tabulka v jeho dolní části. V tabulce je vypsán seznam všech zranitelných zařízení, o kterých je v databázi veden záznam. V tabulce je pro každý záznam o zranitelném zařízení uvedena IP adresa a doménové jména daného zařízení, příslušnost k podsíti a identifikátor software, který danou zranitelnost na zařízení detekoval.

Panel Decide/Act

Všechny informativní a ovládací prvky software fáze Act jsou shromážděny na jediné stránce dashboardu. Protože je software fáze Act těsně provázán se software fáze Decide, sdílí spolu i tuto stránku. Tato sekce popisuje všechny prvky, kterými je možné z dashboardu software fáze Act ovládat. Rozložení prvků na stránce je znázorněno na Obrázku 11.



Obrázek 11: Snímek obrazovky s panelem Decide/Act.



Obrázek 12: Stránka software fáze Act v dashboardu.

Obrázek 12 zvýrazňuje rozdělení panelů podle jejich funkce.

1. Seznam prvků aktivní obrany
2. Seznam misí a konfigurací definovaných v Decide
3. Aktuální hodnota Security threshold
4. Seznam aktuálních událostí ve fázi Act

Seznam prvků aktivní obrany

Seznam obsahuje soupis všech prvků aktivní obrany, jejichž připojení software podporuje. Slouží k získání rychlého přehledu o stavu těchto prvků. Panel seznamu je zobrazen na Obrázku 13. Obsahuje zleva vždy název prvku, souhrnné informace o jeho stavu, a kapacitu. Po kliknutí na název prvku se zobrazí jeho technické informace - IP adresa a port, na kterém běží přístupové rozhraní prvku.

Obrázek 13: Seznam prvků aktivní obrany se jedním dostupným prvkem.

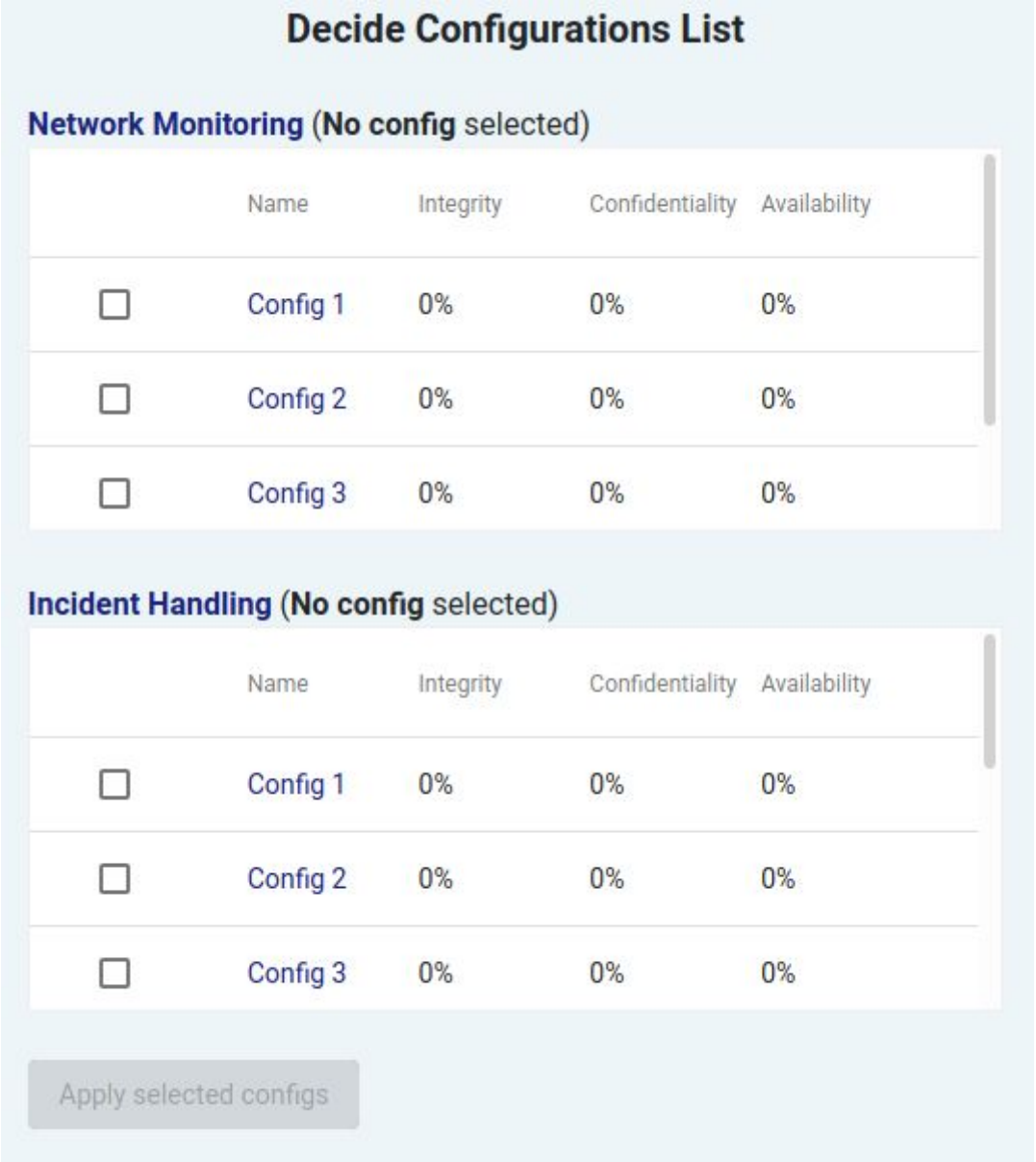
Pro každý podporovaný prvek zobrazuje seznam semaforové hodnocení jeho stavu, které se odvíjí od jeho aktuální kapacity a kontaktnosti. Prvek může nabývat tří stavů (zelená, žlutá, červená) a to za následujících podmínek:

- Zelená - vše v normě.
    - OK - Nesplňena žádná z podmínek níže.
- Žlutá - prvek funguje s výhradami.
    - Capacity 90 % full - Zjištěno naplnění maximální kapacity z 90 %.

    - Last liveness check failed - Poslední kontakt se povedl před více než 10 minutami.
- Červená - prvek má problém, který je nutné vyřešit, jinak není schopen dál operovat.
    - Capacity full - Využitá kapacita prvku je rovna maximální kapacitě.
    - Unreachable - Poslední kontakt se povedl před více než 30 minutami.

Seznam misí a konfigurací

Seznam misí a konfigurací obsahuje soupis všech misí, se kterými pracuje software fáze Decide. Ke každé misi je dále uveden soupis možných konfigurací. Podrobnější popis misí a konfigurací je obsažen v dokumentaci fáze Decide. Panel seznamu misí a konfigurací je zobrazen na Obrázku 14.



Obrázek 14: Seznam misí a konfigurací se dvěma misemi, Network Monitoring a Incident Handling.

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.

Aktuální hodnota Security threshold

Security threshold udává hraniční hodnotu pro blokování zařízení. Pokud je tato hranice pro zařízení překročena, je zařízení zablokováno, ale pouze v případě, že nepatří do konfigurace, kterou si vybral uživatel. Nastavení hodnoty Security threshold je zobrazeno na Obrázku 15.

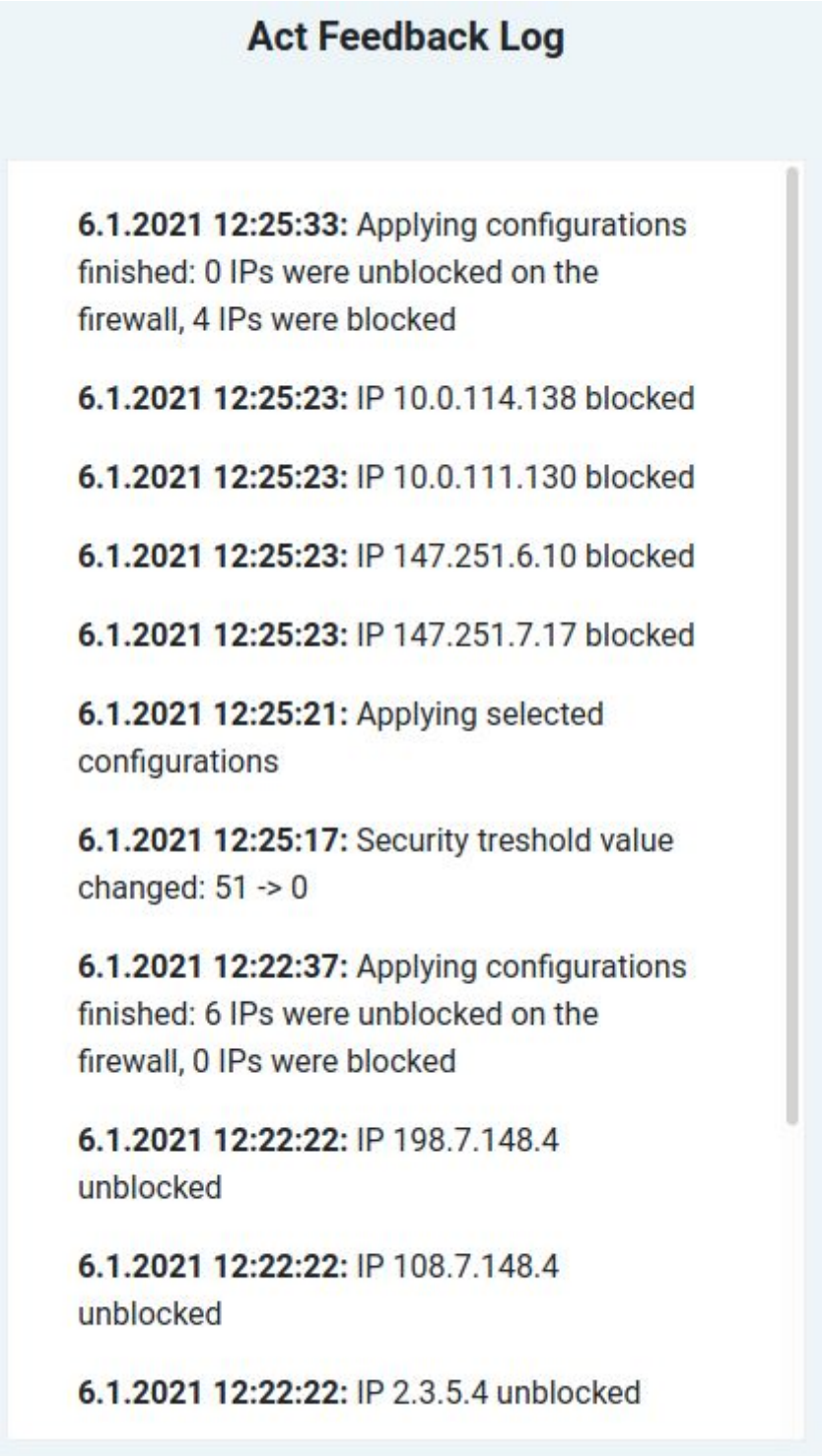
Obrázek 15: Panel nastavení Security threshold.

Situaci můžeme ilustrovat na dvou příkladech. Mise má tři konfigurace. První dvě z nich mají nulové hodnoty pro všechny bezpečnostní parametry a třetí 75% pro všechny bezpečnostní parametry. Security threshold je nastavený na 50. Uživetel si vybere první nulovou konfiguraci. V tomto případě kvůli nulovému hodnocení druhé konfigurace nebudou zablokovány ani zařízení patřící do druhé konfigurace, protože její nulové ohodnocení nepřekročilo threshold. To proto, bezdůvodně neblokovali zařízení, na kterých jsme neodhalili žádné bezpečnostní nedostatky. Zařízení, která v třetí konfiguraci překročili threshold, budou zablokována.

Druhý ilustrační případ se týká mise s třemi konfiguracemi. Nechť konfigurace 1 má nulové hodnocení pro všechny bezpečnostní parametry, konfigurace 2 má 55% pro všechny bezpečnostní parametry a konfigurace 3 má 68% pro důvěrnost, integritu i dostupnost. V tomto případě se administrátor na základě svého expertního názoru rozhodne vybrat si konfiguraci 2. Pro threshold 50% nebude žádné ze zařízení konfigurace 2 zablokované a to i navzdory tomu, že překročilo threshold. Konfigurace byla totiž vybrána uživatelem. Rovněž nebudou zablokována ani zařízení s nulovým hodnocením (konfigurace 1). U ostatních zařízení, která překročila threshold, dojde k blokování.

Seznam aktuálních událostí ve fázi Act

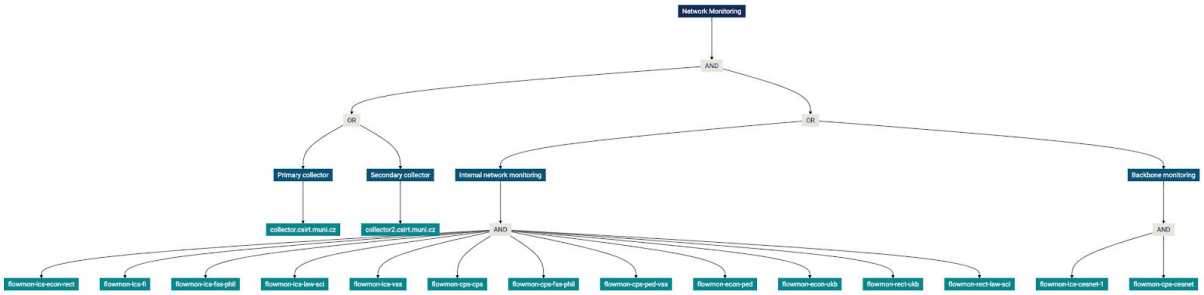
Seznam aktuálních událostí ve fázi Act zobrazuje důležité události, ke kterým dochází v reálném čase. Zobrazovány jsou zejména průběh a výsledky manuálně zadaných operací, a případně chyby či varování, která tyto akce vyvolaly. Příklad obsahu panelu při pokusu o aplikaci konfigurací je zobrazen na Obrázku 16.



Obrázek 16: Seznam aktuálních událostí. Na obrázku je zachycena (zdola nahoru) aplikace konfigurací, změna Security threshold, a opětovná aplikace konfigurací s rozdílným výsledkem.

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 17.



Obrázek 17: Vizualizace popisu mise.

Programátorská dokumentace

Programátorská dokumentace sestává ze dvou částí. Nejprve jsou popsány použité technologie, u kterých jsou popsány principy jejich užití. Následuje sekce věnovaná popisu architektury systému. Následně je popsána implementace systému, která obsahuje popis sdílených komponent a služeb a detaily k implementaci jednotlivých panelů. Samostatně jsou uvedeny poznámky k testování systému a kontinuální integrace.

Použité technologie

Tato sekce obsahuje stručný popis frameworku Angular spolu s dalšími technologiemi souvisejícími s vývojem Angular aplikací, které byly použity pro implementaci vizualizačního nástroje. Styl vizualizačních komponent je založen na frameworku Bootstrap s přidáním metodologie Block-Element-Modifier (BEM), která pomáhá dosáhnout opakovaně použitelného kódu CSS.

Angular

Angular[7] je JavaScript framework založený na TypeScript, který poskytuje nástroje pro vytváření vysoce interaktivních webových a mobilních webových aplikací. Jedná se o open-source software vyvinutý a udržovaný společností Google. Vizualizační plugin popsaný v této zprávě je napsán v Angularu verze 7.

Angular je framework založený na komponentách, který poskytuje deklarativní šablony nabízející jednoduchý způsob, jak vytvořit vysoce flexibilní a opakovaně použitelné komponenty HTML + CSS. Kromě toho kombinuje různé koncepty programování (např. dependency injection, two-way data binding, zero-step offline installation), které vedou k čistšímu kódu a rychlému vývoji napříč platformami. Protože rychlost i výkon jsou důležité pro vývoj webových aplikací, framework také integruje několik funkcí (např. automatic code-splitting, lazy loading), které umožňují rozdělit celý balík aplikací do několika sekcí a načíst je pouze na vyžádání.

Protože Angular je úspěšný softwarový nástroj, je podporován velkou komunitou, která pravidelně rozšiřuje jeho funkce. Framework také poskytuje rozsáhlou dokumentaci, včetně tutoriálů s praktickými příklady, které výrazně zjednodušují přechod k této technologii.

TypeScript

Aplikace Angularu jsou psány v TypeScriptu[8] - statisticky kompilovaném jazyce vyvinutém společností Microsoft s plně otevřeným zdrojovým kódem. Typescript sdílí syntaxi i sémantiku s JavaScriptem a navíc přidává několik konceptů z objektově orientovaných jazyků (např. statické otypování, rozhraní, třídy, a moduly).

[7] https://angular.io
[8] https://www.typescriptlang.org

TypeScript se zaměřuje především na řešení problémů souvisejících s vývojem komplexních aplikací v JavaScriptu při zachování všech výhod prostředí JavaScript. Například zavedení tříd s veřejnými a soukromými členy pomáhá udržovat čistou strukturu a zvyšuje spolehlivost kódu. Statické otypování a abstraktní rozhraní se zabývají nedostatkem dobře definovaných hranic v jednoduchém JavaScriptu, které často vedou k příliš těsnému propojení. Kromě toho poskytuje TypeScript také užitečné nástroje, jako je navigace, pokročilé automatické dokončování a refactoring.

Block-Element-Modifier (BEM)

Aby se zabránilo nedostatku standardů v rozhraní front-end, které často vedou k jen obtížně spravovatelným třídám CSS, implementace se řídí konvencí pojmenování BEM[9]. Tato metodika pomáhá redukovat dědičnost CSS a konflikty stylů a zvyšuje modularitu a opětovnou použitelnost kódu. BEM je zkratka pro Block, Element a Modifier, která popisuje tři základní stavební prvky.

Klíčovou myšlenkou je rozdělit webový obsah na logicky a funkčně nezávislé komponenty - bloky (např. menu, logo, vyhledávání, přihlašovací šablona) a reprezentovat každou s konkrétním atributem třídy. Bloky slouží jako obaly pro prvky - části bloků existující pouze v kontextu jejich nadřazené položky (položka nabídky v nabídce). Nakonec modifikátory fungují jako příznaky, které mění chování nebo styl konkrétního prvku nebo bloku (aktivní položka nabídky oproti pasivnímu).

Bootstrap

Návrh vizualizací je vytvořen pomocí frameworku Bootstrap[10]. Populární open-source front-end knihovna používaná primárně pro vytváření vizuálně atraktivních webů a aplikací. Původně byla vyvinuta jako interní nástroj na Twitteru a teprve později byla zveřejněna pro volné použití. Bootstrap nabízí mnoho snadno použitelných funkcí pro rychlé vytváření prototypů, včetně předběžných sestavování, interaktivních součástí, nástrojů pro rozvržení, doplňků mixins a jQuery. Podporuje také všechny rozšířené prohlížeče.

Vizualizační knihovny

Angular Material UI[11] je open-source komponentová knihovna, která je vyvíjená komunitou ve spolupráci s Angular týmem. Jednotlivé komponenty implementují specifikaci definovanou spoločností Google[12]. Důraz je kladen na jejich použitelnost, přičemž každá komponenta musí obsahovat i definice, které umožňují její použití ve speciálních prohlížečích určených pro slabozraké či nevidomé. Komponenty též zachovávají jednotný design a ovládací prvky. V době zveřejnění software Orient byla poslední stabilní verzí Angular Material UI verze 11.0.2, která obsahuje 33 různých komponent.

[9] http://getbem.com
[10] https://getbootstrap.com
[11] https://material.angular.io/
[12] https://material.io/components

Ngx-charts je open-source knihovna poskytující několik různých typů grafů. Hlavním vývojářem je americká společnost Swimlane. V době zveřejnění software Orient byla poslední stabilní verzí verze 16.0.0. Knihovna je postavena na známé JavaScriptové vizualizační knihovně D3.js[13] a jejím cílem je ulehčit tvorbu grafových vizualizací v prostředí Angular. V Orient z této knihovny vychází koláčový graf.

Ngx-graph je open-source knihovna postavená na knihovně ngx-charts a umožňující tvorbu grafů, které obsahují vrcholy a hrany. V době zveřejnění software Orient byla poslední stabilní verzí verze 7.1.2. V Orient je tato knihovna využívána pro vizualizaci síťových uzlů a elementů s nimi spojených.

REST

REST (REpresentational State Transfer) je architektonický styl rozhraní webových služeb představený v roce 200 a od té doby široce používaný. Je důležité si uvědomit, že se nejedná o jasně definovanou specifikaci, ale o styl. Aplikačné programové rozhranie (API), které sa řídí stylem REST nazývame REST API.

Klíčovým prvkem v REST stylu je tzv. zdroj (resource), s kterým se pomocí webové služby pracuje. Na práci s tímto zdrojem je možné využít různé metody protokolu HTTP (GET, POST, PUT, DELETE atd.). Každý zdroj je definovaný pomocí URI (Uniform Resource Identifier). Přístup REST je vhodný v případech, kdy je možné jednoduše definovat zdroje. Jeho použití je preferované zejména v souvislosti s relačními databázemi, dá se však použít i jinde.

OpenID Connect autentizace

Pro účely autentizace je v Orient využívám standard OpenID Connect. OpenID Connect (OIDC) je rozšířením autorizačního protokolu OAuth 2.0 o autentizační vrstvu. OIDC podporuje několik různých autentizačních schémat, v Orient je využito schéma *Implicit Flow*.

Princip fungování schématu *Implicit Flow* lze popsat ve třech krocích:

- V prvním kroku je uživatel přesměrován na stránku jednotného přihlášení, kterou provozuje správca OIDC v organizace. Provozovatel je také označován jako *OIDC provider*. Na této stránce uživatel vyplní svoje přihlašovací údaje.
- Po vyplnění přihlašovacích údajů a úspěšném přihlášení na stránce provozovatete OIDC je uživatel následně přesměrován na adresu webové aplikace, přičemž v parametru *GET* je odeslán přístupový token typu *Bearer*.
- Webová aplikace zpracuje tento přístupový token a uloží ho do lokální paměti prohlížeče. Webové rozhraní následně používá tento token při autentizaci vůči koncovým bodům, které podporují OIDC autentizaci.

V rámci Orient je využívána open-source knihovna podporující OIDC nazývaná angular-oauth2-oidc[14].

[13] https://d3js.org/
[14] https://github.com/manfredsteyer/angular-oauth2-oidc

Popis implementace

Popis implementace sestává ze sedmi částí. Nejprve je shrnuta výsledná architektura systému včetně jeho rozdělení na komponenty. Následuje Popis implementace sdílených komponent a služeb jako je přihlašování pomocí OIDC a navigace mezi panely. Samostatně je popsáno využití frameworku Angular, na kterém je celá aplikace postavena. Zbývající sekce popisují čtyř panelů vizualizací.

Architektura systému

Systém se skládá z několika částí, hlavní částí je Angular aplikace, ostatními částmi jsou různá aplikační rozhraní. Architekturu tvoří tyto komponenty:

- Angular application - Angular aplikace, hlavní část systému.
- Redirect API - API vzniklo jako mezičlánek pro usnadnění komunikaci Angular aplikace s ostatními nástroji implementovanými v rámci projektu. Je zabezpečené OIDC autentizací. Protože některá API jsou chráněná autentizaci *Basic HTTP*, bylo nutné zřídit toto API, na kterém je možné bezpečně uložit přihlašovací údaje.
- neo4j-rest - API vzniklo pro přístup k datům z Neo4j databáze, částečné splňuje architektonický styl REST.
- Act API - API nástrojů implementujících v projektu CRUSOE fázi Act, slouží pro aplikaci konfigurací a získávání informací o prvcích aktivní obrany.
- Flower API - API poskytuje informace o probíhajících úlohách a běžících workerech systému Celery.
- Neo4j database - grafová databáze Neo4j, Angular aplikace z ní získává data prostřednictvím různých API.

Hlavní částí práce je SPA aplikace vytvořená pomocí frameworku Angular ve verzi 11.0.5, poslední stabilní verzi Angular frameworku dostupnou v době vydání software Orient. Interní označení nástroje je kromě *CRUSOE Orient* i *CRUSOE Dashboard*. Kód aplikacie je složený z těchto programovacích jazyků: TypeScript 57,9 %, HTML 28,9 %, SCSS 11,6 %, JavaScript 0,9 % a CSS 0,4 %. Iniciální kostra aplikace byla vytvořena pomocí nástroje Angular CLI.

Angular aplikace se typicky skládá z různých modulů, komponent a služeb. Zjednodušená souborová struktura aplikace vypadá následovně:

```
├── authentication
   ├── components
   └── services
├── panels
   ├── decide-act
   ├── my-account
   ├── network-visualization
   ├──  panel-overview
   ├── task-manager
   └── vulnerability
```

```
└── shared
    ├── components.
    ├── config.
    ├── interceptors.
    ├── models.
    └──  services.
```

V adresáři *authentication* se nacházejí komponenty související s autentizací (přihlašovací obrazovka, tlačítko pro odhlášení, uživatelský profil) a služba, která zabezpečuje autentizaci pomocí knihovny angular-oauth2-oidc.

V adresáři *panels* jsou komponenty a služby rozdělené podle příslušnosti k jednotlivým panelů,, na které je možné přistupovat prostřednictvím navigace nacházející sa v levé části obrazovky.

Adresář *shared* obsahuje kód, který je využívaný napříč celou aplikací, jde zejména o komponenty jako je horní panel obsahujíci název aplikacie a jméno přihlášeného užívatele, navigace mezi panely a podobně. V adresáři se též nacházejí konfigurační soubory, interceptor (Interceptor je třída, která implementuje rozhraní HttpInterceptor a zabezpečuje vkládání autorizačních hlaviček do HTTP požadavků), modely a služby.

Sdílené komponenty a služby

Následující komponenty a služby jsou užívány napříč komponentami aplikace. Jedná se o:

- Komponent **Navbar** tvoří modrý pruh nacházející se na v horní části obrazovky. Obsahuje název aplikace, jméno právě přihlášeného uživatele, tlačítko pro odhlášení a tlačítko pro zobrazení profilu uživatele.
- Komponent **Sidebar** se nachází v levé části obrazovky a obsahuje menu, prostřednictvím kterého sa může uživatel navigovat mezi jednotlivými panely.
- Služba **AuthService** poskytuje metody určené pro ověření toho, zda je uživatel autentizovaný, a získání údajů o uživateli prostřednictvím OIDC. Služba využívá knihovnu angular-oauth2-oidc.
- Služba **ApiService** poskytuje veřejné metody umožňující komunikaci s API. Aplikace získává a odesílá data zejména použitím metod této služby.
- Služba **ApiInterceptor** implementuje rozhraní HttpInterceptor. Tato služba umožňuje automatické vkládání autorizačního tokenu do hlavičky každého odoslaného požadavku. Není proto potřeba vkládat hlavičku samostatně při každém odeslání požadavku.

Využití frameworku Angular

Angular je framework pro tvorbu webových aplikácii typu SPA (Single Page Application). SPA je druh webové aplikace, která interaguje s uživatelem tak, že dynamicky načítá a přepisuje elementy na aktuální webové stránce. Nedochází tak k načítání celé stránky tak jako u standardních webových aplikací, u kterých se při změně stránky načítá celý její kód. Po iniciálním načtení SPA je dostupný všechen kód potřebný pro samotný chod webové aplikace. Při interakci s uživatelem dochází k získávání potřebných dat z různých API a

jejich následné zobrazení je implementováno pomocí jazyka JavaScript. Výhodou je rychlejší přechod mezi podstránkami a přiblížení se k chování nativní aplikace.

Vývoj frameworku Angular vede tým, který je součástí společnosti Google, do vývoje se však může zapojit i veřejnost, protože jde o open-source framework. Zdrojový kód je dostupný z veřejného repozitáře[15]. Framework využívá programovací jazyk Typescript od společnosti Microsoft. Zdrojové soubory napsané v jazyce Typescript se před spuštěním ve webovém prohlížeči transpilují do jazyka JavaScript, pričemž při této transpilaci probíhá typová kontrola. Programátor tak získává výhody typové kontroly i pro JavaScript.

Framework Angular, označovaný též Angular 2+ nebo Angular v2 a vyšší je následníkem frameworku AngularJS známým též jako Angular 1. V době vydání nástroje Orient byla aktuální stabilní verzí frameworku verze 11.0.3.

Zdrojový kód webové aplikace vytvořené pomocí frameworku Angular sa typicky skladá z třech hlavních součástí: modulů, komponent a služeb. Součástí Angularu je systém vkládání závislostí (dependency injection system), které usnadňuje užívání služeb. Používání těchto součástí je nyní blíže popsáno.

Moduly

Moduly jsou hlavním stavebním prvkem Angular aplikace, nazývame je také *NgModuly*. Predpona *Ng* je zkratka označující Angular, používá se často ve spojení s knihovnami pro Angular. Modul vytváříme jako novou třídu, kterou označujeme dekorátorem *@NgModule*. Konvenci je, aby třída měla v názvu příponu *Module*. Do dekorátoru *@NgModule* vložíme metadatový objekt s následujícími atributy:

- declarations - list, kterým definujeme komponenty, direktivy a roury (pipes) dostupné v modulu,
- imports - list externích modulů, které importujeme do tohoto modulu,
- providers - list dostupných služeb v daném modulu, slouží pro systém vkládání závislostí,
- bootstrap - atribut využívame výhradně v koreňovém modulu a definujeme v něm hlavní komponentu, která se při inicializaci načítá do souboru *index.html*.

Definice třídy s dekorátorem pro modul Angular aplikace tedy může vypadat následovně:

```
1 @NgModule ({
2 // modul obsahuje komponentu ChartComponent
3 declarations : [ ChartComponent ],
4 // modul importuje Angular modul pro praci s formulari
5 imports : [ FormsModule ],
6 // modul exportuje komponentu ChartComponent
7 exports : [ ChartComponent ],
8 // modul poskytuje tridu DataService
9 providers : [DataService],
```

[15] https://github.com/angular/angular

```
10 // prazdne, protoze nejde o korenovy modul
11 bootstrap : []
12 })
13 export class ExampleModule {}
```

Moduly rozdělujeme podle způsobu jejich využití do několika kategorií: doménové moduly, routed moduly, routing moduly, servisní moduly, widget moduly a sdílené moduly. Moduly pomáhají organizovat a dekomponovat kód, také mohou sloužit pro jednodušší sdílení funkcionality napříč různými Angular aplikacemi.

Komponenty

Každá Angular aplikace obsahuje alespoň jednu kořenovou komponentu, ze které se odvíjí hierarchie komponent na dokumentový objektový model (DOM). Typicky se Angular aplikace skládá z několika komponent, které vzájemně komunikují. Takováto dekompozice nám umožňuje znovupoužití komponent na různých místech aplikace či dokonca přesun komponenty do jiné Angular aplikace.

Komponentu vytvoříme jako novou třídu, konvencí je aby tato třída měla v názvu příponu *Component*. Před definici této třídy přidáme dekorátor *@Component*. Dekorátor *@Component* vyžaduje jako vstup metadatový objekt s následujícími atributy:

- selector - řetězec pomocí kterého bude možné přidat komponentu do šablony,
- templateUrl - cesta k *.html* souboru, který představuje šablonu komponenty,
- providers - list dostupných služeb pro danou komponentu, slouží pro systém vkládání závislostí.

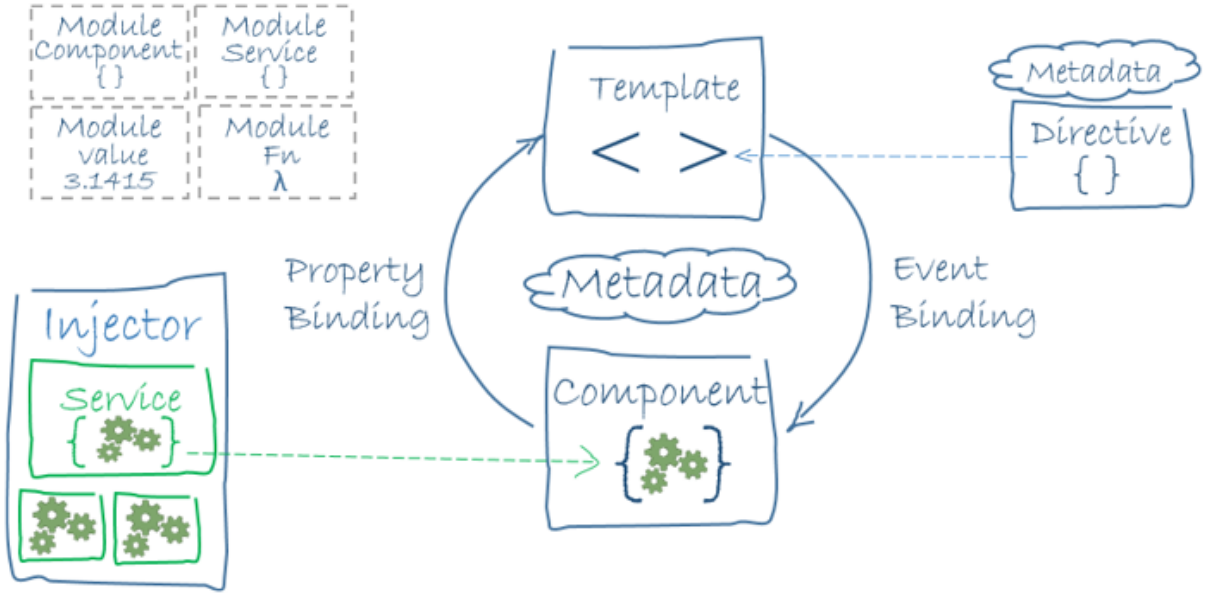
V šabloně je možné přistupovat k metodám a atributům definovaným v komponentové třídě. Tímto způsobem oddělujeme logiku komponenty od toho, co uživatel vidí (šablona).

Angular vytváří, aktualizuje a ničí komponenty na základě aktivity uživatele. Příklad definice jednoduché komponenty vypadá následovně:

```
1  @Component ({
2  // komponenta bude dostupna jako <app-people></app-people>
3  selector : "app-people",
4  // relativni cesta k sablone
5  templateUrl : "people.component.html",
6  // komponenta poskytuje tridu PeopleService
7  providers : [PeopleService],
8  })
9  export class PeopleComponent implements OnInit {
10   people: Person[];
11   selectedPerson: Person;
12
13   constructor(private peopleService: PeopleService) { }
14
15   // Tato metoda je volana Angularom při vzniku komponentu
16   ngOnInit() {
```

```
            this.people = this.peopleService.getPeople();
17    }
18    selectPerson(p: Person) { this.selectedPerson = p; }
19 }
```

Angular podporuje tzv. obousměrné vázání dat (two-way data binding). V praxi to znamená, že pokud změníme hodnotu v třídě komponenty, změna se projaví v šabloně a je automaticky viditelná i pro uživatele. Naopak, pokud uživatel změní hodnotu v šabloně, například napsáním textu do pole formuláře, změna hodnoty v šabloně se přenese do třídy komponenty. Na Obrázku 18 je ilustrované obousměrné vázání dat mezi šablonou a třídou komponenty, ve směru do šablony je označené jako *property binding* a ve směru do komponentové třídy jako *event binding*.



Obrázek 18: Diagram propojení a komunikace jednotlivých částí Angular aplikace.

Služby a vkládání závislostí

Služba v Angular aplikaci je třída, která má jasně vymezený účel. Pomáha tak zjednodušovat komponentové třídy, které by měli obsahovat jen logiku bezprostředně spojenou s danou komponentou. Služby jsou často sdílené mezi různými komponentami, zabezpečují komunikaci s externími API, úpravu dat a další úlohy.

Při komunikaci s externími zdroji často využíváme službu *HttpClient* poskytovanou samotným Angularem. Služba umožňuje odesílat HTTP požadavky pomocí kterých odesíláme či přijímáme data. Data následně zpracováváme s použitím knihovny RxJS[16], která používá návrhový vzor *Observer*.

Název třídy služby by měl mít příponu *Service*, před definicí třídy přidáváme dekorátor *@Injectable*. Nepovinným parametrem tohoto dekorátoru je metadatový objekt s atributem

[16] https://rxjs-dev.firebaseapp.com/guide/overview

providedIn, kterým určíme, v jakém rozsahu bude služba přístupná prostřednictvím systému vkládání závislostí. Atribut může nabývat následujících hodnot:

- root - služba je přístupná napříč aplikací jako singleton (návrhový vzor, ve kterém je v celém programu dostupná jediná instance třídy),
- platform - služba je přístupná napříč více Angular aplikacemi definovanými v jednom webovém rozhraní,
- any - pro každý modul načítaný systémem *lazy load* (systém, při kterém jsou moduly načítané až po tom, co uživatel navštíví URI, které je vyžaduje) je přístupná unikátní instance, pro ostatní moduly je sdílený jeden *singleton*.

Pokud chceme, aby byla dostupná unikátní instance třídy služby pro každou komponentu či modul, přidáme třídu do seznamu *providers*, který byl zmíněn při popisu dekorátorů *@NgModule* a *@Component*.

Třída, která bude dostupná pro celou aplikaci jako *singleton* je definována následovně:

```
1  @Injectable({
2  // Instance tridy bude dostupna pres root dependency injector
3  providedIn: 'root'
4  })
5  class TaxiService {
6  //   Zkraceny   zapis   konstruktoru   v   TypeScriptu,   tzv.
   constructor shorthand
7    constructor(private service: TelephoneService) {}
8  }
```

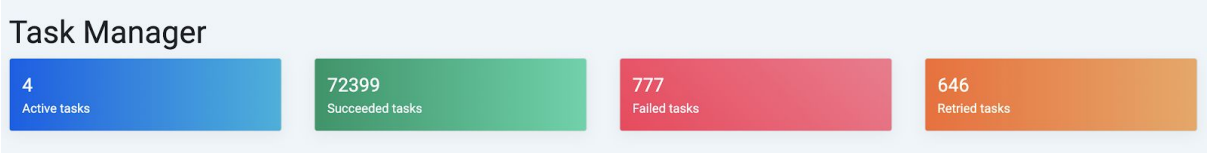
Panel Task Manager

Panel Task Manager slouží ke správě úloh a workerů systému Celery. Využívá API poskytované webovým nástrojem na monitorování a správu Celery s názvem Flower. Pro komunikáciu s Flower API byla vytvořena samostatná třída služby v adresáři *task-manager*. Samotné Flower API ve své původní verzi neumožňovalo jednoduché získávání úloh. Protože jde o open-source projekt, byl kontaktován autor a potřebná funkcionalit a byla přidána přímo do Flower API[17]. Změna byla schválena autorem Flower API a přidána do produkční verze 0.9.7. Komunikace s Flower API probíhá prostředníctvím Redirect API, které zabezpečuje autentizovaný přístup.

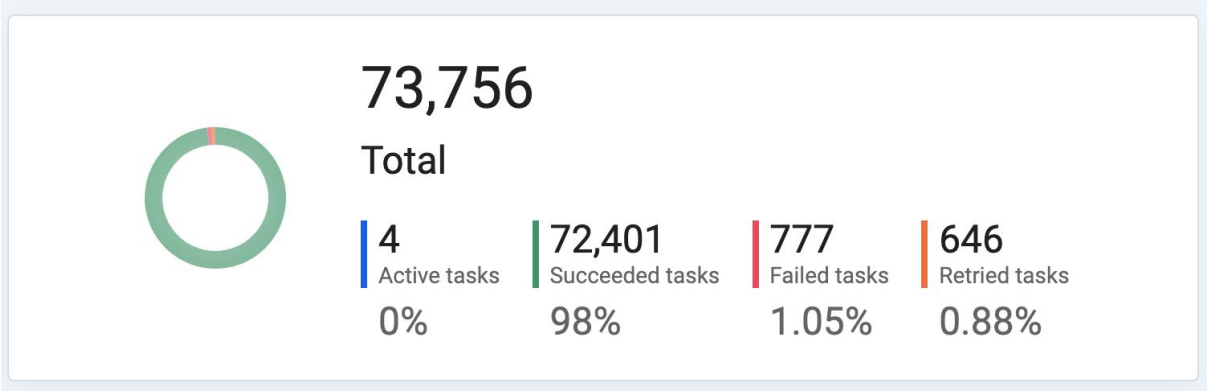
Panel Task Manager obsahuje dvě podstránky: detail úlohy a detail konfigurace workerů. Tyto podstránky jsou tvořeny samostatnými komponentami.

Požadavky na panel Task Manager obsahovaly i požadavek na to, aby byl uživatel informován o změnách ve vykonávání úloh průběžně bez nutnosti obnovit stránku. Tato funkcionalita byla implementována pomocí periodického odesílání HTTP požadavků na Flower API. Bez nutnosti obnovovat stránku je tedy průběžně aktualizována hlavička panelu uvedená na Obrázku 19 a koláčový graf úloh uvedený na Obrázku 20. Koláčový graf je implementován pomocí knihovny *ngx-charts*.

[17] Commit dostupný na adrese https://github.com/mher/flower/commit/11523ca0

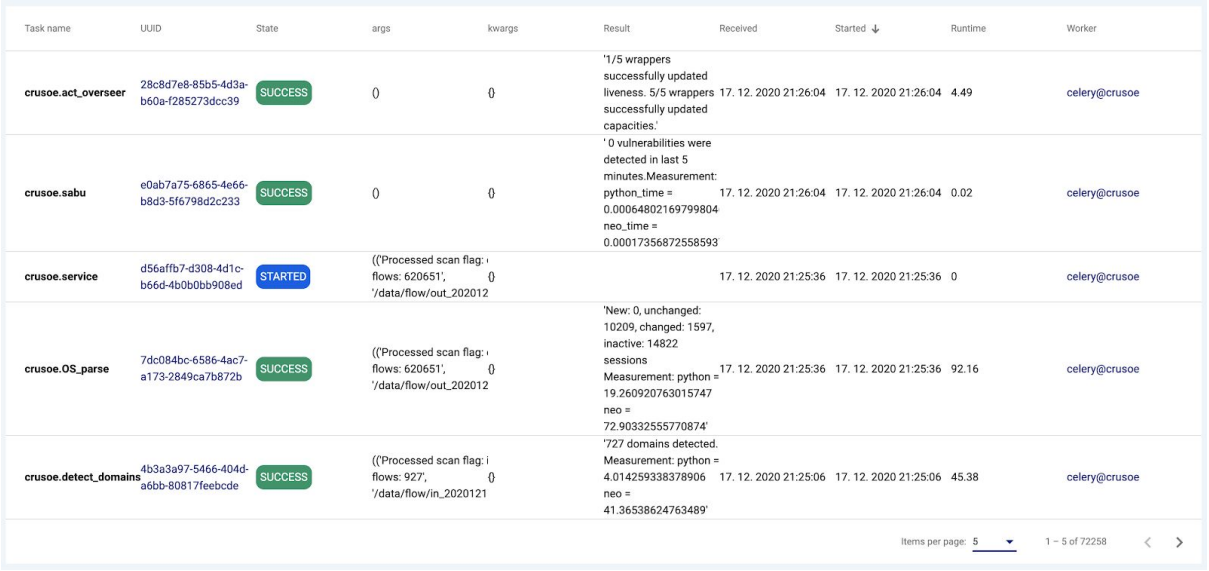


Obrázek 19: Hlavička podstránky v panelu Task Manager.



Obrázek 20: Koláčový graf úloh v panelu Task Manager.

Tabulka uvádějící seznam úloh zobrazená na Obrázku 21 umožňuje uživateli proklik na podstránku uvádějící detailnější informace o dané úloze. Byla implementována pomocí komponenty *mat-table* knihovny *Angular Material*. Umožňuje seřadit úlohy podle času začátku jejich vykonávání vzestupně a sestupně.



Task name	UUID	State	args	kwargs	Result	Received	Started ↓	Runtime	Worker
crusoe.act_overseer	28c8d7e8-85b5-4d3a-b60a-f285273dcc39	SUCCESS	()	{}	'1/5 wrappers successfully updated liveness. 5/5 wrappers successfully updated capacities.'	17. 12. 2020 21:26:04	17. 12. 2020 21:26:04	4.49	celery@crusoe
crusoe.sabu	e0ab7a75-6865-4e66-b8d3-5f6798d2c233	SUCCESS	()	{}	' 0 vulnerabilities were detected in last 5 minutes.Measurement: python_time = 0.00064802169799804 neo_time = 0.00017356872558593	17. 12. 2020 21:26:04	17. 12. 2020 21:26:04	0.02	celery@crusoe
crusoe.service	d56affb7-d308-4d1c-b66d-4b0b0bb908ed	STARTED	(('Processed scan flag: i flows: 620651', '/data/flow/out_202012	{}		17. 12. 2020 21:25:36	17. 12. 2020 21:25:36	0	celery@crusoe
crusoe.OS_parse	7dc084bc-6586-4ac7-a173-2849ca7b872b	SUCCESS	(('Processed scan flag: i flows: 620651', '/data/flow/out_202012	{}	'New: 0, unchanged: 10209, changed: 1597, inactive: 14822 sessions Measurement: python = 19.260920763015747 neo = 72.90332555770874'	17. 12. 2020 21:25:36	17. 12. 2020 21:25:36	92.16	celery@crusoe
crusoe.detect_domains	4b3a3a97-5466-404d-a6bb-80817feebcde	SUCCESS	(('Processed scan flag: i flows: 927', '/data/flow/in_2020121	{}	'727 domains detected. Measurement: python = 4.014259338378906 neo = 41.36538624763489'	17. 12. 2020 21:25:06	17. 12. 2020 21:25:06	45.38	celery@crusoe

Items per page: 5 1 – 5 of 72258

Obrázek 21: Seznam úloh v panelu Task Manager.

Panel Network Visualization

Panel Network Visualization umožňuje uživateli vyhledat na základě IP adresy zařízení v síti. Toto zařízení je následně zobrazeno jako uzel v grafu, pričemž jeho susední uzly blíže popisují toto zařízení, resp. přidávají kontext k danému zařízení. Sousedními uzly jsou: podsíť do které zařízení patří, přiřazené doménové jméno, bezpečnostní události spojené se

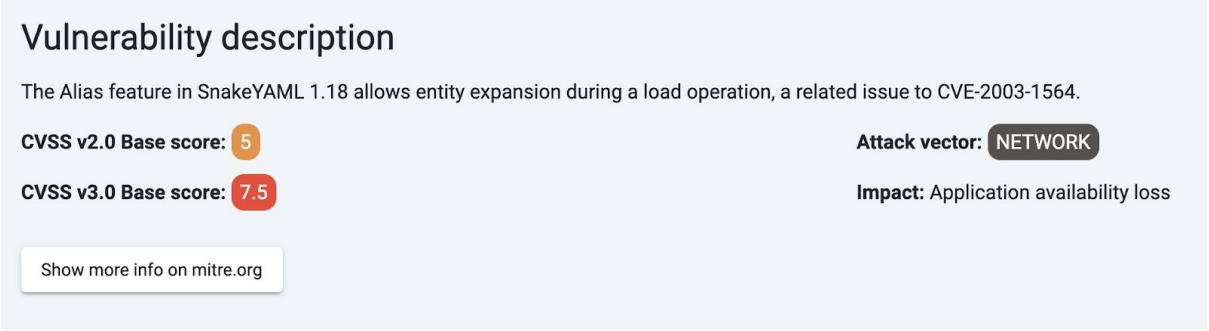
zařízením a software, který sa na zařízení nachází a byl detekovaný, například nástroji implemetujícími fázi Observe. K uzlům představujícím nainstalovaný software náleží uzly, které popisují zranitelnosti nacházející sa na tomto software.

Samotný graf je implementován pomocí knihovny *ngx-graph* ve verzi 8.0.0-rc.1. Pro získání dat využívá panel REST API, na které odešle požadavek, ve kterém jsou explicitně vyjmenována požadovaná data, tedy uzly a k nim náležející hrany z grafové databáze Neo4j, ve které jsou data typicky uložena. Data jsou následně zpracována a rozdělena do dvou proměnných typu pole představujících hrany a uzly grafu.

Panel Vulnerability

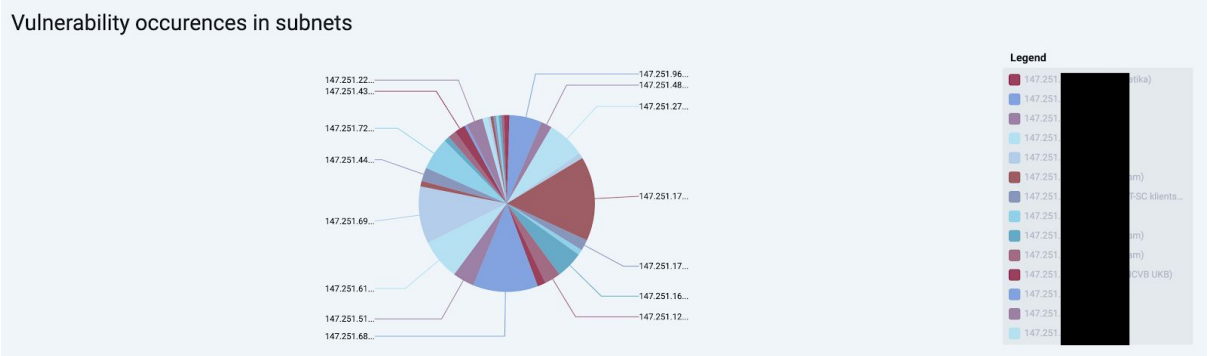
Panel Vulnerability umožňuje uživateli ověřit výskyt zranitelností ve spravované síti. Na panelu se nachází vyhledávací formulář, do kterého uživatel vloží CVE kód zranitelnosti, jejíž přítomnost v síti chce ověřit. Po stisknutí tlačítka *Search* se do REST API odešlou dva požadavky. První ověřuje, zda sa daná zranitelnost nachází v databázi. Pokud je zranitelnost v systému známá (je o ní uložena informace v databázi), jsou z databáze odeslány detaily o dané zranitelnosti, které byly do databáze staženy například z databáze NVD nebo jiného zdroje informací o zranitelnostech. Druhý požadavek slouží k získání seznamu zařízení, na kterých byla daná zranitelnost nalezena.

Základní informace o zranitelnosti jsou zobrazeny v přehledném výpisu, který je uveden na Obrázku 22. Výpis obsahuje i tlačítko s proklikem na stránku s kompletními informacemi o dané zranitelnosti uvedenými na webu cve.mitre.org.



Obrázek 22: Výpis informací o zranitelnosti.

Koláčový graf vizualizující počty výskytů zranitelnosti v podsítích sledované sítě umožňuje uživateli rychle získat přehled o tom, která podsíť je zranitelností nejvíce ohrožená. Tento graf je uveden na Obrázku 23 a implementován pomocí knihovny *ngx-charts*.



Obrázek 23: Vizualizace výskytu zranitelnosti v podsítích.

Uživatel má též k dispozici tabulku, která je tvořena jednotlivými zařízeními, na kterých se hledaná zranitelnost vyskytuje. Tabulka je tvořena čtyřmi sloupci: *IP Address*, *Domain name*, *Subnet*, *Software*. Sloupec *IP Address* obsahuje IP adresu stroje, na kterém byla nalezena daná zranitelnost s možností prokliku na panel Network Visualization, na kterém je po prokliku vidět dané zařízení s příslušným okolím ve formě grafu. Sloupec *Domain name* obsahuje doménové jméno příslušné k IP adrese zařízení, sloupec *Subnet* pak obsahuje adresu příslušné podsítě ve formátu CIDR. Ve sloupci *Software* se nachází název software, který obsahuje vyhledávanou zranitelnost a byl na daném zařízení detekován některým z detekčních nástrojů.

Panel Decide/Act

Panel Decide/Act umožňuje získat přehled o misích organizace, dostupnosti jednotlivých misí a též umožňuje uživateli aplikovat konfigurace misí podle doporučení nástrojů pro podporu rozhodování. Panel je dekomponován na několik různých komponent. Hlavní komponentou je komponenta *DecideAct*.

Jedním z požadavků bylo umožnit uživateli získat přehled o prvcích aktivní obrany. Tyto informace jsou zobrazeny v komponentě *DecideAct*, která je tvořena tabulkou z knihovny *Angular Material*. Tabulka obsahuje tři sloupce: *name*, *status* a *usage*. Uživatel má možnost kliknout na název prvku a zobrazí se mu detailní informace o příslušném prvku aktivní obrany. Toto detailní zobrazení je tvořeno samostatnou komponentou *Pao*, což do budoucna umožňuje tuto komponentu jednoduše rozšířit o další informace.

Dalším požadavkem bylo umožnit uživateli nastavení hodnoty *security threshold*. Tuto funkci zabezpečuje jednoduchý formulář s tlačítkem, po nastavení hodnoty získá uživatel zpětnou vazbu zobrazením malého panelu na spodní straně obrazovky. Tento panel je tvořený komponentou *MatSnackbar* knihovny *Angular Material*.

Dalším požadavkem bylo umožnit uživateli aplikovat různé konfigurace misí, přičemž uživatel bude mít možnost zjistit detailnější informace o těchto konfiguracích. Ve středu obrazovky sa tak nachází výčet misí, přičemž pod každým názvem mise je tabulka s výpisem konfigurací. Uživatel má možnost označit v tabulce některou z misí a aplikovat mise stisknutím příslušného tlačítka. Tlačítko je možné stisknout jen tehdy, když je vybrána konfigurace pro každou uvedenou misi. Uživatel má též možnost kliknout na název mise a

zobrazit tak detailnější popis mise, který je umístěn v samostatné komponentě *Mission*. Tabulka uvádějící konfigurace je tvořena čtyřmi sloupci: *Name*, *Integrity*, *Confidentiality*, *Availability*. Existuje možnost kliknout na název konfigurace a zobrazit tak detailní informace o této konfiguraci spolu s vizualizací toho, jak aplikování dané konfigurace ovlivní dostupnost misí.

Na panelu Decide/Act sa nachází komponenta, která vizualizuje závislost mise na informačních službách a podpůrných IT komponentách. Vizualizace sa nachází v samostatné komponentě, jejímž vstupem je struktura závislostí ve formátu JSON. Vykreslení grafu je zabezpečeno pomocí knihovny *ngx-graph*. Vzhledem k tomu, že jde o samostatnou komponentu, existuje možnosť přenést ji i do jiné Angular aplikace.

Testování

Testování je řešeno formou jednotkových testů, které jsou opakovaně použitelné i pro další rozvoj systému. Součástí aplikace jsou jednotkové testy služeb a komponent. Soubory s testy mají název s příponou .spec, například v případě souboru s názvem api.service.ts se jednotkové testy nacházejí v souboru s názvem api.service.spec.ts ve stejném adresáři. Testy využívají framework Jasmine ve verzi 3.4.0. Na samotné spouště testů je využívána knihovna Karma ve verzi 5.0.0.

Kontinuální integrace

Kontinuální integrace (anglicky continuous integration, dále CI) má za cíl automatizovat integraci změn zdrojového kódu do centrálního repozitáře. Za tímto účelem byl pro tento projekt zvolen nástroj *Gitlab CI/CD*.

Nástroj *Gitlab CI/CD* je konfigurovaný pomocí YAML souboru s názvem *.gitlab-ci.yml* umístěném v kořenovém adresáři repozitáře. Tento konfigurační soubor obsahuje nastavení a instrukce k samotnému průběhu CI. V projektu je v rámci CI ověřováno dodržování konvencí spuštěním příkazu *ng lint* a ověření úspěšnosti vykonání jednotkových testů spuštěním příkazu *ng test* v CI prostředí při každém odeslání změny zdrojového kódu do hlavní větve repozitáře.

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. Autorem software je Lukáš Matta, na návrhu se dále podílel Martin Husák. Autoři by rádi poděkovali Lukáši Sadlekovi a Stanislavu Špačkovi, kteří se podíleli na návrhu některých panelů.