Software pro aplikaci reaktivních opatření na prvcích aktivní obrany počítačové sítě

Stanislav Špaček, Milan Žiaran

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 aplikaci reaktivních opatření na prvcích aktivní obrany počítačové sítě*" (dále Act), 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 fáze Act se skládá z wrapperů prvků aktivní obrany (PAO) a z centrální služby act-overseer. Prvek aktivní obrany představuje bezpečnostní nástroj, který je schopen manipulací se síťovým provozem zastavit útočníka či jinak chránit připojená zařízení a služby, např. firewall, DNS firewall, nebo RTBH. Wrappery "obalí" přístupové rozhraní existujících PAO a navenek tak sjednotí přístupové funkce a návratové typy při volání libovolného prvku. Služba act-overseer pak spoléhá na jednotné rozhraní poskytované wrappery a slouží jako centrální styčný bod mezi PAO a software z ostatních fází projektu.

Služba act-overseer má dvě primární funkce – monitorovací a exekutivní. V rámci monitorovací činnosti act-overseer shromažďuje v reálném čase provozní data z připojených PAO. Tato data zahrnují informace o aktuální dostupnosti a kapacitě prvků. Aktuální data jsou ukládána v centrální databázi projektu pro další zpracování, vizualizaci, a prezentaci uživatelům systému. V rámci exekutivní činnosti act-overseer zpracovává požadavky fáze Decide na blokování či odblokování entit v síti pomocí změny konfigurace PAO a v závislosti na aktuálním stavu daného PAO tyto požadavky vykonává. Software fáze Act tedy představuje prostředek, který převádí výstupy software ostatních fází do podoby konkrétních opatření, tato opatření aplikuje na prvky aktivní obrany, a pomocí nich přímo manipuluje s provozem v síti.

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 nástrojů pro podporu automatizovaných a asistovaných reakcí na bezpečnostní události pomocí prvků aktivní obrany v prostředí chráněné vnitřní sítě. Software udržuje přehled o dostupných prvcích aktivní obrany a zpracovává vstupní příkazy od bezpečnostních operátorů nebo automatizovaných bezpečnostních systémů. Tyto příkazy kontroluje, rozděluje a vykonává na příslušných prvcích aktivní obrany s ohledem na jejich funkci, dostupnost, a volnou kapacitu. Aktuální stav prvků aktivní obrany a výsledky prováděných operací software zobrazuje v samostatné aplikaci dashboardu.

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

Kontaktní osoba: RNDr. Stanislav Špaček
Ústav výpočetní techniky
Masarykova Univerzita
Šumavská 416/15
Brno 602 00
e-mail: spaceks@ics.muni.cz
tel: +420 549 49 6094

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) využít centrální správu informací a jednotné rozhraní pro ovládání všech běžných prvků aktivní obrany, které jsou pro zajištění kybernetické bezpečnosti obvykle v síti nasazovány. Tím je eliminován problém decentralizace a nejednotnosti v přístupu k těmto prvkům, což zajišťuje snadnější ovládání bezpečnostního systému a rychlejší reakce na bezpečnostní incident. Využití výsledku je licencováno bez poplatku.

Popis výsledku

V současné době může být reakce na kyberbezpečnostní incidenty prováděna manuálně, nebo automaticky bez nutného zásahu operátora. Manuální reakce na incidenty spoléhá na expertní operátory, kteří v reálném čase incidenty řeší. Kyberbezpečnostních incidentů ale stále přibývá a nároky na odbornost a schopnosti operátorů úměrně tomu rostou. Automatizace reakce na kyberbezpečnostní incident je neustále se vyvíjející a dosud nevyřešenou problematikou.

Projekt CRUSOE využívá cyklus Observe-Orient-Decide-Act (OODA) a přizpůsobuje jej pro využití v prostředí reakce na kyberbezpečnostní incident. Projekt je dělen do jednotlivých fází podle kroků OODA cyklu a jednotlivé fáze implementují software poskytující podpůrné služby s funkcemi spadajícími do jejich oblasti. Cílem software vyvíjeného v rámci fáze Act projektu CRUSOE je částečně automatizovat reakci na bezpečnostní incidenty. Expertní operátor je stále nezbytný, ale software zjednodušuje manuální operace, nabízí automatizované procesy reakce, a poskytuje zpětnou vazbu o provedených opatřeních. Tím by měl usnadnit, zpřehlednit a zrychlit procesy reakce na incident.

Při vývoji software automatizace reakce na incidenty je nutné vzít v úvahu prostředí, do nějž bude software vstupovat, potažmo již existující nástroje, se kterými bude muset nutně spolupracovat. Bezpečnostní nástroje, neboli prvky aktivní obrany, se vyvíjejí jako prevence nebo reakce na kybernetický útok. Nutně tedy s vývojem procesů útočníků a s objevováním nových možných vektorů útoku vznikají nové a vyvíjejí se stávající PAO. Tento způsob postupného vývoje může vést k odlišným architekturám, implementacím, a tím i k různým způsobům práce s těmito nástroji. V typické síti, již má software CRUSOE chránit, se tedy prvky aktivní obrany vyskytují v různých fyzických či logických umístěních, jejich přístupová rozhraní podporují rozdílné sady funkcí, a návratové typy těchto funkcí jsou v různém formátu či s různou syntaxí.

Výše popsaná heterogenita a decentralizace je nežádoucí z pohledu operátora i z pohledu částečně automatizovaného systému. Operátor je nucen znát různé koncepce ovládání a interpretace dat pro různé PAO. Systém je rovněž nucen přizpůsobit se práci s množstvím různých prostředí. Před provedením částečné automatizace reakce na incident je tedy nutné vyřešit problémy centralizace a unifikace prvků aktivní obrany. Cílem software fáze Act není navrhovat ani vytvářet nové prvky aktivní obrany. Naopak základní myšlenkou je co nejefektivněji využít již existující bezpečnostní infrastrukturu a stávající bezpečnostní nástroje. Software se tedy snaží zajistit centralizaci a unifikaci PAO tak, aby mohl běžet i paralelně s existující bezpečnostní infrastrukturou.

Centralizace prvků aktivní obrany je dosaženo zavedením jediné centrální služby (act-overseer), která slouží jako kontaktní bod pro všechny prvky. Pokud software z jiné fáze projektu potřebuje manipulovat s některým PAO, děje se tak přes tuto centrální službu. Centrální služba má dvě primární funkce - (i) monitorovací funkci, tedy poskytovat aktuální

informace o stavu všech PAO a (ii) exekutivní funkci, tedy vykonávat konfigurace PAO na základě požadavků software z jiných fází projektu.

Unifikace prvků aktivní obrany je dosaženo zavedením wrapperů, které obalí stávající přístupové rozhraní prvku. Každý prvek tak má jasně definovanou množinu funkcí, kterou podporuje, a striktně daný formát a syntaxi vstupů a výstupů. Software fáze Act definuje pět kategorií běžných prvků aktivní obrany. Prvky byly do těchto kategorií rozděleny na základě funkce a typu entit v síti, se kterými pracují. Kategorie jsou následující:

- **DNS firewall** je bezpečnostní prvek, který umí blokovat dotazy na konkrétní domény. Typicky jde o DNS resolver vybavený blacklistem známých škodlivých domén. Dotaz na doménu, která se nachází na blacklistu, není přeposlán ani odpovězen. Příkladem DNS firewallu je DNS Response Policy Zones[1] implementovaný v BIND 9 DNS serveru.
- **Firewall** monitoruje síťový provoz, typicky na hranici vnitřní sítě, a umožňuje zablokovat přístup z vnější sítě k zařízením ve vnitřní síti, na základě IP adresy zařízení. Příkladem jednoduchého kompatibilního firewallu je například základní firewall v systému Linux - UFW[2].
- **Mail Filter** umožňuje zablokovat ve vnitřní síti přeposílání zpráv z konkrétních zdrojových e-mailových adres. Funkci tohoto PAO odpovídá poštovní server s podporou filtrování zpráv (např. Sendmail).
- **Remote Triggered Black Hole (RTBH)** je prvek aktivní obrany, který umožňuje zastavit nežádoucí síťový přenos dříve, než dosáhne chráněné sítě. Podrobněji je RTBH definováno v dokumentu společnosti CISCO[3].
- **User Blocker** je schopen blokovat přístup konkrétních uživatelů na základě jejich identifikátoru, ke zdrojům nacházejícím se v interní síti. Příkladem blokování uživatelů je například zamknutí uživatelského účtu v prostředí Microsoft Active Directory.

Software fáze Act byl vyvíjen jako součást systému CRUSOE, pro plné využití funkcí je tedy nutné použít jej spolu se software vyvinutým v ostatních fázích CRUSOE. Software ale může fungovat i samostatně, při dodržení minimální konfigurace obsahující nezbytné závislosti, mimo systém CRUSOE. Je možné využít pouze wrappery, vyvinuté pro prvky aktivní obrany. V takovém případě bude software sloužit jako mezilehlá vrstva API mezi prvky aktivní obrany a jiným nadřazeným systémem pro reakci na incidenty. Je také možné využít wrappery a monitorovací funkci centrální služby act overseer bez funkce exekutivní. Software pak bude poskytovat aktuální stavy všech připojených PAO.

Software navržený a vyvinutý ve fázi Act projektu CRUSOE slouží tedy primárně dvěma účelům. Jednak umožňuje přijímat rozhodnutí vydaná ve fázi Decide a na jejich základě

[1] https://www.dnsrpz.info/
[2] https://help.ubuntu.com/community/UFW
[3] https://www.cisco.com/c/dam/en/us/products/collateral/security/ios-network-foundation-protection-nfp/prod_white_paper0900aecd80313fac.pdf

manipulovat se síťovým provozem, a jednak poskytuje informace o důsledku těchto rozhodnutí zpět na začátek OODA cyklu do fáze Observe.

Metody a technologie

Při návrhu a vývoji software byly použity následující metody a technologie.

Wrappery

Funkce typu wrapper je definována jako funkce v softwarové knihovně nebo počítačovém programu, jejímž hlavním účelem je volání druhého podprogramu nebo systémového volání s malým nebo žádným dalším výpočtem[4]. Funkce typu wrapper se používají k usnadnění psaní počítačových programů tím, že se odstraní podrobnosti základní implementace podprogramu.

V projektu CRUSOE vystupuje v roli programu software fáze Act a v roli podprogramu prvek aktivní obrany. Wrapper přikryje implementaci PAO, která může být libovolná, a zavádí jednotné rozhraní s jasně definovanou množinou funkcí a dále formátem a syntaxí vstupů a výstupů, kterou PAO musí podporovat. Zároveň není nutné zasahovat do existujícího rozhraní prvku a případné existující a již nasazené bezpečnostní nástroje zůstávají funkční. Wrappery tak zajišťují unifikaci přístupových rozhraní PAO, na kterých je možné stavět další bezpečnostní funkce.

Nevýhodou wrapperů je nutnost dopsat před nasazením logiku funkcí wrapperu. Aby bylo dosaženo co nejširší kompatibility, jsou wrappery definovány jako framework bez jakékoli konkrétní implementace definovaných funkcí. Jejich obsah úzce závisí na konkrétním existujícím rozhraní PAO, ke kterému má být příslušný wrapper nasazen. V nejlepším případě wrapper využije již existující funkci rozhraní a pouze přeformátuje její vstup či výstup. Pokud ale existující rozhraní funkci neposkytuje, je nutné ji doimplementovat buď do samotného rozhraní, nebo do wrapperu.

Přes zmíněnou nutnost implementace logiky poskytují wrappery nezbytnou variabilitu, která umožňuje nasadit software fáze Act do různorodých prostředí. Software fáze Act tak není vázán na konkrétní prvky aktivní obrany či jejich různé implementace.

Modularita

Při vývoji software fáze Act byla snaha vytvořit modulární, spíše než jednolitý celek. Modulární celek je možné lépe přizpůsobit variabilitě prostředí, do kterého má být zasazen. Software je rozdělen na balíček act-overseer, zahrnující logickou část, a na balíček act-component, obsahující jednotlivé wrappery. Software jako celek je závislý na některých prvcích vyvinutých v rámci projektu, ale mimo fázi Act (viz kapitola Instalace, podkapitola Příprava). Jde o základní součásti systému jako je centrální databáze Neo4j a organizační služba Celery. Při návrhu byl kladen důraz na omezení těchto závislostí na nutné minimum

[4] Siler, Brian. *Special Edition Using Visual Basic 6*. Que Corp., 1998.

Balíček logiky tvoří, narozdíl od balíčku wrapperů, dále nedělitelný celek služby act-overseer složený ze čtyř navzájem propojených komponent – monitorování, exekutiva, konfigurace, a přístupové rozhraní. Vztahy mezi těmito komponentami jsou podrobněji popsány v podkapitole Architektura. Balíček logiky musí být instalován jako celek, ale je možné po instalaci využívat pouze část služeb v závislosti tom, jaké další software z ostatních fází projektu Crusoe byly nasazeny. Monitorování PAO je proces provozovaný výhradně v rámci software fáze Act a nutných závislostí. Je tak možné jej použít v každém případě. Oproti tomu exekutivní správa PAO je závislá na výstupech software fáze Decide a pokud není tento software nasazen, služba nebude dostupná.

Wrappery jsou vzájemně nezávislé a je tedy možné použít je samostatně. Pokud se v prostředí, kam je software nasazován, některý PAO nevyskytuje, jeho wrapper nemusí být nasazován. Naopak ale každý PAO, který má být spravován software fáze Act musí mít svůj wrapper nasazen. Postup pro případné vyřazení wrapperu z instalace software je popsán v kapitole Instalace, podkapitole Příprava.

Možné modulární variace nasazení software jsou tedy následující:

- **Libovolná kombinace wrapperů** – zajistí pouze unifikaci rozhraní prvků aktivní obrany. Další služby je nutné zajistit jinými, např. již existujícími nástroji.
- **Libovolná kombinace wrapperů + služba act-overseer** – zajistí jednak unifikaci rozhraní prvků aktivní obrany a také následnou centralizaci sběru dat a správy těchto prvků. Tato možnost je doporučena při nasazení všech software vyvinutých v rámci projektu Crusoe a dokumentace se vztahuje právě k této variantě.

Django Class-Based Views

Pro implementaci software fáze Act byl použit framework Django. Django je open-source webový aplikační framework napsaný v programovacím jazyce Python. Endpointy přístupových rozhraní software jsou implementovány pomocí Class-Based Views dědících z třídy APIView. Detailní popis techniky Class-Based Views je možné najít ve veřejně dostupné dokumentaci[5].

Ansible

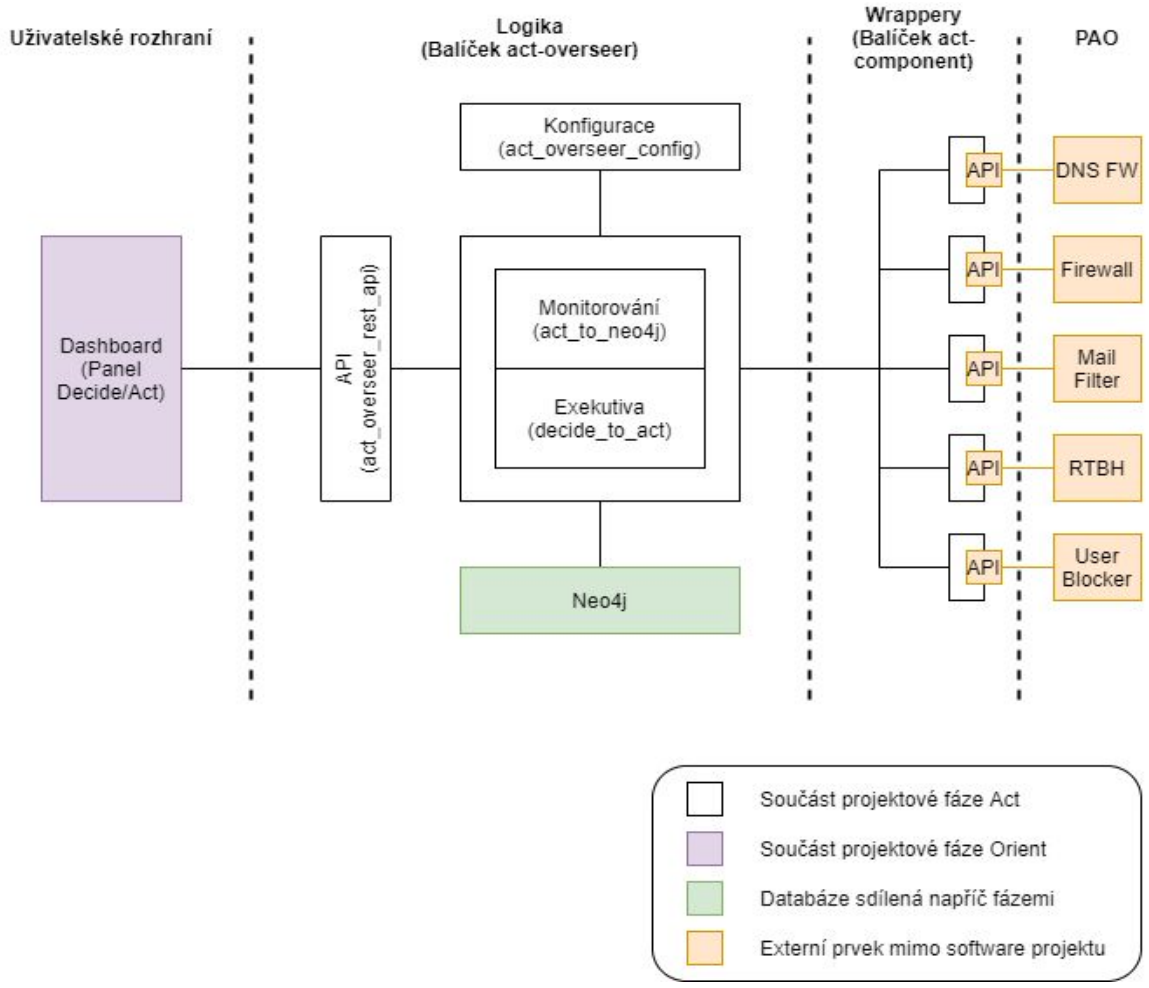
Pro automatizaci nasazení software vyvinutého v rámci projektu Crusoe byl použit nástroj Ansible. Repozitář Crusoe Ansible obsahuje kód, který po počáteční přípravě instaluje software vyvinutý v projektu automaticky. Kód a dokumentace nástroje Ansible jsou dostupné jako open-source[6]. Instalace software je možná i bez využití Ansible, pak je ale nutné provést všechny kroky popsané v Ansible rolích manuálně. Tato dokumentace doporučuje a dále popisuje nasazení software pomocí nástroje Ansible.

[5] https://www.django-rest-framework.org/api-guide/views/
[6] https://www.ansible.com/

Architektura

Zvolená architektura software vychází z modulárního přístupu a je zobrazena na obrázku 1. Na nejvyšší úrovni se software fáze Act skládá ze dvou balíčků – služba act-overseer a balíček wrapperů act-component. Balíček act-component se dále skládá jen z jednotlivých wrapperů pro různé PAO. Oproti tomu službu act-overseer lze rozdělit na menší komponenty. Balíček act-overseer zahrnuje implementaci základních funkcí služby, monitorování a exekutivu, soubor s aktuální konfigurací služby, a přístupové rozhraní.

Pro znázornění zapojení software mezi ostatní software vyvinutý v projektu, jsou na obrázku obsaženy i prvky spadající mimo fázi Act. Příslušnost těchto prvků k různým fázím i mimo projekt je popsána v legendě. Prvním napojením je přístupové rozhraní služby act-overseer. Přístupové rozhraní služby je možné ovládat ze stránky Decide/Act v dashboardu projektu. Toto napojení zajišťuje přenos informací z předchozích fází cyklu OODA do fáze Act. Druhý komunikační kanál je napojen přímo na logiku služby. Monitorovací a exekutivní funkce mají přístup přímo do centrální projektové databáze Neo4j. Aktualizací dat v databázi je zajištěna zpětná vazba na operace prováděné s PAO a tedy obrátka cyklu OODA. Wrappery jsou napojeny na přístupová rozhraní PAO, nacházející se mimo projekt CRUSOE. Toto napojení je třeba doimplementovat na základě prostředí, do kterého je software nasazován.



Obr. 1: Schéma propojení modulů software fáze Act a napojení mezi ostatní software vyvinutý v rámci projektu CRUSOE. Název v závorce označuje název modulu tak, jak je uložený v repozitáři projektu.

Balíček act-overseer

Organizační služba je základní logickou komponentou software fáze Act. Sdružuje v sobě moduly zajišťující monitorovací a exekutivní funkce na PAO a také konfigurační údaje nezbytné pro běh těchto modulů. Tato komponenta je jediným centrálním prvkem v jinak decentralizované architektuře PAO. Skládá se z následujících součástí:

- Monitorování
- Exekutiva
- Konfigurační soubor
- Přístupové rozhraní

Monitorování

Monitorovací funkce zajišťuje aktuální data o stavu PAO v databázi Neo4j. Při instalaci software fáze Act je tato funkce spuštěna a okamžitě aktualizuje data v databázi. Další spuštění monitorování je prováděno pomocí projektové organizační služby Celery a případně podle potřeby. Monitorování je spouštěno periodicky v intervalech nastavitelných v konfiguraci Celery. Monitorování může být vyvoláno rovněž vyvoláním exekutivní funkce a provedením změn konfigurace PAO.

Monitorování volá funkci wrapperů pro zjištění maximální a aktuální kapacity, a funkci pro ověření živosti PAO. Získaná data okamžitě zapisuje do databáze, kde jsou přístupná software ostatních fází projektu. Data nejsou před zápisem do databáze nijak transformována ani upravována. Úspěšná i neúspěšná volání funkcí wrapperů jsou monitorovací funkcí logována a ukládána do definované lokace. Podrobněji je algoritmus monitorování popsán v programátorské dokumentaci.

Exekutiva

Exekutivní funkce poskytuje ve spolupráci se software fáze Decide částečnou automatizaci konfigurace PAO. Funkci spouští operátor manuálně vybráním doporučených konfigurací pomocí GUI v dashboardu a povelem k provedení nastavení. Po skončení autokonfigurace funkce informuje operátora o průběhu nastavení a případných chybách.

Funkce z GUI dashboardu obdrží seznam doporučených konfigurací, které mají být aplikovány na síť. Detaily daných konfigurací zjistí funkce přímo z databáze Neo4j. Poté pomocí procesů vlastní logiky (podrobně popsáno v uživatelské a programátorské dokumentaci) transformuje konfigurace na seznamy odblokací a blokací. Tyto seznamy pak pomocí volání příslušných funkcí wrapperu aplikuje na daný PAO. Výsledky blokovacích i odblokovacích operací jsou službou logovány. Důležité události a celkový výsledek nastavení lze sledovat v GUI dashboardu v reálném čase. Podrobněji je algoritmus exekutivy popsán v programátorské dokumentaci.

Konfigurační soubor

Konfigurační soubor act-overseer se nachází v /act-overseer/data/act_overseer_config. Obsahuje 5 nastavitelných parametrů, které jsou pro běh služby nezbytné. Soubor obsahuje následující parametry (hodnoty slouží jako příklad):

```
{
  "security_treshold": 50,
  "log_path": "/var/log/crusoe/",
  "user": "user",
  "password": "pass",
  "server_url": "http://172.18.1.10:8088"
}
```

Parametr *security_treshold* obsahuje aktuální hodnotu parametru platnou pro běh exekutivní funkce služby act-overseer. Detailní popis využití a významu této hodnoty se nachází v uživatelské a programátorské dokumentaci. Parametr *log_path* obsahuje cestu k adresáři, do nějž má služba act-overseer ukládat soubory zaznamenávající události při běhu funkcí. Parametry *user* a *password* obsahují přihlašovací údaje k projektovému REST API sdružujícímu přístup do databáze Neo4j s ostatními funkcemi. Parametr *server_url* obsahuje url, nebo ip adresu a port včetně protokolu, kde lze najít projektové REST API.

Přístupové rozhraní

Přístupové REST API poskytuje funkce pro ovládání chování služby act-overseer. Umožňuje přečíst a změnit hodnotu parametru *security_treshold*. Dále umožňuje provést reinicializaci prvků aktivní obrany v databázi, pokud dojde k přidání, odebrání, nebo změně existujícího prvku. Poslední klíčovou funkcí rozhraní je spuštění exekutivní funkce služby act-overseer a autokonfigurace PAO podle konfigurací doporučených software fáze Decide a vybraných operátorem. Chování monitorovací funkce není ovládáno přes toto rozhraní, ale přes konfiguraci organizační služby Celery. Podrobná specifikace funkcí rozhraní je uvedena v příloze B.

Balíček act-component

Balíček act-component obsahuje wrappery vyvinuté pro definované kategorie PAO. Wrappery definují sadu funkcí, které musí PAO podporovat, a také formát a syntaxi vstupů a výstupů těchto funkcí. Wrappery jsou rozděleny do složek, každá složka obsahuje všechny nutné závislosti wrapperu, aby bylo možné nasadit wrappery odděleně na různá zařízení. Podrobné specifikace wrapperů jsou součástí této dokumentace a nachází se v příloze A. Tato specifikace je dodána i v elektronické formě v tomto balíčku ve složce /act-component/specification.

Pro potřeby testování software v projektu CRUSOE byl vyvinut zvláštní simulovaný prvek aktivní obrany. Testování software v reálném prostředí a na reálných PAO by jinak mohlo způsobit výpadky spojení či nedostupnost služeb. Tento simulovaný prvek aktivní obrany nahrazuje reálný firewall a nachází se ve složce /act-component/simulated-pao-firewall.

Prvek se navenek chová stejně, jak by se choval reálný firewall, ale neaplikuje provedená opatření na síť. Prvek je možné použít k testování celku všech software vyvinutých v rámci projektu CRUSOE ve virtuálním i reálném prostředí. Jeho instalace namísto wrapperu reálného firewallu je popsána v kapitole Instalace.

Návod k instalaci

Návod k instalaci software fáze Act vychází z použití Ansible. Software obsahuje vše potřebné pro automatické nasazení jak software fáze Act. Před spuštěním instalace je nutné změnit obsah některých konfiguračních Ansible souborů v závislosti na prostředí, kam je software nasazován. Tyto změny jsou popsané v podkapitole Příprava.

Příprava

- Rozbalení archivu software do lokace /crusoe/act/.
- Vyplnění hesla k rozhraní služby act-overseer v ansible/ansible/group_vars/all/vault. Soubor Soubor vault je poté třeba zašifrovat pomocí nového hesla a toto vyplnit do souboru ansible/vault_pass.
- Úprava souboru /anisble/act.yml
    - Obsažená konfigurace slouží jako příklad zahrnující všechny dostupné součásti.
    - Je nutné vyplnit IP adresy a porty strojů, na kterých budou součásti software nasazeny.
    - Službu act-overseer je doporučeno nasadit na server s ostatními software z jiných fází, na zadaném portu poběží přístupové rozhraní.
    - Wrappery
        - Upozornění: wrapper neobsahuje logiku, tu je nutné před nasazením doplnit - liší se podle prostředí, kam je wrapper nasazován.
        - Je možné nasadit je na libovolný server v síti, ale doporučen je server, na kterém běží PAO, k němuž se wrapper vztahuje.
        - Pokud se některý z podporovaných PAO v síti nevyskytuje, měl by být z act.yml smazán.
        - Při testovacím nasazení je možné použít simulovaný PAO firewall. U wrapperu s parametrem *wrapper: firewall* je potom třeba změnit parametr *dst_wrapper* z hodnoty *firewall-wrapper* na *simulated-pao-firewall*.
        - U zbylých PAO a act overseer jsou uvedené parametry povinné.
        - ip … ip adresa serveru, kam má být wrapper instalován
        - portnumber … port serveru, kde má wrapper běžet
        - server_name … FQDN serveru, kam má být wrapper instalován
        - pao_name …. název PAO {dnsfw, firewall, userBlock, MailFilter, rtbh}
        - maxCapacity: "0"
        - usedCapacity: "0"
        - freeCapacity: "0" … iniciální kapacity není nutné upravovat, po instalaci jsou automaticky doplněny službou act-overseer

Instalace

Repozitář Crusoe Ansible definuje virtuální prostředí, do něhož je možné software nasadit. Instalaci je možné zahájit příkazem **vagrant up** v kořenovém adresáři repozitáře. Instalace pak proběhne automaticky. Při nasazení do jiného než připraveného testovacího prostředí je třeba spustit Ansible přímo, s parametrem playbook.yml.

Instalace byla testována pro Ansible verze 2.9.10. Pokud je software nasazován do virtuálního prostředí definovaného v repozitáři, vyžaduje dále na hostitelském zařízení software Vagrant 2.2.9 a VirtualBox 6.0.24 r139119. Instalace bude pravděpodobně možná i na novějších verzích těchto nástrojů.

Po instalaci zkontrolujte, zda je v konfiguračním souboru služby act-overseer (/usr/local/lib/python3.7/dist-packages/act_overseer/data/act_overseer_config) v parametru server_url vyplněno url či IP:port REST API. Je nutné tento parametr vyplnit včetně protokolu http (např. http://172.18.1.10:8088). Tato adresa ale musí být mezi povolenými adresami v Apache sites-enabled na serveru s rozhraním centrální databáze Neo4j. Pokud act-overseer běží na stejném serveru jako přístupové rozhraní (doporučené a zároveň výchozí nastavení), lze použít místo adresy výraz *localhost* který není nutné v Apache explicitně povolovat.

Kontrola výsledků instalace

Po instalaci je doporučeno provést kontrolu běhu všech nasazených služeb. Pro kontrolu běhu lze použít postupy popsané dále v sekci Uživatelská dokumentace - Troubleshooting.

Uživatelská dokumentace

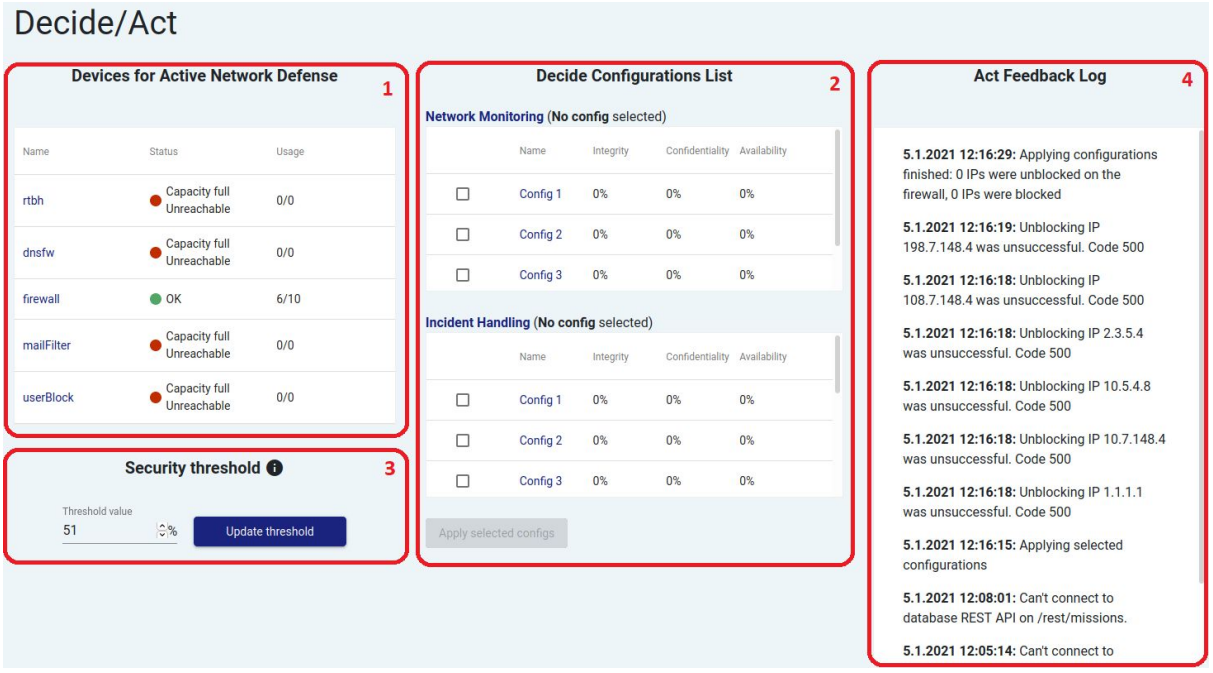
Software fáze Act je primárně určen pro dva případy užití:

1. Centralizace informací z dostupných prvků aktivní obrany
2. Provádění automatického nastavení prvků aktivní obrany podle konfigurací doporučených software fáze Decide

Tato sekce popisuje všechny části software, se kterými bude operátor pracovat při běžném použití.

Dashboard 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 2.



Obr. 2: Stránka software fáze Act v dashboardu.

Obrázek 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 3. 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.

Obr. 3: Seznam prvků aktivní obrany. Je připojen jediný prvek (firewall), ostatní nebyly nasazeny.

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:

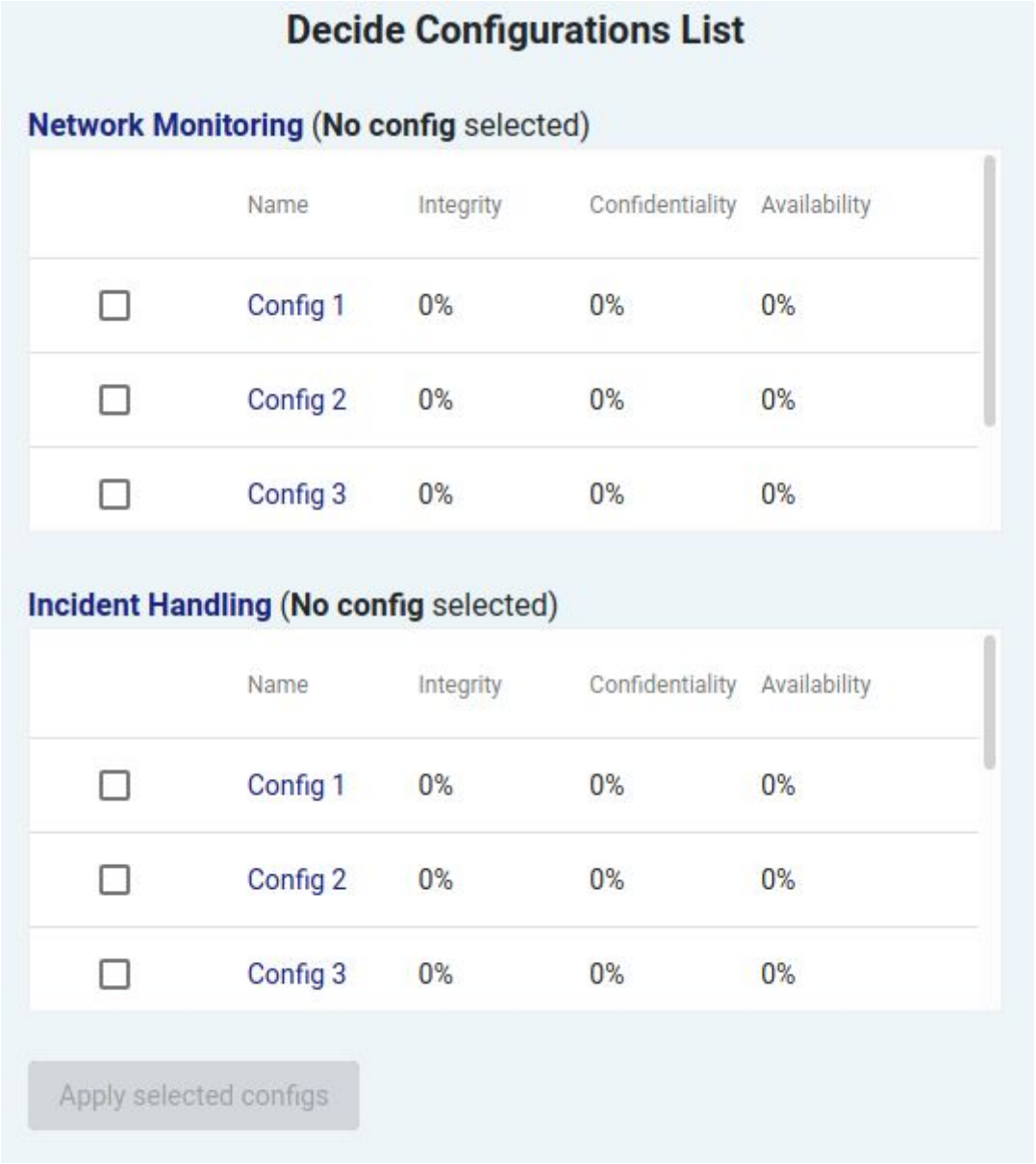
- Zelená - vše v normě.
    - OK - Nesplněna žádná z podmínek níže.
- Žlutá - prvek funguje s výhradami.
    - Capacity 90 % full - Zjištěno naplnění maximální kapacity z 90 %.
    - Last liveness check failed - Poslední kontakt se povedl před více než 10 minutami.
- Červená - prvek má problém, který 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.

Poznámka: Intervaly kontroly živosti je možné nastavit v centrální organizační službě Celery.

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. Z pohledu fáze Act každá konfigurace vyžaduje zablokování či naopak odblokování konkrétních strojů v síti tak, aby bylo naplněno poslání mise. Panel seznamu misí a konfigurací je zobrazen na obrázku 4.



Obr. 4: Seznam misí a konfigurací. Jsou definovány 2 mise - 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 je

možné zobrazit mapu zařízení, které tato mise vyžaduje. 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.

Poznámka: Před aplikací konfigurací je nutné vybrat pro **každou** misi **právě jednu** konfiguraci.

Aktuální hodnota 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 dokumentaci software fáze Decide. 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>.

Obr. 5: Panel nastavení Security threshold.

Význam nastavení parametru Security threshold je následující. Při aplikaci konfigurace bez nastavení Security threshold by docházelo k blokování všech zařízení, která nejsou kritická pro misi, k níž se konfigurace vztahuje. Zařízení by tedy byla blokována bez ohledu na jejich míru ohrožení. Takový postup je zbytečně omezující, neboť ve standardním prostředí není nutné blokovat zařízení s relativně nízkou mírou ohrožení.

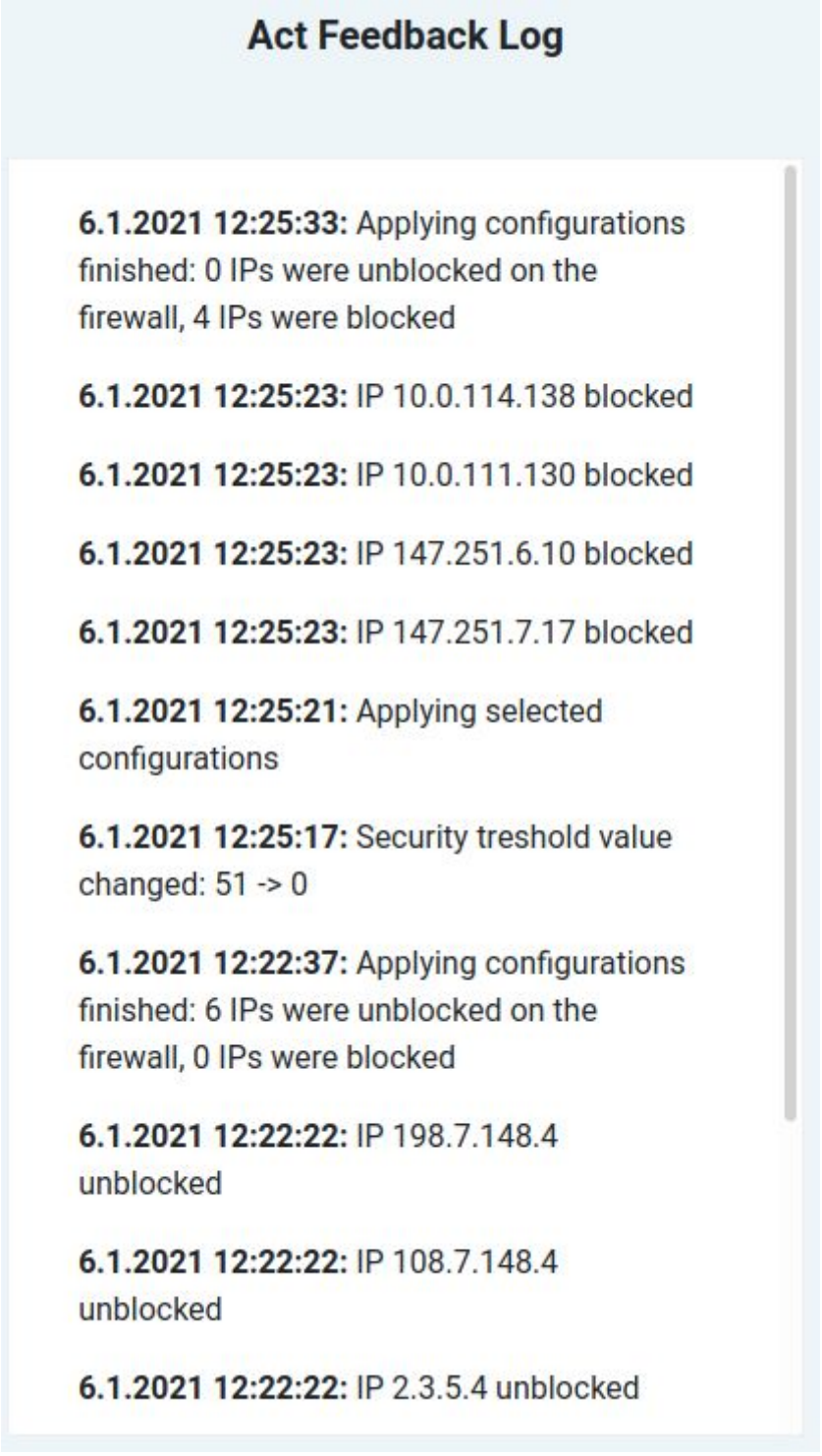
Nastavení hodnoty Security threshold umožňuje vyjmout z blokování zařízení s mírou ohrožení **menší** než je nastavená hodnota. Na obrázku 5 je zobrazena Security threshold s hodnotou 51. Pokud by byly konfigurace aplikovány na síť s tímto nastavením, budou zablokována všechna zařízení s mírou ohrožení vyšší než 51. Naopak zařízení s nižší mírou ohrožení nebudou blokována, přestože to některá konfigurace doporučuje.

Při nastavení hodnoty Security threshold 0 se systém při aplikaci konfigurací na síť chová jako by tento parametr nebyl implementován a blokováno je vše podle konfigurací. Při opačném extrému, Security threshold = 100, naopak nedojde k blokování žádných zařízení.

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 6. Zprávy zobrazované v panelu jsou z důvodu zachování přehlednosti filtrované a nezachycují kompletní podobu logu software fáze Act. Při řešení potíží s během software přesahujících běžné použití je doporučeno zkontrolovat log soubory software (viz podkapitola Troubleshooting).



Obr. 6: Seznam aktuálních událostí ve fázi Act. Na obrázku je zachycena (zdola nahoru) aplikace konfigurací, změna Security treshold, a opětovná aplikace konfigurací s rozdílným výsledkem.

Monitorování prvků aktivní obrany

Akce popsané v této kapitole popisují funkce dostupné pro případ užití 1, tedy centralizace informací z dostupných prvků aktivní obrany. Aktuální stav všech prvků aktivní obrany je možné sledovat na panelu Seznam prvků aktivní obrany. Stav je aktualizován v pravidelných intervalech nastavitelných v konfiguračním souboru centrální organizační služby Celery. Popis dat zobrazovaných v panelu se nachází v předchozí kapitole.

Změny prvků aktivní obrany

Pokud je nutné po nasazení software fáze Act provést změny na prvcích aktivní obrany, je nutné provést reinicializaci prvků. Změny vyžadující reinicializaci zahrnují:

- Přidání nebo odebrání prvku aktivní obrany.
- Přesun prvku aktivní obrany na jinou IP adresu a/nebo port.

Reinicializace prvků se provádí přes vlastní funkci /act/initialize rozhraní služby act-overseer. Této funkci je třeba poslat (např. příkazem curl) JSON soubor ve formátu podle příkladu v repozitáři act-overseer.

Cesta k souboru je /act-overseer/specification/ood-apis/pao-init.json. Tento soubor je nutné opravit tak, aby odpovídal aktuálním umístěním prvků aktivní obrany.

Postup:

- Pokud se prvek nenachází v síti, je možné jej z inicializačního souboru smazat.
- U zbylých prvků upravte parametry IP a port tak, aby ukazovaly na wrapper příslušného prvku.
- Ostatní parametry (kapacity, kontakt) nechte v základním nastavení, budou automaticky vyplněny monitorovací službou po reinicializaci.
- Příkazem curl předejte JSON soubor endpointu /act/initialize rozhraní act-overseer

Aplikace konfigurací na síť

Akce popsané v této kapitole popisují funkce dostupné pro případ užití 2, tedy provádění automatického nastavení prvků aktivní obrany podle konfigurací doporučených software fáze Decide.

Panel Seznam misí a konfigurací zobrazuje všechny definované mise a k nim dostupné konfigurace. Kliknutím na název mise je možné zobrazit její detaily – slovní popis a hodnocení důležitosti mise. Seznam misí je aktualizován automaticky, pokud je do databáze přidána nová mise.

Ke každé misi pak Decide poskytuje možné konfigurace. Konfiguraci si lze zjednodušeně představit jako seznam zařízení, která musí být dostupná pro splnění mise. Z pohledu software fáze Act tedy konfigurace určuje, ke kterým zařízením je možné blokovat přístup,

aby mise mohla být splněna a zároveň zařízení v síti nebyla dostupná, pokud to není nezbytné pro plnění některé mise. Kliknutím na název konfigurace je možné zobrazit detailní informace, zejména seznam dostupných zařízení a mapu sítě. Seznam dostupných zařízení výslovně jmenuje zařízení, která zajistí běh mise v dané konfiguraci. Mapa sítě zobrazuje všechna zařízení, se kterými konfigurace pracuje. Zvýrazněna červeně jsou ta zařízení, která konfigurace doporučuje blokovat. Detailní popis problematiky misí a konfigurací se nachází v dokumentaci software fáze Decide.

Výběr nejvhodnější konfigurace pro danou misi závisí na několika kritériích. Základními kritérii jsou požadavky na integritu, dostupnost, a důvěrnost. Každá konfigurace je ohodnocena mírou ohrožení daného atributu vyjádřenou v procentech. **Nižší** hodnota u daného atributu tedy znamená **lepší** výsledek dané konfigurace oproti jiným s vyšší mírou ohrožení. Dalším kritériem je požadavek na provoz konkrétního zařízení, které ale není kritické pro plnění žádné mise, případně může být pro výkon mise nahrazeno jiným zařízením. Kontrola zařízení dostupných pro danou konfiguraci ukáže, zda aplikace konfigurace na síť nezablokuje takové zařízení.

Jakmile je pro každou misi vybrána v panelu právě jedna konfigurace, je možné aplikovat je na prvky aktivní obrany stiskem tlačítka Apply selected configs. To zahájí proces služby act-overseer. Postup tohoto procesu je možné sledovat v panelu Seznam aktuálních událostí ve fázi Act. V tomto panelu se nejprve zobrazí zpráva o zahájení, poté o úspěchu či neúspěchu provedení jednotlivých blokování a odblokování, a nakonec závěrečná zpráva o ukončení procesu. Příklad je zobrazen na obrázku 5.

Poznámka: Při aplikaci nových konfigurací jsou předchozí konfigurace z prvků aktivní obrany smazány, aby se zabránilo postupnému překryvu konfigurací.

Postup:

- Pro každou misi je nutné vybrat právě jednu konfiguraci, např. na základě nejnižší míry ohrožení integrity, důvěrnosti, nebo dostupnosti.
- Proces aplikace konfigurací je zahájen stiskem tlačítka Apply selected configs.
- V panelu Seznam aktuálních událostí je zobrazen průběh procesu.

Troubleshooting

Tato kapitola obsahuje postupy pro kontrolu běhu jednotlivých součástí software fáze Act. Zdokumentovány jsou také některé problémy, které by se mohli při provozu software objevit, a jejich možná řešení.

- Jsou dostupné všechny instalované wrappery na zadaných portech?
    - Každý wrapper je možné kontaktovat dotazem na REST rozhraní ve formátu <ip>:<port>/<název_pao>/capacity. Pokud na tento dotaz vrací wrapper odpověď, lze jej považovat za funkční. Obdobně lze volat i ostatní endpointy wrapperu a sledovat odpovědi.
    - Je možné, že spadla služba wrapperu a je nutné ji restartovat.
- Je dostupná centrální služba act-overseer?

    - Centrální službu act-overseer lze kontaktovat dotazem na REST rozhraní ve formátu localhost:<port>/act/treshold. Pokud na tento dotaz vrací služba odpověď, lze ji považovat za funkční.
    - Je možné, že se služba act-overseer zastavila a je nutné ji restartovat.
- Běží automatický monitoring prvků aktivní obrany?
    - Automatický monitoring PAO v nastavených intervalech (5 minut v počátečním nastavení) zjišťuje aktuální živost a kapacitu všech připojených prvků. Pokud služba běží, měly by se kapacity PAO v databázi a potažmo na stránce Act v dashboardu změnit na aktuální ihned po dokončení instalace.
    - Pokud dochází k chybě při zápisu do databáze, jsou tyto chyby zachyceny v logu /var/log/crusoe/act_overseer.log na serveru, kde běží act-overseer.
- Dochází k nějakým jiným chybám při běhu služeb?
    - Všechny služby logují svůj běh, chyby a provedené operace jsou ukládány do zvláštních log souborů
    - /var/log/crusoe/act_overseer.log
        - běh služby monitorující PAO
    - /var/log/crusoe/act_overseer_rest_api.log
        - příkazy zadané skrze rozhraní služby act_overseer
    - /var/log/crusoe/act_decide_to_act.log
        - běh služby automatické konfigurace PAO na základě konfigurací doporučených software fáze Decide
- Aplikace konfigurací na mise neproběhne s chybou “Can’t connect to database REST API on /rest/missions”.
    - Službě act-overseer se nepovedlo připojit k REST rozhraní databáze Neo4j. Nejprve zkontrolujte, že rozhraní běží na dané IP:port, nebo url.
    - Pokud rozhraní běží, zkontrolujte, zda je v konfiguračním souboru act-overseer (/usr/local/lib/python3.7/dist-packages/act_overseer/data/act_overseer_config) v parametru server_url vyplněno url či IP:port REST API. Je nutné tento parametr vyplnit včetně protokolu http (např. http://10.0.0.1:8088).
- Simulovaný firewall nedokáže blokovat ani odblokovat žádné IP adresy (chyba 500).
    - Soubor firewallu nemá nastavena dostatečná oprávnění k provedení změn.
    - Nastavte souboru /var/www/simulated-pao-firewall/firewall_wrapper_project /simulated_pao_firewall přístupová práva 666. Soubor se nachází na serveru, kde byl nasazen firewall wrapper.

Programátorská dokumentace

Software fáze Act je primárně určen pro dva případy užití:

1. Centralizace informací z dostupných prvků aktivní obrany
2. Provádění automatického nastavení prvků aktivní obrany podle konfigurací doporučených software fáze Decide

Pro splnění těchto případů užití byly navrženy dva algoritmy, které byly následně implementovány pomocí jazyka Python a frameworku Django. Tato sekce je věnována popisu těchto algoritmů a také funkcí, jimiž byly algoritmy implementovány. Dále jsou popsány všechny další implementované pomocné prvky, jako přístupová rozhraní a wrappery.

Algoritmus centralizace informací z PAO

Centralizace informací probíhá podle následujícího algoritmu:

1. Act overseer kontroluje živost všech PAO periodicky v pevně daném intervalu *t*.
2. Výsledky ihned aktualizuje v Neo4J.
3. Panel Decide/Act jako součást dashboardu si s intervalem *t* zjišťuje z Neo4J data o dostupnosti a kapacitě PAO.

Algoritmus autokonfigurace PAO

1. Operátor vybere v dashboardu právě jednu konfiguraci pro každou misi a potvrdí provedení. Dashboard odešle na endpoint /act/protect_missions_assets act_overseer_api seznam vybraných dvojic mise-konfigurace.
2. Overseer získá seznam všech misí z Neo4j (endpoint DECIDE/missions) a ověří, že json obsahuje všechny mise, pokud ne, vyhodí příslušnou chybu a skončí.
3. Overseer z databáze získá seznam všech strojů a seznam strojů kritických pro dvojice mise-konfigurace.
    a. endpoint DECIDE/missions/hosts vrací IP strojů ze **všech** misí a ke každému tu nejhorší (nejvyšší) hodnotu ohrožení availability, confidentiality a integrity, které má v některé z misí.
    b. endpoint DECIDE/mission/<name>/configuration/<config_id>/hosts vrací IP strojů důležitých pro misi v dané konfiguraci
    c. tedy množinový rozdíl výsledků volání funkcí API: {DECIDE/missions/hosts} - {DECIDE/mission/<first-mission>/configuration/<config-id1>/hosts} - ... - {DECIDE/mission/<last-mission>/configuration/<config_id2>/hosts} = seznam POTENCIÁLNÍCH blokací
    d. {seznam POTENCIÁLNÍCH blokací} - {stroje s bezp. hodnocením < security treshold} = seznam NOVÝCH blokací (SNB)
        i. endpoint DECIDE/missions/hosts vrací seznam všech hostů v misích + jejich nejhorší parametry confidentiality, availability a integrity (ošetřit případ, kdy stroj nebude v seznamu - zalogovat a neblokovat)

        ii. bezpečnostní hodnocení stroje = aritmetický průměr parametrů confidentiality, availability a integrity stroje
4. Řešení problému kdy se původní konfigurace překrývá s novou konfigurací
    a. z funkce wrapperu overseer zjistí seznam aktuálně blokovaných IP adres na Firewallu = seznam blokací (SB)
    b. IP je obsažena v SNB && SB -> žádná akce
    c. IP je v SNB && !SB -> zablokovat IP
    d. IP je v !SNB && SB -> odblokovat IP
    e. seřadí seznam pravidel - odblokace první (šetříme kapacitu)
    f. vzniklý seznam pravidel již lze aplikovat na PAO (firewall)
5. Overseer zjistí aktuální parametr freeCapacity a dostupnost PAO pomocí funkcí wrapperu.
    a. Zkontroluje volnou kapacitu na PAO, ke kterým má seznam akcí (firewall). Volná kapacita je spočtena jako freeCapacity + počet míst uvolněných odblokováním
    b. Vrátí chybu na operace na operace mimo kapacitu. Tyto operace neprovede ani částečně (pokud nebude volná kapacita, proces skončí s chybou)
6. Vykoná sekvenčně akce ze seznamů pravidel na příslušných PAO.
7. Zjistí pomocí wrapperu aktuální hodnoty parametrů PAO. Aktualizuje záznamy PAO v Neo4J (kapacita, dostupnost), všude, kde došlo ke změně (nedostupnost, změna kapacity).
8. Na závěr zobrazí overseer shrnutí operací v dashboardu (počet success/fail, blokací/odblokací ..).

Logování

Software loguje následující operace:

- úspěch/neúspěch u automaticky spouštěných akcí (kontrola živosti, konfigurací, ..)
- úspěch/neúspěch u manuálně spuštěných operací (přicházející z dashboardu, ..)
- každou operaci na prvku aktivní obrany
- chybové stavy a selhání komunikace

Události jsou ukládány v následujících souborech:

- /var/log/crusoe/act_overseer.log
    - běh služby monitorující PAO
- /var/log/crusoe/act_overseer_rest_api.log
    - příkazy zadané skrze rozhraní služby act_overseer
- /var/log/crusoe/act_decide_to_act.log
    - běh služby automatické konfigurace PAO na základě konfigurací doporučených software fáze Decide

Formát logu:
Každá zpráva je tvořena čtyřmi parametry oddělenými mezerou.

- date datum zachycení události
- time čas zachycení události
- severity úroveň závažnosti události {info, warning, error}

- message zpráva popisující událost

Příklad události:

2020-12-08 13:10:43,845 INFO Security treshold value changed: 50 -> 60

Balíček act-overseer

Act-overseer se skládá ze 4 modulů - act_overseer_config, act_overseer_rest_api, act_to_neo4j, a decide_to_act. Funkce act-overseer jsou tedy dále rozděleny podle modulu, ke kterému náleží.

Modul act_overseer_config

Konfigurační soubor act_overseer_config obsahuje:
"security_treshold" - Hodnota bezpečnostního hodnocení, která se dá číst a měnit pomocí dotazů na act_overseer_rest_api.
"log_path" - Cesta k adresáři, ve kterém se budou ukládat logy komponenty.
"user" - Uživatelské jméno pro přístup k databázovému REST API.
"password" - Heslo pro přístup k databázovému REST API.
"server_url" - URL serveru s databází.

Modul act_overseer_rest_api

Přístupové rozhraní act_overseer_rest_api umožňuje ovládat a provádět nastavení funkcionality balíčku act-overseer. REST API je vytvořeno pomocí Django frameworku. Má tři endpointy:

/protect_missions_assets - Tato funkce slouží k zahájení konfigurace PAO na základě konfigurací doporučených software fáze Decide. Očekává POST data v následujícím tvaru:

```
[
   {
      "name": "jméno mise č.1",
      "config_id": 2
   },
   {
      "name":”jméno mise č.2",
      "config_id": 1
   }
]
```

Pokud jsou data vyhodnocena jako validní, zavolá se hlavní funkce modulu decide_to_act, popsaná níže, a proběhne konfigurace PAO.

/treshold - Akceptuje dva dotazy, GET a PUT. Pomocí GETu je možné aktuální hodnotu bezpečnostního hodnocení přečíst, pomocí PUTu zapisovat. Formát dat pro PUT je následující:

```
{
   “security_treshold”: 50
```

}

/log - Pomocí endpointu log je možné získat posledních 100 zpráv v logu act-overseer zobrazovaného na dashboardu.

Modul act_to_neo4j

Modul act_to_neo4j zajišťuje komunikaci mezi balíčkem act-overseer a databází Neo4j. Udržuje v databázi aktuální data prvků aktivní obrany - IP adresu, port, kapacitu, a živost. Spouští se periodicky, nebo také voláním z decide_to_act modulu pro potřebu aktualizace nových údajů do databáze. Periodu spouštění je možné nastavit v konfiguraci organizační služby Celery. Modul má tyto funkce:

filename(name)

Vrací absolutní cestu ke zdroji předaném v argumentu ‘name’. Software act-overseer tuto funkci využívá pro lokalizaci certifikačního souboru v adresáři ‘data’ v balíku act-overseer.
Parametry:

- **name**: Název zdroje.

get_ip_and_port(pao, wrappers)

Vrací IP a port daného PAO wrapperu.
Paramtery:

- **pao**: Název prvku aktivní obrany.
- **wrappers**: Seznam všech wrapperů prvků aktivní obrany.

get_paos(user, passwd, server_url, logger)

Vrací JSON se všemi kontaktními údaji (IP, port) na dostupné prvky aktivní obrany. Při neúspěšném volání databázového REST API vrací None.
Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd:** Heslo pro přístup k databázovému REST API.
- **server_url:** URL serveru, na kterém act-overseer běží.
- **logger:** Soubor pro logování.

update_last_contact(pao, user, passwd, server_url, logger)

Funkce aktualizuje čas posledního kontaktu s wrapperem prvku aktivní obrany. Vrací objekt typu Response. Při neúspěšném volání databázového REST API vrací None.
Parametry:

- **pao:** Název prvku aktivní obrany.
- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd:** Heslo pro přístup k databázovému REST API.
- **server_url:** URL serveru, na kterém act-overseer běží.
- **logger:** Soubor pro logování.

check_and_update(pao, user, passwd, server_url, logger)

Funkce zjistí živost daného prvku aktivní obrany a následně ji aktualizuje v databázi pomocí funkce **update_last_contact()**. Vrací informaci, zda se povedlo aktualizovat živost v databázi.

Parametry:

- **pao:** Název prvku aktivní obrany.
- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd:** Heslo pro přístup k databázovému REST API.
- **server_url:** URL serveru, na kterém act-overseer běží.
- **logger:** Soubor pro logování.

update_capacity(pao, user, passwd, capacity_type, capacity_value, server_url, logger)

Funkce v databázi aktualizuje daný typ kapacity prvku aktivní obrany. Vrací objekt typu Response. Při neúspěšném volání databázového REST API vrací None.

Parametry:

- **pao:** Název prvku aktivní obrany.
- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd:** Heslo pro přístup k databázovému REST API.
- **capacity_type:** Typ kapacity (jedno z ‘maxCapacity’, ‘usedCapacity’ a ‘freeCapacity’).
- **capacity_value:** Hodnota dané kapacity.
- **server_url:** URL serveru, na kterém act-overseer běží.
- **logger:** Soubor pro logování.

retrieve_and_update_capacity(pao, user, passwd, server_url, logger)

Funkce zjistí kapacity daného prvku aktivní obrany a aktualizuje je v databázi pomocí funkce **update_capacity()** Vrací počet úspěšně aktualizovaných kapacit.

Parametry:

- **pao:** Název prvku aktivní obrany.
- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd:** Heslo pro přístup k databázovému REST API.
- **server_url:** URL serveru, na kterém act-overseer běží.
- **logger:** Soubor pro logování.

update_db(user, passwd, server_url, logger=structlog.get_logger())

Hlavní funkce modulu act_to_neo4j. Funkce pomocí výše popsaných funkcí aktualizuje živost a kapacity pro všechny prvky aktivní obrany. Vrací řetězec s informací o počtu úspěšně aktualizovaných živostí a kapacit prvků aktivní obrany.

Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd**: Heslo pro přístup k databázovému REST API.
- **server_url:** URL serveru, na kterém act-overseer běží.
- **logger:** Soubor pro logování. Liší se při periodickém volání a při volání z modulu decide_to_act.

Modul decide_to_act

Modul decide_to_act obsahuje logiku zodpovědnou za nastavení PAO podle konfigurací software fáze Decide. Modul obsahuje funkce na blokování a odblokování IP adres na prvku aktivní obrany a dále funkce, které rozhodují, které IP adresy je potřeba blokovat a které naopak odblokovat. Tyto jsou popsány níže:

get_missions(user, passw, logger, dashboard_log, server_url)

Funkce získá seznam všech misí pomocí databázového REST API. Ten následně taky vrátí.
V případě neúspěšného volání REST API vyhazuje výjimku.

Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passw:** Heslo pro přístup k databázovému REST API.
- **logger:** Soubor pro logování.
- **dashboard_log**: Log zobrazovaný na dashboardu.
- **server_url:** URL serveru, na kterém act-overseer běží.

get_configurations(user, passw, mission, logger, dashboard_log, server_url)

Funkce získá seznam konfigurací pro danou misi. Tento seznam následně vrací. Pokud se funkci nepovede kontaktovat databázové REST API, vyhazuje výjimku.

Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passw:** Heslo pro přístup k databázovému REST API.
- **mission:** Jméno mise, které přísluší vrácené konfigurace.
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **server_url:** URL serveru, na kterém act-overseer běží.

get_fw_ip_and_port(user, passw, logger, server_url)

Funkce vrací IP a port firewall wrapperu.

Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passwd:** Heslo pro přístup k databázovému REST API.
- **logger:** Soubor pro logování.
- **server_url:** URL serveru, na kterém act-overseer běží.

get_hosts(user, passw, logger, dashboard_log, server_url)

Funkce získá seznam všech zařízení v databázi (identifikovaných IP adresou) pomocí databázového REST API. Tento seznam následně vrací. V případě neúspěšného volání REST API vyhazuje výjimku.

Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passw:** Heslo pro přístup k databázovému REST API.
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.

- **server_url:** URL serveru, na kterém act-overseer běží.

get_important_mission_hosts(user, passw, mission, configuration, logger, dashboard_log, server_url)

Funkce vrací seznam všech zařízení, která jsou kritická pro běh dané mise v dané konfiguraci.
Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passw:** Heslo pro přístup k databázovému REST API.
- **mission:** Jméno mise, které přísluší vrácené konfigurace.
- **configuration:** ID konfigurace.
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **server_url:** URL serveru, na kterém act-overseer běží.

potential_blocking(user, passw, missions_and_configurations, logger, dashboard_log, server_url)

Funkce porovnává seznam všech zařízení se seznamy kritických zařízení důležité hosty pro dané páry mise-konfigurace. Výsledkem jeí seznam potenciálních blokací, t.j. hosty, které nejsou důležité pro žádnou z misí v dané konfiguraci.
Parametry:

- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passw:** Heslo pro přístup k databázovému REST API.
- **missions_and_configurations:** Seznam dvojic (mise, konfigurace).
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **server_url:** URL serveru, na kterém act-overseer běží.

get_firewall_health(logger, dashboard_log, firewall_ip_and_port)

Funkce z firewall wrapperu zjistí jeho živost a vrací True pokud volání proběhlo úspěšně, jinak False.
Parametry:

- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

get_firewall_capacities(logger, dashboard_log, firewall_ip_and_port)

Funkce z firewall wrapperu zjistí jeho kapacity a tyto kapacity vrací.
Parametry:

- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

get_treshold(logger, dashboard_log)

Funkce vrací hodnotu bezpečnostního tresholdu.
Parametry:

- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.

remove_less_than_treshold(potential_blockings, logger, dashboard_log)

Funkce vrací nový seznam blokací, získaný odstraněním hostů ze seznamu potenciálních blokací ‘potential_blockings’, které mají bezpečnostní hodnocení menší než bezpečnostní hodnocení treshold.

Parametry:

- **potential_blockings:** Seznam potenciálních blokací.
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.

get_average_security_value(host, logger)

Funkce vrací průměrnou bezpečnostní hodnotu ze tří hodnot v hostovi:
availability, confidentiality, integrity.

Parametry:

- **host:** Slovník reprezentující hosta.
- **logger:** Soubor pro logování.

get_blocked_ips(logger, dashboard_log, firewall_ip_and_port)

Funkce získá z firewall wrapperu seznam blokovaných IP adres. Tento seznam následně vrací. V případě neúspěšného volání firewall wrapperu vyhazuje výjimku.

Parametry:

- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

is_already_blocked(ip, firewall_ip_and_port)

Funkce zkontroluje, zda-li je IP na firewallu již blokována.

Parametry:

- **ip:** IP, o které chceme zjistit, jestli je na firewallu blokována.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

block_ip(ip, logger, dashboard_log, firewall_ip_and_port)

Funkce zablokuje IP adresu na firewallu.

Parametry:

- **ip:** IP, kterou chceme zablokovat.
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

unblock_ip(ip, logger, dashboard_log, firewall_ip_and_port)

Funkce odblokuje IP adresu na firewallu.

Parametry:

- **ip:** IP, kterou chceme zablokovat.

- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

unblock_list(blocked_ips_list, to_block_list)

Funkce vrací seznam IP adres, které budou na firewallu odblokovány.
Parametry:
- **blocked_ips_list:** Seznam blokovaných adres.
- **to_block_list:** Seznam nových blokací.

block_list(to_block_list, blocked_ips_list)

Funkce vrací seznam IP adres, které by měly být na firewallu blokovány.
Parametry:
- **to_block_list:** Seznam nových blokací.
- **blocked_ips_list:** Seznam blokovaných adres.

can_unblocks_and_blocks_be_performed(blocked_ips, list_of_new_blockings, logger, dashboard_log, firewall_ip_and_port)

Funkce zjistí, zda-li odblokace a blokace mohou být na firewallu provedeny, tedy je-li dostupná dostatečná kapacita a prvek je kontaktní. Vrací True, pokud ano, False jinak.
Parametry:
- **blocked_ips:** Seznam blokovaných adres.
- **list_of_new_blockings:** Seznam nových blokací.
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **firewall_ip_and_port:** IP adresa a port firewall wrapperu.

run_decide_to_act(user, passw, missions_and_configurations, logger, dashboard_log, server_url)

Hlavní funkce modulu decide_to_act, která je spouštěna z REST API act-overseera po aplikovaní vybraných konfigurací pro mise v dashboardu. Vrací informaci o počtu odblokovaných a blokovaných IP adres na firewallu.
Parametry:
- **user:** Uživatelské jméno pro přístup k databázovému REST API.
- **passw:** Heslo pro přístup k databázovému REST API.
- **missions_and_configurations:** Seznam dvojic (mise, konfigurace).
- **logger:** Soubor pro logování.
- **dashboard_log:** Log zobrazovaný na dashboardu.
- **server_url:** URL serveru, na kterém act-overseer běží.

Balíček act-component

Balíček obsahuje sadu wrapperů pro 5 definovaných kategorií prvků aktivní obrany. Každý wrapper definuje množinu funkcí pro správu prvku. Všechny definované funkce jsou nezbytné pro připojení prvku software fáze Act. Wrappery neobsahují logickou část, kvůli

zachování obecnosti a možnosti nasazení do různých prostředí s různě implementovanými prvky aktivní obrany. Proto je nutné před nasazením tuto logiku doplnit a napojit wrapper na existující přístupové rozhraní prvku. Detailní specifikace funkcí wrapperu je uvedena v Příloze A.

Kromě pěti wrapperů pro PAO byl vyvinut i simulovaný firewall pro testování projektového software bez ovlivnění reálné konfigurace sítě. Nachází se ve složce /act-component/simulated-pao-firewall. Simulovaný firewall je představován datovou strukturou, která obsahuje dva definované parametry – maximální kapacitu a blacklist. Maximální kapacita ohraničuje maximální možný počet položek blacklistu. Blacklist obsahuje seznam aktuálně simulovaně blokovaných IP adres, spolu s doprovodnými informacemi definovanými ve wrapperu. Dále má prvek implementovány všechny funkce definované svým wrapperem. Tento prvek je funkčním celkem připraveným pro nasazení, kombinuje funkce wrapperu, přístupového rozhraní, i samotného PAO firewall.

Příloha A

Příloha obsahuje podrobné specifikace funkcí wrapperů, včetně definice datových typů a formátu vstupních a výstupních parametrů. Specifikace jsou popsány v jazyce YAML a jsou kompatibilní se standardem OpenAPI ve verzi 3.0.0.

DNS FW API

```
openapi: 3.0.0
info:
  version: v1
  title: CRUSOE Act API Wrapper for DNS Firewall
  description: API wrapper for Active Network Defense devices of project CRUSOE
servers:
  - description: SwaggerHub API Auto Mocking
    url: 'https://virtserver.swaggerhub.com/MadGeckoo/act/v1'
paths:
  '/dnsfw/health':
    get:
      description: Returns a health check for the DNS FW
      responses:
        '200':
          description: Successfully returned health check
          content:
            application/json:
              schema:
                type: object
                properties:
                  serviceStatus:
                    type: string
        '400':
          description: Invalid request
        '503':
          description: Service unavailable
  '/dnsfw/capacity':
    get:
      description: Returns DNS FW capacities
      responses:
        '200':
          description: Successfully returned current capacities
          content:
            application/json:
              schema:
                type: object
                properties:
                  maxCapacity:
                    type: integer
                  usedCapacity:
                    type: integer
                  freeCapacity:
```

```
              type: integer
    '400':
      description: Invalid request
    '403':
      description: Function not supported
'/dnsfw/rules':
  get:
    description: Returns a list of all DNS FW rules
    responses:
      '200':
        description: Successfully returned list of DNS FW rules
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleZone:
                    type: string
                  ruleDomain:
                    type: string
                  ruleTarget:
                    type: string
                  ruleReason:
                    type: string
                  ruleNote:
                    type: string
      '400':
        description: Invalid request
  post:
    description: Add DNS FW rule to a DNS FW
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleZone
              - ruleDomain
              - ruleTarget
            properties:
              ruleZone:
                type: string
              ruleDomain:
                type: string
              ruleTarget:
                type: string
```

```
            ruleReason:
              type: string
            ruleNote:
              type: string
    responses:
      '200':
        description: Return ID of the added rule
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
      '400':
        description: Invalid request
  '/dnsfw/rules/{ruleId}':
    get:
      description: Get details of a rule with ruleID from a DNS FW
      parameters:
        - name: ruleId
          in: path
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: Here are the details
          content:
            application/json:
              schema:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleZone:
                    type: string
                  ruleDomain:
                    type: string
                  ruleTarget:
                    type: string
                  ruleReason:
                    type: string
                  ruleNote:
                    type: string
        '400':
          description: Invalid request
    put:
      description: Change a reason, or zone or target for a rule with ruleID from a DNS FW
      parameters:
        - name: ruleId
```

```
    in: path
    required: true
    schema:
      type: integer
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          properties:
            ruleZone:
              type: string
            ruleTarget:
              type: string
            ruleReason:
              type: string
            ruleNote:
              type: string
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            type: object
            properties:
              ruleId:
                type: integer
              ruleZone:
                type: string
              ruleDomain:
                type: string
              ruleTarget:
                type: string
              ruleReason:
                type: string
              ruleNote:
                type: string
    '400':
      description: Invalid request
delete:
  description: Delete a rule with ruleId from a DNS FW
  parameters:
    - name: ruleId
      in: path
      required: true
      schema:
        type: integer
  responses:
    '200':
```

```
      description: Rule with the ruleId deleted
      content:
        application/json:
          schema:
            type: object
            properties:
              ruleId:
                type: integer
              ruleZone:
                type: string
              ruleDomain:
                type: string
              ruleTarget:
                type: string
              ruleReason:
                type: string
              ruleNote:
                type: string
    '400':
      description: Invalid request
'/dnsfw/{ruleDomain}':
  get:
    description: Get all rules with the specified ruleDomain from a DNS FW
    parameters:
      - name: ruleDomain
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Successfully returned list of rules for the specified ruleDomain
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleZone:
                    type: string
                  ruleDomain:
                    type: string
                  ruleTarget:
                    type: string
                  ruleReason:
                    type: string
                  ruleNote:
                    type: string
```

```
    '400':
      description: Invalid request
    '404':
      description: Domain not found
  put:
    description: Change a zone or target or reason or note for all rules with the specified ruleDomain
    parameters:
      - name: ruleDomain
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              ruleZone:
                type: string
              ruleTarget:
                type: string
              ruleReason:
                type: string
              ruleNote:
                type: string
    responses:
      '200':
        description: All rules for the ruleDomain were changed
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleZone:
                    type: string
                  ruleDomain:
                    type: string
                  ruleTarget:
                    type: string
                  ruleReason:
                    type: string
                  ruleNote:
                    type: string
      '400':
        description: Invalid request
```

```
    '404':
      description: Domain not found
  delete:
    description: Delete all rules for the specified ruleDomain
    parameters:
      - name: ruleDomain
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: All rules for the ruleDomain deleted
      '400':
        description: Invalid request
      '404':
        description: Domain not found
```

Wrapper FW

```
openapi: 3.0.0
info:
  version: v1
  title: CRUSOE Act API Wrapper for firewall
  description: API wrapper for Active Network Defense devices of project CRUSOE. Firewall blcks
access to the internal services from the outside network.
servers:
  - description: SwaggerHub API Auto Mocking
    url: 'https://virtserver.swaggerhub.com/MadGeckoo/act/v1'
paths:
  '/firewall/health':
    get:
      description: Returns a health check for firewall
      responses:
        '200':
          description: Successfully returned health check
          content:
            application/json:
              schema:
                type: object
                properties:
                  serviceStatus:
                    type: string
        '400':
          description: Invalid request
        '503':
          description: Service unavailable
  '/firewall/capacity':
    get:
      description: Returns capacities for firewall
      responses:
```

```
    '200':
      description: Successfully returned current capacities
      content:
        application/json:
          schema:
            type: object
            properties:
              maxCapacity:
                type: integer
              usedCapacity:
                type: integer
              freeCapacity:
                type: integer
    '400':
      description: Invalid request
    '403':
      description: Function not supported
'/firewall/blocked':
  get:
    description: Returns a list of blocked IP adresses for firewall
    responses:
      '200':
        description: Successfully returned list of blocked IPs
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleIp:
                    type: string
                  rulePort:
                    type: integer
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
  post:
    description: Block an IP on firewall, port is optional
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleIp
            properties:
```

```
            ruleIp:
              type: string
            rulePort:
              type: integer
            ruleReason:
              type: string
    responses:
      '200':
        description: Return ID of a given IP address block rule
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
      '400':
        description: Invalid request
  '/firewall/blocked/{blockedId}':
    get:
      description: Get details of a rule with ruleID from firewall
      parameters:
        - name: blockedId
          in: path
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: Here are the details
          content:
            application/json:
              schema:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleIp:
                    type: string
                  rulePort:
                    type: integer
                  ruleReason:
                    type: string
        '400':
          description: Invalid request
    delete:
      description: Delete a rule with blockedId from firewall
      parameters:
        - name: blockedId
          in: path
          required: true
```

```
      schema:
        type: integer
    responses:
      '200':
        description: Rule for blockedId deleted
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
                  type: string
                rulePort:
                  type: integer
                ruleReason:
                  type: string
      '400':
        description: Invalid request
  '/firewall/blocked/{blockedId}/port':
    put:
      description: Change port for a rule with ruleID on the firewall
      parameters:
        - name: blockedId
          in: path
          required: true
          schema:
            type: integer
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                rulePort:
                  type: integer
      responses:
        '200':
          description: Return the new entry
          content:
            application/json:
              schema:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleIp:
                    type: string
                  rulePort:
```

```
              type: integer
            ruleReason:
              type: string
    '400':
      description: Invalid request
'/firewall/blocked/{blockedId}/reason':
  put:
    description: Change reason for a rule with ruleID on the firewall
    parameters:
      - name: blockedId
        in: path
        required: true
        schema:
          type: integer
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              ruleReason:
                type: string
    responses:
      '200':
        description: Return the new entry
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
                  type: string
                rulePort:
                  type: integer
                ruleReason:
                  type: string
      '400':
        description: Invalid request
'/firewall/{blockedIp}':
  get:
    description: Get all rules with specified blockedIp on firewall
    parameters:
      - name: blockedIp
        in: path
        required: true
        schema:
          type: string
    responses:
```

```
    '200':
      description: Successfully returned list of rules for given IP
      content:
        application/json:
          schema:
            type: array
            items:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
                  type: string
                rulePort:
                  type: integer
                ruleReason:
                  type: string
    '400':
      description: Invalid request
    '404':
      description: IP not found
  put:
    description: Change the reason for all rules with specified IP
    parameters:
      - name: blockedIp
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleReason
            properties:
              ruleReason:
                type: string
    responses:
      '200':
        description: Reason changed succesfully
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
```

```
                type: integer
              ruleIp:
                type: string
              rulePort:
                type: integer
              ruleReason:
                type: string
      '400':
        description: Invalid request
      '404':
        description: IP not found
  delete:
    description: Delete all rules for IP
    parameters:
      - name: blockedIp
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: All rules for the IP deleted
      '400':
        description: Invalid request
      '404':
        description: IP not found
'/firewall/{blockedIp}/{blockedPort}':
  delete:
    description: Delete a rule containing the IP and port
    parameters:
      - name: blockedIp
        in: path
        required: true
        schema:
          type: integer
      - name: blockedPort
        in: path
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: Rule deleted
      '400':
        description: Invalid request
      '404':
        description: IP or port not found
```


Wrapper Mail Filter


```
openapi: 3.0.0
```

```
info:
  version: v1
  title: CRUSOE Act API Wrapper for Mail Filter
  description: API wrapper for Active Network Defense devices of project CRUSOE
servers:
  - description: SwaggerHub API Auto Mocking
    url: 'https://virtserver.swaggerhub.com/MadGeckoo/act/v1'
paths:
  '/mailFilter/health':
    get:
      description: Returns a health check for mail filter
      responses:
        '200':
          description: Successfully returned health check
          content:
            application/json:
              schema:
                type: object
                properties:
                  serviceStatus:
                    type: string
        '400':
          description: Invalid request
        '503':
          description: Service unavailable
  '/mailFilter/capacity':
    get:
      description: Returns mail filter capacities for mail filter
      responses:
        '200':
          description: Successfully returned current capacities
          content:
            application/json:
              schema:
                type: object
                properties:
                  maxCapacity:
                    type: integer
                  usedCapacity:
                    type: integer
                  freeCapacity:
                    type: integer
        '400':
          description: Invalid request
        '403':
          description: Function not supported
  '/mailFilter/blocked':
    get:
      description: Returns a list of e-mail adresses blocked by mailFilter
      responses:
        '200':
```

```
      description: Successfully returned list of blocked e-mails
      content:
        application/json:
          schema:
            type: array
            items:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleAddress:
                  type: string
                ruleFrom:
                  type: boolean
                ruleTo:
                  type: boolean
                ruleReason:
                  type: string
    '400':
      description: Invalid request
  post:
    description: Block an e-mail address by mailFilter
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleAddress
              - ruleFrom
              - ruleTo
            properties:
              ruleAddress:
                type: string
              ruleFrom:
                type: boolean
              ruleTo:
                type: boolean
              ruleReason:
                type: string
    responses:
      '200':
        description: Return ID of a given e-mail address block rule
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
```

```
    '400':
      description: Invalid request
'/mailFilter/blocked/{ruleId}':
  get:
    description: Get details of a rule with the ruleID from mailFilter
    parameters:
      - name: ruleId
        in: path
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: Here are the details
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleAddress:
                  type: string
                ruleFrom:
                  type: boolean
                ruleTo:
                  type: boolean
                ruleReason:
                  type: string
      '400':
        description: Invalid request
  put:
    description: Change a reason, from and to for a rule with ruleID on the mailFilter
    parameters:
      - name: ruleId
        in: path
        required: true
        schema:
          type: integer
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleFrom
              - ruleTo
              - ruleReason
            properties:
              ruleFrom:
```

```
            type: boolean
          ruleTo:
            type: boolean
          ruleReason:
            type: string
  responses:
    '200':
      description: Changed, return the new entry
      content:
        application/json:
          schema:
            type: object
            properties:
              ruleId:
                type: integer
              ruleAddress:
                type: string
              ruleFrom:
                type: boolean
              ruleTo:
                type: boolean
              ruleReason:
                type: string
    '400':
      description: Invalid request
delete:
  description: Delete a rule with ruleId from the mailFilter
  parameters:
    - name: ruleId
      in: path
      required: true
      schema:
        type: integer
  responses:
    '200':
      description: MailFilter rule with ruleId deleted
      content:
        application/json:
          schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleAddress:
                  type: string
                ruleFrom:
                  type: boolean
                ruleTo:
                  type: boolean
                ruleReason:
                  type: string
```

```
    '400':
      description: Invalid request
'/mailFilter/{ruleAddress}':
  get:
    description: Get all rules with specified ruleAddress
    parameters:
      - name: ruleAddress
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Successfully returned list of rules for given e-mail address
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleAddress:
                    type: string
                  ruleFrom:
                    type: boolean
                  ruleTo:
                    type: boolean
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
      '404':
        description: E-mail address not found
  put:
    description: Change a reason and direction for all rules with the specified e-mail address
    parameters:
      - name: ruleAddress
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleFrom
```

```
            - ruleTo
            - ruleReason
          properties:
            ruleFrom:
              type: boolean
            ruleTo:
              type: boolean
            ruleReason:
              type: string
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleAddress:
                    type: string
                  ruleFrom:
                    type: boolean
                  ruleTo:
                    type: boolean
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
      '404':
        description: e-mail address not found
  delete:
    description: Delete all rules for specified e-mail address and direction (from or to)
    parameters:
      - name: ruleAddress
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: All rules for given e-mail address and direction deleted
      '400':
        description: Invalid request
      '404':
        description: No rules for the given address found
/mailFilter/from:
  get:
    description: Get all rules with specified ruleAddress in 'from' direction
```

```
    responses:
      '200':
        description: Successfully returned list of rules for 'from' direction
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleAddress:
                    type: string
                  ruleFrom:
                    type: boolean
                  ruleTo:
                    type: boolean
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
      '404':
        description: No e-mails blocked in 'from' direction
  /mailFilter/to:
    get:
      description: Get all rules with specified ruleAddress 'to'  direction
      responses:
        '200':
          description: Successfully returned list of rules for 'to' direction
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    ruleId:
                      type: integer
                    ruleAddress:
                      type: string
                    ruleFrom:
                      type: boolean
                    ruleTo:
                      type: boolean
                    ruleReason:
                      type: string
      '400':
        description: Invalid request
      '404':
        description: No e-mails blocked in 'to' direction
```

Wrapper RTBH

```
openapi: 3.0.0
info:
  version: v1
  title: CRUSOE Act API Wrapper for RTBH
  description: API wrapper for Active Network Defense devices of project CRUSOE
servers:
  - description: SwaggerHub API Auto Mocking
    url: 'https://virtserver.swaggerhub.com/MadGeckoo/act/v1'
paths:
  '/rtbh/health':
    get:
      description: Returns a health check for rtbh
      responses:
        '200':
          description: Successfully returned health check
          content:
            application/json:
              schema:
                type: object
                properties:
                  serviceStatus:
                    type: string
        '400':
          description: Invalid request
        '503':
          description: Service unavailable
  '/rtbh/capacity':
    get:
      description: Returns capacities for rtbh
      responses:
        '200':
          description: Successfully returned current capacities
          content:
            application/json:
              schema:
                type: object
                properties:
                  maxCapacity:
                    type: integer
                  usedCapacity:
                    type: integer
                  freeCapacity:
                    type: integer
        '400':
          description: Invalid request
        '403':
          description: Function not supported
  '/rtbh/blocked':
```

```
  get:
    description: Returns a list of blocked IP addresses for RTBH interface
    responses:
      '200':
        description: Successfully returned list of blocked IPs
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleIp:
                    type: string
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
  post:
    description: Block an IP on RTBH interface
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleIp
            properties:
              ruleIp:
                type: string
              ruleReason:
                type: string
    responses:
      '200':
        description: Return ID of a given IP address block rule
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
      '400':
        description: Invalid request
  '/rtbh/blocked/{blockedId}':
    get:
      description: Get details of a rule with ruleID from RTBH
      parameters:
```

```
    - name: blockedId
      in: path
      required: true
      schema:
        type: integer
    responses:
      '200':
        description: Here are the details
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
                  type: string
                ruleReason:
                  type: string
      '400':
        description: Invalid request
  put:
    description: Change a reason for a rule with ruleID on the RTBH
    parameters:
      - name: blockedId
        in: path
        required: true
        schema:
          type: integer
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleReason
            properties:
              ruleReason:
                type: string
    responses:
      '200':
        description: Reason changed, return the new entry
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
```

```
              type: string
            ruleReason:
              type: string
    '400':
      description: Invalid request
  delete:
    description: Delete a rule with blockedId from RTBH
    parameters:
      - name: blockedId
        in: path
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: Rule for blockedId deleted
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
                  type: string
                ruleReason:
                  type: string
      '400':
        description: Invalid request
'/rtbh/{blockedIp}':
  get:
    description: Get all rules with specified blockedIp on RTBH
    parameters:
      - name: blockedIp
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Successfully returned list of rules for given IP
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleIp:
```

```
              type: string
            ruleReason:
              type: string
  '400':
    description: Invalid request
  '404':
    description: IP not found
put:
  description: Change the reason for all rules with specified IP
  parameters:
  - name: blockedIp
    in: path
    required: true
    schema:
      type: string
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          required:
            - ruleReason
          properties:
            ruleReason:
              type: string
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            type: array
            items:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleIp:
                  type: string
                ruleReason:
                  type: string
    '400':
      description: Invalid request
    '404':
      description: IP not found
delete:
  description: Delete all rules for specified IP
  parameters:
  - name: blockedIp
    in: path
```

```
        required: true
        schema:
          type: string
    responses:
      '200':
        description: All rules for the IP deleted
      '400':
        description: Invalid request
      '404':
        description: IP not found
```

Wrapper User Blocker

```
openapi: 3.0.0
info:
  version: v1
  title: CRUSOE Act API Wrapper for User Blocker
  description: API wrapper for Active Network Defense devices of project CRUSOE
servers:
  - description: SwaggerHub API Auto Mocking
    url: 'https://virtserver.swaggerhub.com/MadGeckoo/act/v1'
paths:
  '/userBlock/health':
    get:
      description: Returns a health check for user block interface with given id
      responses:
        '200':
          description: Successfully returned health check
          content:
            application/json:
              schema:
                type: object
                properties:
                  serviceStatus:
                    type: string
        '400':
          description: Invalid request
        '503':
          description: Service unavailable
  '/userBlock/capacity':
    get:
      description: Returns user block capacities for user block interface with given id
      responses:
        '200':
          description: Successfully returned current capacities
          content:
            application/json:
              schema:
                type: object
                properties:
                  maxCapacity:
```

```
              type: integer
            usedCapacity:
              type: integer
            freeCapacity:
              type: integer
    '400':
      description: Invalid request
    '403':
      description: Function not supported
'/userBlock/blocked':
  get:
    description: Returns a list of blocked users for userBlock
    responses:
      '200':
        description: Successfully returned list of blocked users
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleUser:
                    type: string
                  ruleBlockedFrom:
                    type: string
                  ruleBlockedTo:
                    type: string
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
  post:
    description: Block a user by userBlock
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleUser
              - ruleBlockedFrom
              - ruleBlockedTo
              - ruleReason
            properties:
              ruleUser:
                type: string
              ruleBlockedFrom:
```

```
            type: string
          ruleBlockedTo:
            type: string
          ruleReason:
            type: string
    responses:
      '200':
        description: Return ID of a given user block rule
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
      '400':
        description: Invalid request
  '/userBlock/blocked/{ruleId}':
    get:
      description: Get details of a rule with ruleID from userBlock
      parameters:
        - name: ruleId
          in: path
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: Here are the details
          content:
            application/json:
              schema:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleUser:
                    type: string
                  ruleBlockedFrom:
                    type: string
                  ruleBlockedTo:
                    type: string
                  ruleReason:
                    type: string
        '400':
          description: Invalid request
    put:
      description: Change a reason or date of the user block end for a rule with ruleID on the userBlock
      parameters:
        - name: ruleId
          in: path
```

```
      required: true
      schema:
        type: integer
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - ruleTo
              - ruleReason
            properties:
              ruleBlockedTo:
                type: string
              ruleReason:
                type: string
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleUser:
                  type: string
                ruleBlockedFrom:
                  type: string
                ruleBlockedTo:
                  type: string
                ruleReason:
                  type: string
      '400':
        description: Invalid request
  delete:
    description: Delete a rule with ruleId from userBlock
    parameters:
      - name: ruleId
        in: path
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: UserBlock rule with ruleId deleted
        content:
          application/json:
            schema:
```

```
            type: object
            properties:
              ruleId:
                type: integer
              ruleUser:
                type: string
              ruleBlockedFrom:
                type: string
              ruleBlockedTo:
                type: string
              ruleReason:
                type: string
    '400':
      description: Invalid request
'/userBlock/{user}':
  get:
    description: Get all rules for user
    parameters:
      - name: user
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Successfully returned list of rules for given user
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  ruleId:
                    type: integer
                  ruleUser:
                    type: string
                  ruleBlockedFrom:
                    type: string
                  ruleBlockedTo:
                    type: string
                  ruleReason:
                    type: string
      '400':
        description: Invalid request
      '404':
        description: User not found
  put:
    description: Change a reason and expiration date for all rules for specified user
    parameters:
      - name: user
```

```
    in: path
    required: true
    schema:
      type: string
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          properties:
            ruleBlockedTo:
              type: string
            ruleReason:
              type: string
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            type: array
            items:
              type: object
              properties:
                ruleId:
                  type: integer
                ruleUser:
                  type: string
                ruleBlockedFrom:
                  type: string
                ruleBlockedTo:
                  type: string
                ruleReason:
                  type: string
    '400':
      description: Invalid request
    '404':
      description: User not found
delete:
  description: Delete all rules for specified user
  parameters:
    - name: user
      in: path
      required: true
      schema:
        type: string
  responses:
    '200':
      description: All rules for given user deleted
    '400':
```

description: Invalid request
'404':
description: No rules for given user found

Příloha B

Příloha obsahuje podrobnou specifikaci přístupového rozhraní REST API služby act-overseer. Specifikace je popsána v jazyce YAML a je kompatibilní se standardem OpenAPI ve verzi 3.0.0.

```
openapi: 3.0.0
info:
  version: v1
  title: CRUSOE Act API for Act Overseer
  description: API specifying the access points to the ACT phase for other phases
servers:
  - description: SwaggerHub API Auto Mocking
    url: 'https://virtserver.swaggerhub.com/MadGeckoo/act/v1'
paths:
  '/act/protect_missions_assets':
    post:
      description: Act expects to receive a json with the list of the mission-configuration pairs.
All missions are required, with exactly one configuration chosen per mission.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                required:
                  - name
                  - config_id
                properties:
                  name:
                    type: string
                  config_id:
                    type: string
      responses:
        '200':
          description: Applying configurations
        '403':
          description: Mission list does not match the list in neo4j, i.e. is not complete
        '404':
          description: Configuration ID does not exist
        '400':
          description: Invalid request
  '/act/treshold':
```

```
    get:
      description: Returns the current security treshold, devices with lower rating will be
blocked
      responses:
        '200':
          description: Successfully returned treshold
          content:
            application/json:
              schema:
                type: object
                properties:
                  treshold:
                    type: integer
        '400':
          description: Invalid request
    put:
      description: Changes current security treshold to a new one
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                treshold:
                  type: integer
      responses:
        '200':
          description: New treshold set
        '400':
          description: Invalid request
```

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 Stanislav Špaček a Milan Žiaran.