VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

INTERAKTIVNÍ ZOBRAZENÍ VIRTUALIZOVANÉHO PROSTŘEDÍ INSTANCÍ KYBERNETICKÉ ARÉNY

INTERACTIVE DISPLAY OF THE VIRTUALIZED ENVIRONMENT OF CYBER ARENA INSTANCES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Adam Tuček

AUTHOR

VEDOUCÍ PRÁCE — Ing. Tomáš Gerlich

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Adam Tuček ***ID:*** 227855

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Interaktivní zobrazení virtualizovaného prostředí instancí Kybernetické arény

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem práce je návrh a implementace interaktivního zobrazení virtualizovaného prostředí cloudové platformy OpenStack. Navržené řešení bude implementováno formou uživatelsky přívětivé komponenty do cyber range platformy Kybernetické arény, kde budou moci uživatelé vidět topologii virtualizovaného prostředí spuštěné instance. Kromě vizualizace bude umožněna také základní interakce s virtuálním strojem, kterým může být například virtualizovaná linuxová distribuce Kali Linux. Pro implementaci využijte jednu z dostupných JavaScript knihoven pro vizualizaci dat.

DOPORUČENÁ LITERATURA:

[1] STODŮLKA, T.; FUJDIAK, R. Budování Cyber Range platformy s technologií cloud computingu. Brno: Vysoké učení technické v Brně, 2022. 153 s. ISBN: 978-80-214-6064-5.

[2] LINGAYAT, A.; SINGH, A.; NAIK, V.; BADRE, R. R.; GUPTA, A. K. Horizon, a web-based user interface for managing services in openstack: an introspection. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2018. p. 1–6. ISBN: 978-1-5386-4430-0.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Tomáš Gerlich

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá návrhem skriptu pro vizualizaci virtualizovaného prostředí. V teoretické částí je definován pojem platforem cyber range a jejich přínosy v oblasti výuky kybernetické bezpečnosti. Následně jsou popsány technologie využité k realizaci vizualizace a na konec teoretické částí je popsán postup výběru vizualizační knihovny, která byla použita pro vlastní implementaci. V praktické části práce jsou nejprve stanoveny základní požadavky, čemuž následuje grafický návrh výstupu a následně je popsána struktura vizualizovaných dat. Na závěr je popsána samotná implementace a funkcionalita jednotlivých částí vizualizačního skriptu. Výsledkem této práce je vlastní skript, který je schopen vizualizovat topologie sítí a zároveň umožnit základní uživatelskou interakci. Výsledné řešení bakalářské práce bylo integrováno do Kybernetické arény.

KLÍČOVÁ SLOVA

BUTCA, Cyber range, JavaScript, grafová struktura, Vis.js, vizualizace dat

ABSTRACT

The bachelor thesis focuses on the design of a script for visualizing a virtualized environment. In the theoretical part, the concept of cyber range platforms and their benefits in the field of cyber security education are defined. Subsequently, the technologies used to realize the visualization are described. Finally, the theoretical part covers the process of selecting the visualization library used for the actual implementation. In the practical part of the thesis, the initial basic requirements are determined, followed by the graphical design of the output. Next, the structure of the visualized data is described. Finally, the implementation of the individual parts of the visualization script and their functionality are discussed. The outcome of this work is a customized script capable of visualizing network topologies and providing basic user interaction. The resulting solution from the bachelor thesis was integrated into the Cyber Arena.

KEYWORDS

BUTCA, Cyber range, JavaScript, graph structure, Vis.js, data visualisation

TUČEK, Adam. *Interaktivní zobrazení virtualizovaného prostředí instancí Kybernetické arény*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 49 s. Bakalářská práce. Vedoucí práce: Ing. Tomáš Gerlich

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Adam Tuček
VUT ID autora:	227855
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Interaktivní zobrazení virtualizovaného prostředí instancí Kybernetické arény

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Tomáši Gerlichovi a odbornému konzultantovi panu Bc. Willimu Lazarovovi za odborné vedení, čas, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Problematika informační bezpečnosti je v současnosti jedním z nejaktuálnějších témat ve světě moderních technologií. Digitalizace, nedostatečná implementace bezpečnostních prvků a krize způsobená globální pandemií viru COVID-19 vyústily v celosvětový nárůst kyberzločinu o 600% [1]. Je tedy potřeba vzdělávat nejen nové odborníky na informační bezpečnost, ale také současné vývojáře, aby se zranitelnostem předcházelo již ve stádiu vývoje aplikací. Řešení této problematiky mohou nabízet platformy cyber range, jejichž hlavním cílem je vzdělávání a testování simulovaných kybernetických hrozeb, potenciálních následků a možností prevence.

Teoretická část bakalářské práce se zabývá obecně platformami cyber range a jejich architekturou, kde jsou blíže popsány jednotlivé vrstvy a možnosti využití těchto platforem ve výuce. Popsána je také samotná platforma Kybernetická aréna s oficiálním názvem BUTCA (Brno University of Technology Cyber Arena), pro kterou je výstup této práce určen. V další části jsou popsány konkrétní frontendové technologie, které byly použity k vlastní implementaci vizualizace. Dále je provedena analýza JavaScript knihoven pro vizualizaci dat a na jejím základě je zvolena vhodná knihovna, která byla využita v praktické části této práce. Praktická část rozebírá postup návrhu aplikace, ve kterém jsou ustanoveny základní požadavky a vytvořen předběžný grafický návrh. Následně je ustanoven formát dat určených ke zpracování, a nakonec je popsán postup při vývoji vlastního skriptu, jeho funkcionalita, samotný grafický výstup a implementace do platformy BUTCA.

1 Platformy cyber range

CR (cyber range) představují kontrolované, izolované, bezpečné prostředí, které simulují sítě, systémy, aplikace a nástroje spojené simulovaným prostředím internetu. Uplatnění nacházejí v mnoha odvětvích, jako je např. výzkum, vývoj a testování software, testy bezpečnostních procedur v rámci firem nebo při výuce studentů informační bezpečnosti. Platforma samotná je tvořena fyzickým, nebo virtualizovanými komponenty, případně kombinací těchto dvou [2, 3].

1.1 Přínosy platforem cyber range ve výuce

Jedním z nejdůležitějších prvků prevence kybernetických útoků je seznámení se s potenciálními hrozbami a útoky. Poučení o možnostech útoků ukáže uživateli, jak se chovat, aby se potenciálním útokům předešlo, případně jak činit pokud k útoku došlo nebo dochází. Správně poučená osoba pak může jednoduše své znalosti sdílet dále. Platformy cyber range tedy poskytují prostředí, ve kterém lze vytvářet různé scénáře kyberbezbečnostních situací, a tedy převést a ověřit vlastní teoretické znalosti v praxi plně legálním způsobem [2].

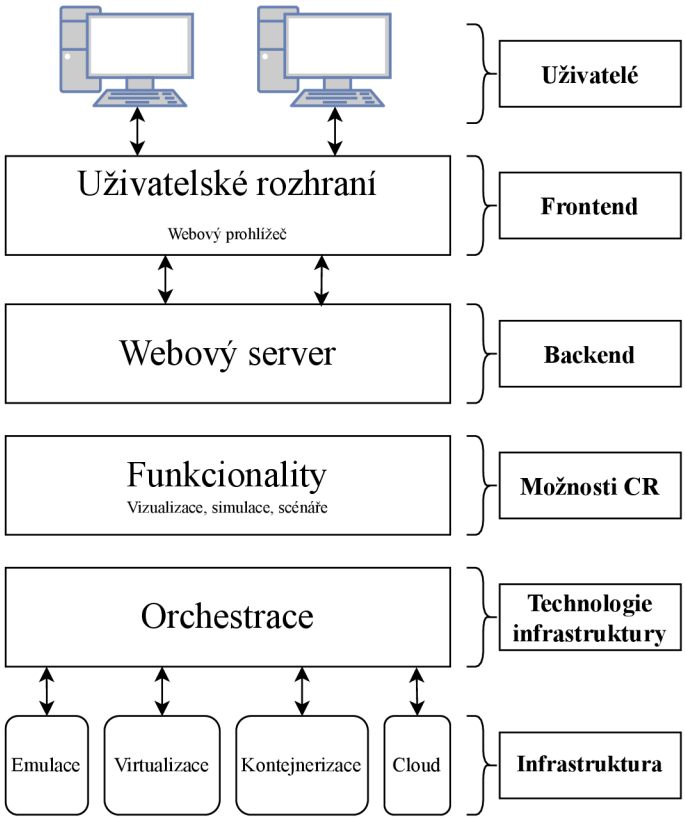
Jedním z nejpopulárnějších médií pro tuto výuku jsou hry CTF (Capture the Flag). V základu jsou založeny na principu školy hrou. Uživateli je problematika prezentována ve formě souvislého příběhu, který jej vede k prohloubení znalostí o konkrétní problematice. Uživatel následuje příběh, plní úkoly a získává kontrolní vlajky, typicky ve formě textu, které ověřují dostatečné pochopení problému. Hlavní výhoda této metody spočívá ve vysoké interaktivitě. Oproti jiným metodám výuky, jako jsou informační videa, jsou hry CTF zábavnější a poutavější [2].

1.2 Architektura

Základní architekturu platformy CR lze najít na obrázku 1.1, musí splňovat několik základních parametrů [2]:

1. Virtualizace komunikační sítě.
2. Virtualizace koncových specifických zařízení SCADA/ICS [1].
3. Prostředí musí být odolné vůči útokům.

[1] SCADA/ICS (Supervisory Control And Data Acquisition/Industrial Control System). SCADA jsou dohledové systémy stojící nad samotným hardware, sloužící ke sběru dat o průmyslových procesech v reálném čase [4]. Pojem ICS označuje kolekci kontrolních systémů sloužících k ovládání průmyslových procesů [5].



Obr. 1.1: Architektura platformy cyber range

1.2.1 Virtualizace

Virtualizace je proces vytváření virtuálních instancí počítačových systémů ve vrstvtách abstrahovaných od skutečného hardware. Fyzický systém se nazývá host, vytvořené virtuální stroje poté hosté. Tyto systémy jsou spravovány tzv. hypervisory, které představují software zajišťující tvorbu, řízení a provoz virtuálních strojů. Hypervisory lze rozdělit na dva druhy:

1. **Bare metal** – hypervisor běží přímo na fyzickém hardware, v podstatě se chovají jako operační systémy.
2. **Hosted** – běží jako tradiční software v operačním systému.

Mezi nejznámější současné hypervisory patří KVM (Kernel-based Virtual Machine), VMware a VirtualBox.

Virtualizace poskytuje řešení pro mnoho problému v oblasti správy serverů. Umožňuje segmentaci větších systému do menších částí, a tedy efektivnější využití zdrojů – samotná výpočetní kapacita, ale také náklady spojené s provozem většího množství serverů, například klimatizování. Virtuální stroje také běží v sandboxovém prostředí, což omezuje jejich schopnost ovlivňovat hostitelský stroj [2, 6].

1.2.2 Kontejnerizace

Na rozdíl od virtualizace při kontejnerizaci nedochází k abstrahování celého systému. Kontejnery virtualizují pouze vlastní operační systém, zabalují veškeré potřebná závislosti konkrétní aplikace pro její správné fungování, pracují izolovaně se sdílenými zdroji a jsou spravovány za pomoci běhových prostředí. Hlavní výhodou je menší náročnost na hardwarové zdroje [2, 7].

1.2.3 Uživatelské rozhraní

Uživatelské rozhraní je element sloužící k propojení uživatele a zbytku platformy CR. Prezentuje uživateli scénář, dle kterého postupuje, zadává úlohy, validuje odpovědi a zpřístupňuje uživateli konkrétní prvky scénáře, jako jsou konzole virtuálních strojů či simulace prostředí. Prezentace probíhá ve webovém prohlížeči, který komunikuje s webovým serverem [2].

1.2.4 Funkcionalita

Funkcionalita představuje vrstvu, která rozšiřuje základní fungování CR o širší možnosti. Umožňuje simulaci internetu, útoků, prostředí nebo sběr dat o uživatelích, jako je doba řešení, jejich úspěšnost a mnoho dalších tzv. user experience[2] funkcí [2].

1.2.5 Orchestrace

Vrstva orchestrace využívá hardware infrastrukturu k vytváření jednotlivých instancí scénářů. Implementuje technologie pro virtualizaci, případně kontejnerizaci a technologie pro vytvoření dostatečného množství prostředí pro všechny uživatele [2].

1.2.6 Infrastruktura

Infrastruktura je nejnižší vrstva architektury, která představuje fyzické nasazení CR. Vytváří se zde jednotlivé instance scénářů. Ty mohou běžet přímo na fyzickém hardware, což je ale složité na nasazení a finančně náročné. Efektivnější je využití kontejnerizace nebo virtualizace, které byly popsány výše. Pro ty je další možné využití cloudových služeb. Zde se pak zabýváme aspekty, jako je plánovaná délka provozu, či náročnost na hardware, podle kterých se poté rozhodneme mezi veřejným (např. Amazon Web Services) nebo privátním (např. OpenStack) cloudem [2].

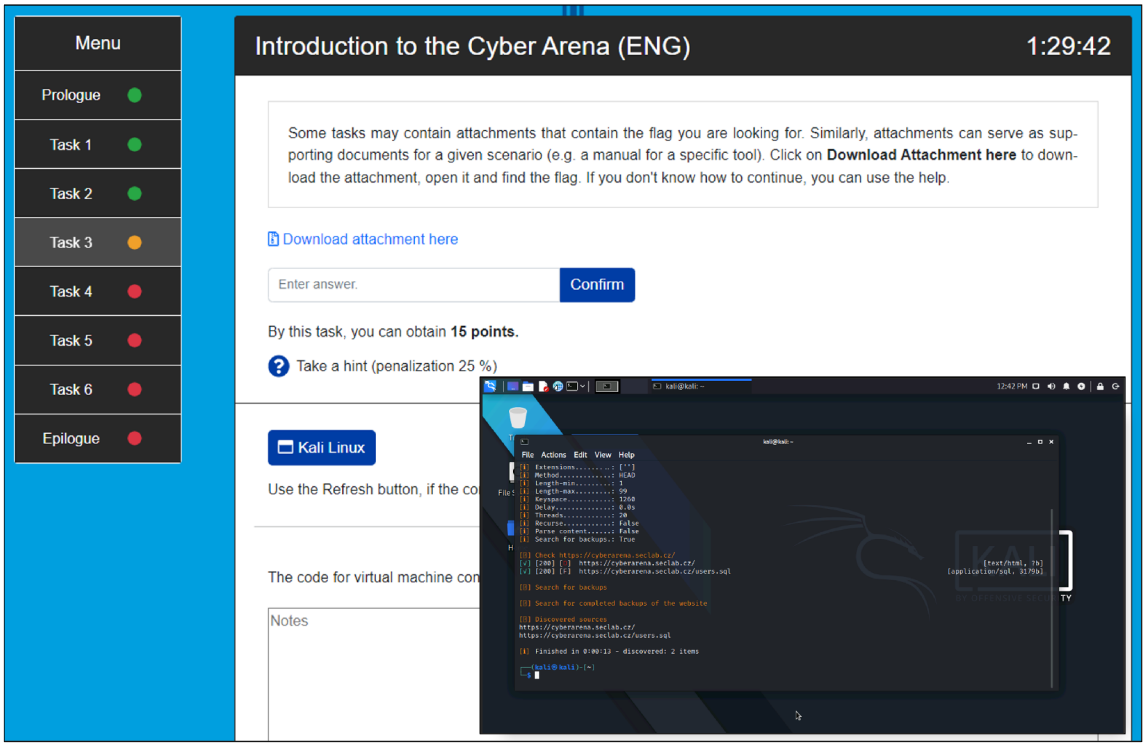
[2]User experience je anglické označení pro funkce, které sice nejsou nutné pro správné fungování, ale obecně zlepšují požitek uživatele.

1.3 Platforma BUTCA

Základem infrastruktury platformy BUTCA[3] je využití cloud computingu, konkrétně platformy OpenStack. Jedná se open source projekt sloužící k tvorbě privátního cloudu, který v rámci platformy zajišťuje virtualizaci, orchestraci a síťování jednotlivých kyberbezpečnostních her. Orchestrace samotná je zajištěna nástrojem OpenStack Heat, který slouží k automatizaci komplexních procesů, a zajišťuje správné vytváření virtuálních prostředí pro jednotlivé hráče.

Další důležitou komponentou je AWX, projekt sponzorovaný firmou Red Hat postavený na automatizačním nástroji Ansible obohaceném o aplikační rozhraní, které umožňuje vzdálenou správu. V platformě BUTCA běží na samotných virtuálních strojích a zajišťuje komunikací mezi OpenStackem a trenovací aplikací.

Třetí částí jsou samotné trénovací aplikace, které kombinují jednotlivá rozhraní pro správu platformy BUTCA a samotných scénářů. Aplikace běží na jednotlivých virutálních strojích nad OpenStackem. Platforma v současnosti disponuje několika kyberbezpečnostními scénáři zaměřenými mj. na síťovou bezpečnost, kryptografii, webovou bezpečnost, reverzní inženýrství, bezpečnost operačních systémů, steganografii a exploitaci [2]. Ukázku konkrétní instance vybraného scénáře v platformě BUTCA lze vidět na obrázku 1.2.



Obr. 1.2: Ukázka instance platformy BUTCA

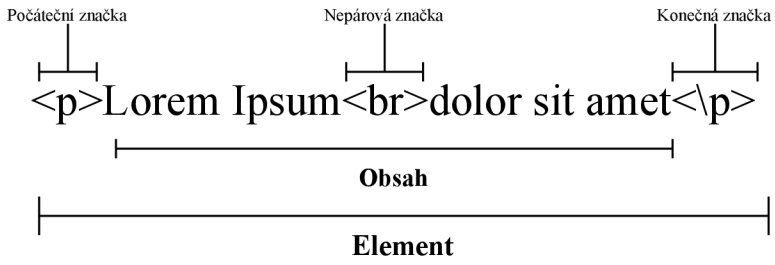
[3]Webová prezentace platformy BUTCA je veřejně dostupná na `https://butca.vut.cz/`.

2 Frontendové technologie

Hlavním cílem této práce je zobrazení vizualizace dat prostřednictvím webového prohlížeče. Na základě tohoto požadavku je tedy nutná práce se současnými webovými standardy. Samotná webová stránka je v rámci této práce vytvořena ve značkovacím jazyce HTML (Hypertext Markup Language) a její parametry jsou dále upraveny pomocí jazyka CSS (Cascading Style Sheets), konkrétně s využitím preprocesoru SASS (Syntactically Awesome Style Sheets). Vizualizace a uživatelské interakce jsou poté vytvořeny s pomocí programovacího jazyku JavaScript a vizualizační knihovny, jež je podrobně rozebrána v další kapitole bakalářské práce.

2.1 HTML

Celým názvem Hypertext Markup Language je standardizovaný značkovací jazyk dokumentů pro zobrazení ve webových prohlížečích. HTML samotné popisuje smysl a strukturu webové stránky jako množinu elementů. Ty jsou definovány počáteční značkou, obsahem a konečnou značkou. Výjimkou jsou tzv. nepárové značky, které konečnou značku nemají. Typ značky přisuzuje elementů určité vlastnosti pro zobrazení (například zda se jedná o nadpis nebo zvýrazněný text). Strukturu elementu lze najít na obrázku 2.1. Tyto elementy vytváří stromovou strukturu, kterou webové prohlížeče přečtou a dle daných značek určují, jak mají být zobrazeny uživateli [8].



Obr. 2.1: Struktura HTML elementu

2.1.1 DOM

DOM (Document Object Model) je standardizovaný stromový model HTML dokumentu, kde každý vrchol je objekt. Slouží jako propojení nezávislé na platformě, či jazyku, mezi webovým dokumentem a skriptovacími jazyky, jako je například JavaScript. Umožňuje také přiřazení event handlerů k jednotlivým vrcholům [9].

2.1.2 Canvas

Canvas je HTML element použitý k vykreslování útvarů. Samotný element slouží pouze jako kontejner, přičemž k vykreslení je použit JavaScript [10]. Nadstavbou Canvasu je WebGL (Web Graphics Library), výkonné JavaScript rozhraní sloužící k vykreslování dvojrozměrných nebo trojrozměrných grafik bez využití pluginů. WebGL je také schopné využít hardwarovou akceleraci uživatelského zařízení [11].

2.1.3 SVG

SVG (Scalable Vector Graphics) je element, který definuje kontejner pro vektorovou grafiku. Narozdíl od Cavasu má samotný element základní funkce pro vykreslování obrazců, a není tedy nutné použít skriptovací jazyk [12].

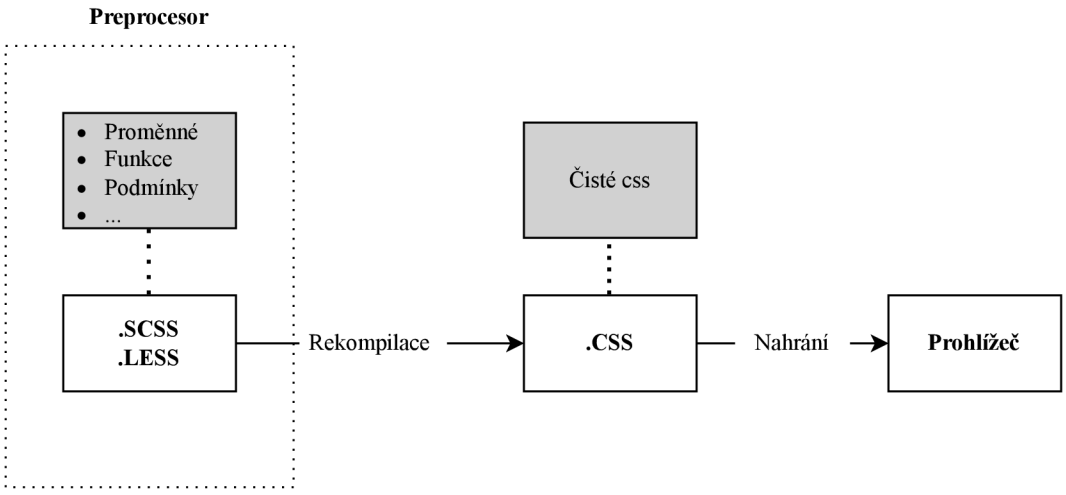
2.2 CSS

Další z webových standardů CSS (česky kaskádové styly) je jazyk pro definici vzhledu webové stránky. CSS lze psát přímo do HTML souboru, typicky je ale oddělován do samostatného souboru a následně přiřazen dokumentům, které má stylovat. CSS má velmi jednoduchou syntaxy, kde v souboru specifikujeme selektor (například `id` konkrétního elementu), který chceme upravovat, a poté mu pomocí deklarací přiřadíme množinu vlastností. Můžeme upravovat například barvu, velikost, styl písma, odsazení textu nebo umístění elementů.

CSS samotné neumožňuje využití klasických programovacích struktur, jako jsou například proměné, funkce, cykly a podmínky. Umožňuje sice jednoduché seskupování selektorů do skupin, přesto ale následkem absence těchto struktur dochází k častému kopírování deklarací vlastností mezi různými selektory a celkově vyšší obtížnosti udržování. Tato problematika je řešena preprocesory [13, 14].

2.2.1 Preprocesory

Jedná se o supersetové jazyky, které rozšiřují CSS o výšše zmíněné chybějící programátorské prvky. Kompilátor preprocesoru konvertuje volání funkcí a proměnné do čistého CSS (viz obrázek 2.2). Umožňují jednoduché a rychlé psaní CSS kódu. Mezi nejpopulárnější preprocessory patří SASS a Less (Leaner Style Sheets) [14].



Obr. 2.2: Umístění preprocesorů ve vztahu k CSS

2.3 JavaScript

Třetím webový standardem je JS (JavaScript). JS je dynamický vysokoúrovňový programovací jazyk, který může být buď interpretovaný, nebo JIT (just-in-time) kompilovaný, tedy spuštěný program je překládán v době provádění, ne před ní. Jazyk podporuje objektový orientovaný, procedurální i funkcionální přístup k programování. JS je známý především jako programovací jazyk pro vývoj webových aplikací, ale lze jej také využít i v jiných newebových prostředích (například Adobe Acrobat). JavaScript běží nejčastěji na klientské straně ve webovém prohlížeči a může určovat pokročilé chování webové aplikace, jako jsou např. události [15].

Standardy JavaScriptu jsou založeny na skriptovacím jazyku ECMAScript, který je vyvíjen společností Ecma International. Standardy jsou pravidelně aktualizovány, přičemž současná verze je 14. edice ECMAScript 2023, a její specifikace mohou být nalezeny v dokumentu ECMA-262 [15, 16].

3 Analýza knihoven pro vizualizaci

V následující kapitole je rozebrán postup výběru knihovny pro praktickou část této práce, který se skládá z metodiky výběru a knihoven, které jsou postupně analyzovány. Nakonec je popsána vybraná knihovna pro vlastní implementaci.

3.1 Metodika výběru knihovny

Pro výběr knihovny byly nastaveny základní parametry, které určují, zda knihovna může být využita pro požadovanou vizualizaci dat:

1. **Open source** – pro použití v praktické části musí být knihovna volně dostupná s otevřeným zdrojovým kódem.
2. **Možnost vykreslování grafů** – knihovny někdy zaměňují „chart“ a „graf“.
3. **Základní funkcionality** – knihovna musí podporovat základní funkcionality pro řešení praktické části. Důležitá je zde především možnost úpravy vzhledu, podpora eventů a možnost zobrazení atributů jednotlivých objektů sítě.
4. **Dokumentace** – knihovna musí být dostatečně zdokumentována.

3.2 Seznam knihoven

Pro analýzu byly vybrány následující knihovny:

1. **D3.js** – všestranná knihovna sloužící k dynamické, interaktivní vizualizaci dat. Knihovna nabízí široké možnosti stylů vizualizace, která je ale složitější než u více specializovaných knihoven. K vykreslení používá SVG [17].
2. **Cytoscape.js** – knihovna typicky využívána v bioinformatice k vizualizaci komplexních sítí a jejich integraci s atributy [18, 19].
3. **Vis.js** – dynamická, modulární knihovna, význačná především svojí jednoduchostí. Výstup knihovna vykresluje na canvas [20].
4. **Keylines** – komerční knihovna, populární pro svoji výkonnost a množství funkcí pro vykreslování grafů na canvas [21].
5. **Sigma.js** – specializovaná knihovna, která staví na algoritmech popsaných v knihovně graphology. Významná je využitím WebGL pro výstup, což umožňuje rychlejší vykreslení většího množství vrcholů. [22, 23].
6. **VivaGraph.js** – knihovna cílíci na rozšiřitelnost a přizpůsobitelnost. Vnitřní alogirtmy pro vytváření grafů jsou definovány v knihovně ngraph [24].
7. **Chart.js** – jednoduchá, flexibilní knihovna pro vizualizaci dat na canvas [25].
8. **Two.js** – knihovna pro vizualizaci dvojrozměrných diagramů [26].
9. **Go.js** – flexibilní knihovna pro vytváření různých interaktivních diagramů s automatickými layouty a řadou dalších funkcí pro jejich snadné vytvaření [27].

10. **ApexCharts.js** – knihovna pro vizualizaci grafů s důrazem na velikost data setu a přizpůsobení vizuální prezentace [28].
11. **Canvas.js** – knihovna s jednoduchým rozhraním, nabízí až 30 typů grafů [29].
12. **Dygraphs.js** – rychlá, nativně interaktivní a flexibilní knihovna, umožňuje interpretaci velkého množství dat [30].
13. **Highcharts.js** – knihovna pracuje s formátem SVG, umožňuje vytvoření širokého množství interaktivních, přístupných grafů [31].
14. **Ogma.js** – knihovna využívající modulární architekturu a WebGL. Umožňuje vykreslení až stovky tisíc vrcholů [32].
15. **yFiles** – komerční knihovna nabízí široké množství diagramů a až 100 typů rozložení vizualizovaných dat [33].

3.3 Výběr knihoven

Na základě stanovené metodiky v podkapitole 3.1 byl proveden důkladný výběr vhodných knihoven pro realizaci semestrální práce, který je uveden v tabulce 3.1.

Tab. 3.1: Výběr knihovny na základě stanovených parametrů

Název knihovny	Open source	Vizualizace grafu	Funkcionality	Dokumentace
D3.js	✓	✓	✓	✓
Chart.js	✓	✗	✗	✓
Cytoscape.js	✓	✓	✓	✓
Keylines	✗	✓	✓	✓
Sigma.js	✓	✓	✓	✗
VivaGraph.js	✓	✗	✓	✗
Two.js	✓	✗	✗	✓
Vis.js	✓	✓	✓	✓
Go.js	✗	✓	✓	✓
ApexCharts.js	✓	✗	✗	✓
Canvas.js	✗	✗	✗	✓
Dygraphs.js	✓	✗	✗	✓
Highcharts.js	✓	✗	✗	✓
Ogma.js	✗	✓	✓	✓
yFiles	✗	✓	✓	✓

Z tabulky vyplývá, že knihovny, které splňují všechny stanovené podmínky pro zpracování praktické části jsou **D3.js**, **Cytoscape.js** a **Vis.js**.

3.3.1 D3.js

D3 je JavaScript knihovna umožňující manipulaci HTML prvků na základě dat. Knihovna je založena na známých webových standardech, jako je HTML, SVG nebo CSS a nevyžaduje tedy užívání jiných frameworků. D3 umožňuje vázání dat na DOM a jejich následnou selekci a manipulaci pomocí funkcí. Knihovna samotná se skládá z oficiálních a komunitních modulů, které lze využívat nezávisle na sobě. Zároveň je také velice lehká, rychlá a podporuje práci s velkým data setem. Nativně knihovna podporuje nahrávání dat z formátu JSON (JavaScript Object Notation), CSV (Comma-separated values), TSV (Tab-Separated Values) a XML (Extensible Markup Language). Efektivní manipulace dat dělá z D3 jednu z nejflexibilnějších dostupných knihoven [17].

3.3.2 Vis.js

Vis.js je dynamická vizualizační knihovna pro webové prohlížeče. Knihovna samotná se skládá z 5 komponent:

- **Data Set**, který umožňuje správu nestrukturovaných dat a vytváření dvoucestné vazby mezi daty a vizualizací. Ukládá data z formátu JSON jako objekty. Ty poté můžou být dynamicky upravovány, přidávány nebo odstraňovány. Vizualizace která s tímto Data Setem pracuje pak tyto změny živě zobrazuje.
- **Graph3D**, **Graph2D** a **Timeline** pro vizualizaci hranových, sloupcových grafů, korelačních diagramů a časových os.
- **A Network**, modul umožňující vizualizaci sítí skládající se z hran a vrcholů. Modul umožňuje interaktivní konfiguraci a filtrování, přizpůsobení hran a vrcholů, seskupování vrcholů a jejich následné přizpůsobení dle dané skupiny, interakci s vrcholy a hranami, manipulaci dat, výběr metody seskupení vrcholů a fyzikální simulace při pohybu s vrcholy.

Při vizualizaci uživatel definuje data, kontejner (oblast kde bude graf vykreslen) a nastavení, ve kterém uživatel určuje a upravuje parametry zobrazení, jako je například barva, velikost a vzhled vrcholů a hran, nebo fyzikální interakce [20].

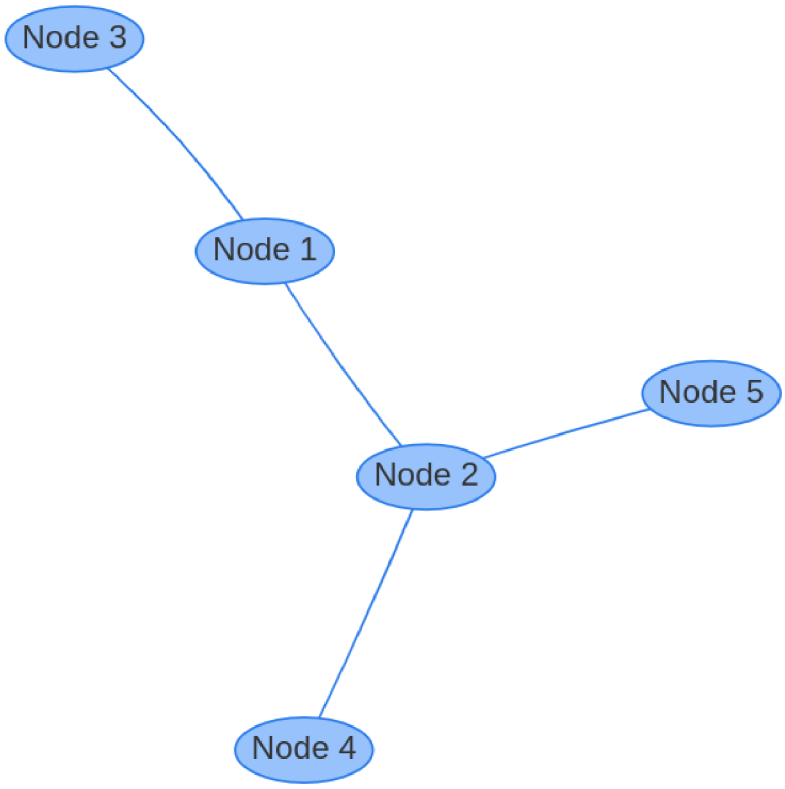
3.3.3 Cytoscape.js

Knihovna původně určena k biologickému vývoji, dnes je využívána jako platforma pro vizualizaci komplexních dat a jejich analýzu. Existuje jako sesterský projekt k desktopové aplikaci napsané v jazyce Java. Cytoscape je optimalizovaný a schopný vizualizace velkého množství dat. K vytvoření vizualizace používá layout, který může

být vytvořen manuálně, nebo automaticky a CSS, ve kterém je definován způsob zobrazení, čímž odděluje vizuální prezentaci od dat [18, 19].

3.4 Knihovna vybraná pro zpracování vizualizace

K práci byla vybrána knihovna Vis.js. Hlavním důvodem k volbě byly vestavěné možnosti knihovny, které splňují požadavky stanovené v zadání této práce – je schopna vizualizace topologie sítě, podporuje eventy, přizpůsobení vrcholů, práci s daty ve formátu JSON a k využití není potřeba jiných frameworků. Knihovna je zároveň dobře zdokumentovaná a poměrně populární, je tedy možné na internetu dohledat velké množství diskuzí, ukázek a jiných řešených případů. V rámci této práce jsou využity moduly `Data Set` a `Network`. Základní výstup vizualizace ve Vis.js bez implementace dalších funkcionalit je na obrázku 3.1.



Obr. 3.1: Ukázka základní vizualizace pomocí Vis.js

4 Praktická část

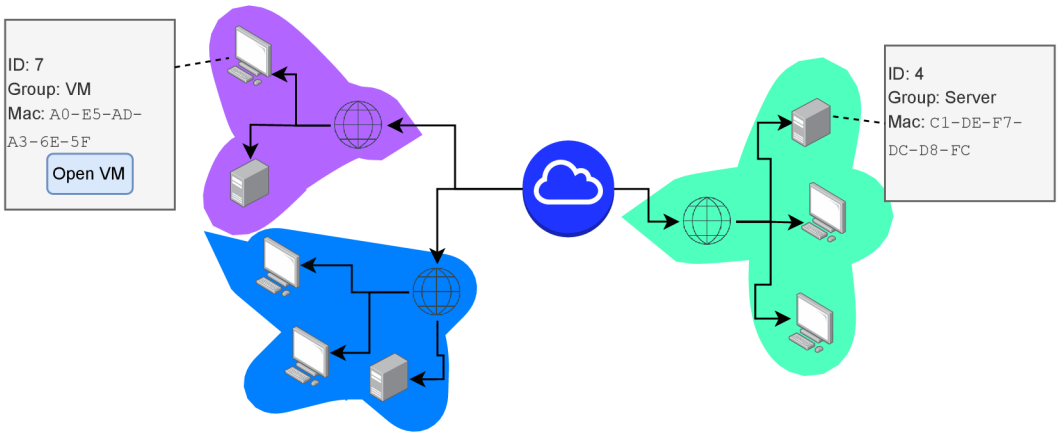
4.1 Návrh prostředí

Hlavním cílem bakalářské práce je návrh skriptu pro vizualizaci dat popisujících topologii sítě. Z plánů k budoucímu nasazení vyplývá několik základních parametrů, které by měly být aplikací splněny:

- **Kvalitní vizuální zpracování** – výsledná vizualizace by měla být graficky přívětivá. Bude jednoznačně popisovat topologii sítě, včetně rozdělení jednotlivých podsítí a grafickou identifikaci typů zařízení.
- **Škálovatelnost** – aplikace musí být schopná vizualizovat data bez zásahu do samotného skriptu. Případné další rozšíření by mělo být jednoduché, intuitivní a nevyžadovat zásahy do funkcí skriptu netýkajících se vytvoření samotné vizualizace. Měla by také mít možnost do vizualizace přidávat zařízení mimo vizualizované virtuální prostředí, jako jsou například webové stránky.
- **Dynamičnost** – aplikace je schopna dynamicky reagovat na změny v datech a přizbůsobit jim vizualizaci.
- **Výkon** – vizualizace velkého množství dat nemá vliv na plynulost.

4.1.1 Grafický návrh

Na základě požadavků stanovených na vizuální zpracování dat byl vytvořen grafický návrh možného řešení (viz obrázek 4.1). V návrhu jsou odděleny jednotlivé uzly a z jejich ikon lze identifikovat reprezentované zařízení. S pomocí podbarvení různými barvami jsou také zaznačeny jednotlivé sítě. Samotné uzly nesou informace, ke kterým lze přistoupit přes kontextové menu, které je přizpůsobeno konkrétnímu typu uzlu (např. identifikátor daného zařízení).



Obr. 4.1: Grafický návrh řešení vizualizace

4.1.2 Statický JSON

V rámci bakalářské práce jsou data určená k vizualizaci dána v souboru ve formátu JSON. Tento soubor (viz výpisy 4.1 a 4.2) lze rozdělit na dvě části. První část `nodes` obsahuje vlastní informace o objektech individuálních uzlů topologie sítě. Lze zde najít unikátní identifikátor (`id`), skupinu, která dále určuje jak bude uzel během vizualizace zpracováván (`group`), unikátní identifikátor sítě, do které uzel zapadá (`netId`), informaci o url adrese zařízení, jenž daný uzel v topologie reprezentuje (`url`) a číslo portu využitého zařízením (`port`).

Výpis 4.1: Výpis uzlů ve formátu JSON

```
1  {
2    "nodes": [
3      {
4        "id": "0",
5        "group": "internet"
6      },
7      {
8        "id": "1",
9        "group": "Network",
10       "netId": "2",
11     },
12     {
13     "id":"3",
14     "name": "Kali,
15     "group":"VM",
16     "netId": "2",
17     "port": "5555"
18     }
19     {
20      "id": "2",
21      "group": "Webpage",
22      "netId": "2",
23      "url": "https://myshop.rocks/L6eBlGUMIB"
24     }
25   ]
26 }
```

Druhá skupina, objekty `edges`, určuje, které uzly topologie budou ve výstupu vizualizace propojeny. K tomuto jsou využity `id` zmíněné výše. Pomocí klíčového slova `from` určíme, ze kterého uzlu bude spoj vycházet a `to` určí, do kterého uzlu bude spoj směrován.

Výpis 4.2: Výpis hran ve formátu JSON

```
{
  "edges": [
    {
      "from": "0",
      "to": "1"
    },
    {
      "from": "0",
      "to": "2"
    },
    {
      "from": "0",
      "to": "3"
    },
    {
    "from":"2",
    "to":"4"
    }
  ]
}
```

4.2 Vývoj skriptu pro vizualizaci

Dle analýzy provedené v kapitole 3 je k vytvoření vizualizace využita knihovna Vis.js. Před implementací vizualizace bylo zřízeno vhodné prostředí. Pro účely testování je aplikace spouštěna ve webovém prohlížeči na stránce lokálního hosta a bylo využito webových standardů popsaných v kapitole 2. Nejprve byl vytvořen HTML dokument (viz výpis 4.3), který je interpretován vlastním webovým prohlížečem. Zároveň do dokumentu byly přidány odkazy na na použitou vizualizační knihovnu, knihovnu Material Design Icons, která poskytuje ikony pro vizualizaci, použitý soubor kaskádových stylů a samotná JavaScript aplikace, která zajišťuje vnitřní logiku tvorby vizualizace a její následné funkcionality. Dále jsou také v sekci `body` vytvořeny elementy pro přepínač barevného režimu `switch`, samotný kontejner pro vizualizaci `net` a okno kontextovéhu menu `pop-up`, které bude zobrazovat informace o jednotlivých uzlech vizualizace.

Výpis 4.3: Dokument HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <script src="/JavaScript/app.js" type="module">
    </script>
    <script src="https://unpkg.com/vis-network/standalone/
    umd/vis-network.min.js"></script>
    <link rel="stylesheet" href="/Styles/style.css">
    <link href="https://fonts.googleapis.com/
    css2?family=Material+Symbols+Outlined"
    rel="stylesheet"/>
    <link href="https://cdn.jsdelivr.net/npm/@mdi/
    font@7.2.96/css/materialdesignicons.min.css"
    rel="stylesheet" />
    <base href="index.html">
</head>

<body>
    </script>
    <label class="switch">
        <input id="checkbox" type="checkbox">
        <span class="slider round"></span>
    </label>
    <span class="material-symbols-outlined">
        dark_mode
    </span>

    <div id="net"> </div>

    <div id="pop-up" >
        <p> Jméno: <span id = name></span></p>
        <p> Skupina: <span id = group_name></span></p>
        <p id="urlPar"> URL <span id="url"></span></p>
        <p id="ipPar"> IP: <span id = "ip"> </span></p>
        <p id="portPar">Port: <span id="port"></span> </p>
        <button id="VMButton">Otevřít</button>
    </div>

</body>
</html>
```

K samotnému vykreslení dochází použitím funkce `vis.Network()`. Ta k vytvoření sítě vyžaduje definici tří parametrů. Prvním parametrem je kontejner pro zobrazení. Vizualizace probíhá na webové stránce, v těle HTML souboru je tedy vytvořen element typu `div` a je mu přiřazen identifikátor `id` (viz výpis 4.3). Tímto byl vytvořen samotný kontejner, protože je ale `div` samotný pouze strukturní element, nemá zatím žádné vlastnosti, jako je velikost nebo barva. S využitím kaskádových stylů budou elementu přiřazeny atributy (viz výpis 4.4). Použitím preprocesoru SASS jsou nejdříve definovány proměnné použité barvy a stylu okraje, což umožňuje opakované použití bez nutnosti nové definice a snadnou hromadnou refaktorizaci, pokud bude barva změněna. Dále je pomocí selektoru `#id` vybrán konkrétní element a jsou mu přiřazeny atributy výšky, šířky, barvy, okraje a pozice. Tímto byla dokončena tvorba kontejneru, pomocí funkce `document.getElementById()` je uložen do proměnné, která následně bude sloužit jako parametr vizualizační funkce.

Výpis 4.4: Nastavení atributů kontejneru

```
1  $backg-color: rgb(255, 255, 255);
2  $popupColor: rgb(255, 255, 255);
3  $border-def:1px solid black;
4
5  #net {
6      width: 700px;
7      height: 550px;
8      background-color: $backg-color;
9      border: $border-def;
10     position: relative;
11
12 }
```

Druhým parametrem vizualizace jsou `data`. Tento parametr je zpracován za pomocí JavaScriptu (viz výpis 4.5). Nejprve je importován soubor ve formátu json. Následně jsou vytvořeny jednotlivé data sety, které umožňuji dvojcestné vázání a tedy dynamické reakce na změny dat. Nakonec je vytvořen samotný objekt dat.

Výpis 4.5: Vytvoření dat

```
1 import jsonData from "./DataSet.json" assert {type:"json"};
2 let nodes = new vis.DataSet(jsonData.nodes);
3 let edges = new vis.DataSet(jsonData.edges);
4 let data = {
5   nodes: nodes,
6   edges: edges,
7 };
```

Třetím a posledním parametrem jsou `options`, objekt představující možnosti vizualizace. Knihovna Vis.js nabízí více než sto uživatelských nastavení. Pro tuto práci jsou nejdůležitější parametry `nodes` a `groups` (viz výpis 4.6). První zmíněný určuje základní parametry všech vrcholů, jejich velikost a barvu. Nastavení `groups` obsahuje parametry pro konkrétní skupiny definovány v kapitole 4.1.2. Nejprve je vybrána skupina a následně jsou přiřazeny vlastnosti. `Shape: "icon"` dále umožní přizpůsobení vrcholů a přiřazení ikon z balíčku Material Design Icons pomocí jejich kódu. Parametry nastavené v `groups` (například barva) zároveň přepisují nastavení provedené v `nodes`. Kompletní použité nastavení vizualizace je dostupné v příloze A.

Výpis 4.6: Nastavení uzlů a skupin vizualizace

```
1  nodes: {
2      shape: "icon",
3      icon: {
4        size: 50,
5        color: "black",
6      },
7  },
8
9   groups: {
10     Internet: {
11       shape: 'icon',
12       icon: {
13         face: 'Material Design Icons',
14         code: '\u{F0163}',
15         color: 'cyan'
16       }
17     },
```

Vytvořením objektu options byly definovány veškeré potřebné parametry. Vizualizace topologie sítě je vytvořena pomocí funkce `vis.Network()` (viz výpis 4.7).

Výpis 4.7: Volání funkce pro vytvoření vizualizace

```
1        let network = new vis.Network(container,
2                                      data,
3                                      options);
```

4.3 Ukázka vizualizace

Samotná vizualizace vytvořená v předchozí části, ale nesplňuje veškeré požadované funkcionality, jako je kontextové menu, či vizuální rozdělení sítí.

Kontextové menu je vytvořeno na podobném principu, jako kontejner vizualizace v minulé kapitole. Nejprve je v HTML souboru definován `div`, v něm jsou vypsány názvy vlastností jako odstavce, a pomocí elementu `<span>` jsou vytvořeny textová pole, kam budou vkládány hodnoty pro zvolený uzel. Element kontextové menu je stylován pomocí kaskádových stylů, je určena jeho velikost, hrana, styl písma, především je ale v základu skryt a automaticky přizpůsobuje svoji velikost zobrazenému obsahu.. Logiku jeho správného zobrazení zajišťuje JavaScript (viz výpis 4.8). Nejprve je definován event `selectNode`, který se automaticky zavolá v případě, že uživatel klikne na uzel a vrátí jej jako parametr. Event zavolá funkci `nodeClickedEvent` a předá jí zvolený uzel jeho původní dataset jako parametr.

Ve funkci je nejprve získán identifikátor zvoleného uzlu a pozice kurzoru v rámci celé stránky, která určí kde bude otevřeno kontextové menu.

Následně jsou přečteny parametry uzlu, které jsou vloženy do textových polí kontextového menu, a v případě že je nastavena hodnota `VMButton` uzlu na `true`, je zobrazeno tlačítko. Vybraný uzel je také zvýrazněn změnou barvy.

Výpis 4.8: Funkce pro otevření kontextového menu

```
1  network.on('selectNode', function (params) {
2    nodeClickedEvent(params, jsonData)
3  })
4
5  function nodeClickedEvent(params, dataset) {
6    let selectedNodeId = params.nodes[0]
7    let cord = window.event
8    let x = cord.pageX
9    let y = cord.pageY
10
11   popUpDiv.style.top = y + 'px'
12   popUpDiv.style.left = x + 'px'
13   popUpDiv.style.display = ''
14   popUpDiv.style.visibility = 'visible'
15   lastNodeClicked= dataset.nodes.find
16   (item => item.id === selectedNodeId)
17   if (
18     dataset.nodes.find(
19     item => item.id === selectedNodeId)
20     .VMButton == 'true'
21   ) {
22     VMButton.style.visibility = 'visible'
23     VMButton.style.display = ''
24   } else {
25     VMButton.style.visibility = 'hidden'
```

```
26      VMButton.style.display = 'none'
27    }
28    popUpNodeName.innerHTML = dataset.nodes.find(
29      item => item.id === selectedNodeId
30    ).name
31    popUpNodeGroup.innerHTML = dataset.nodes.find(
32      item => item.id === selectedNodeId
33    ).group
34    popUpNodeIp.innerHTML = dataset.nodes.find(
35      item => item.id === selectedNodeId
36    ).ip
37    const nodeUrl = dataset.nodes.find(
38    item => item.id === selectedNodeId).url
39    if (nodeUrl == undefined) {
40      urlPar.hidden = true
41    } else {
42      popUpNodeUrl.innerHTML = nodeUrl
43      urlPar.hidden = false
44    }
45    const nodeIp = dataset.nodes.find(
46    item => item.id === selectedNodeId).ip
47    if (nodeIp == undefined) {
48      ipPar.hidden = true
49    } else {
50      popUpNodeIp.innerHTML = nodeIp
51      ipPar.hidden = false
52    }
53    const nodePort = dataset.nodes.find(
54    item => item.id === selectedNodeId).port
55    if (nodePort == undefined) {
56      portPar.hidden = true
57    } else {
58      popUpNodePort.innerHTML = nodePort
59      portPar.hidden = false
60    }
61    let highlightedNode = network.body.nodes[selectedNodeId]
62    highlightedNode.setOptions({
63      icon: {
64        color: 'red'
65      }
66    })
67  }
```

Kontextové menu je uzavřeno pomocí eventu `deselectNode` (4.9), který odpovídá kliknutí mimo uzel, nebo také zvolení jiného uzlu. Kontextové menu a tlačítko pro virtuální stroje jsou skryty.

Výpis 4.9: Uzavření kontextového menu

```
1 network.on('deselectNode', function (params) {
2   popUpDiv.style.display = 'none'
3   VMButton.style.display = 'none'
4 })
```

Druhou funkcí vizualizace je grafické znázornění jednotlivých sítí. K tomu je využit event `afterDrawing` (viz výpis 4.10), který je vyvolán po vytvoření vizualizace a je oproti eventu `beforeDrawing` spuštěn pouze pří interakci s uzly nebo změnou dat. Po vytvoření vizualizace je vytvořená síť stabilizována funkcí `stabilize`, která manuálně vyvolá fyzikální interakci mezi uzly, a zároveň.

Jako parametr funkci pro zbarvení podsítí předán kontext vizualizace a použitý dataset. Nejprve je vytvořena mapa, do které budou následně uloženy páry klíč (`netId`) a příslušná barvy.

Pomocí cyklu `for` je iterováno seznamem uzlů, uzly spadající do skupiny `internet`, `router`, případně uzly, které v datech nemají přiřazenou příslušnost k podsítí jsou nastaveny na jejich základní barvy. Dále jsou získány identifikátory uzlu (`id`) a jeho sítě (`netId`), definována je proměnná pro samotnou barvu a vytvořena je proměna pro náhodné číslo. Pokud mapa neobsahuje identifikátor sítě, je za pomocí náhodného čísla vybrána nová barva ze stanoveného seznamu validních barev, která je dále uložena do mapy, a tedy přiřazena konkrétní podsíti.

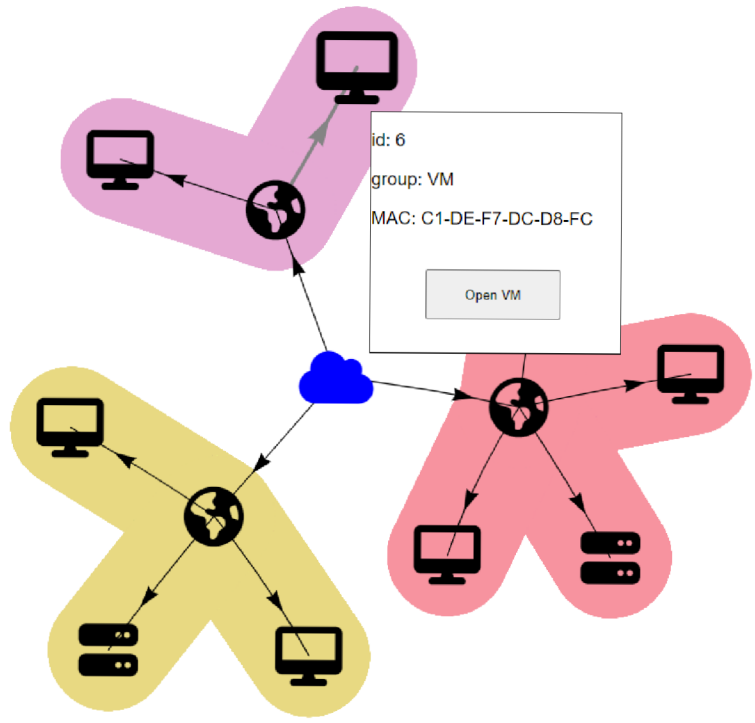
V případě, že mapa již hodnotu barvy pro konkrétní síť obsahuje, je tato barva přiřazena proměnné. Tímto tedy bylo zajištěno, že jednotlivé podsítě budou mít svoji vlastní náhodnou barvu. Nakonec je zvolená barva přiřazena iterovanému uzlu.

V rámci práce bylo navrženo více možných řešení značení podsítí. Prvotní řešení, viz obrázek 4.2, ke značení sítí využívá základních funkcí kreslení na canvas a eventu `beforeDrawing`, které umožnili vytvoření dojmu podkreslení sítí přiřazenou barvou. Tato metoda také využívala náhodné generaci barev, kde je náhodné číslo převedeno do hexadecimálního zápisu barvy. Toto řešení se ale ukázalo jako výpočetně náročné, především při vizualizaci většího množství uzlů a podsítí, a také méně přehledné. Z těchto důvodů bylo zvoleno řešení (viz obrázek 4.3), kde je rozlišení provedeno změnou barvy uzlu v závislosti na příslušnosti ke konkrétní podsítí, a možné barvy jsou předem definovány, které je popsáno výše.

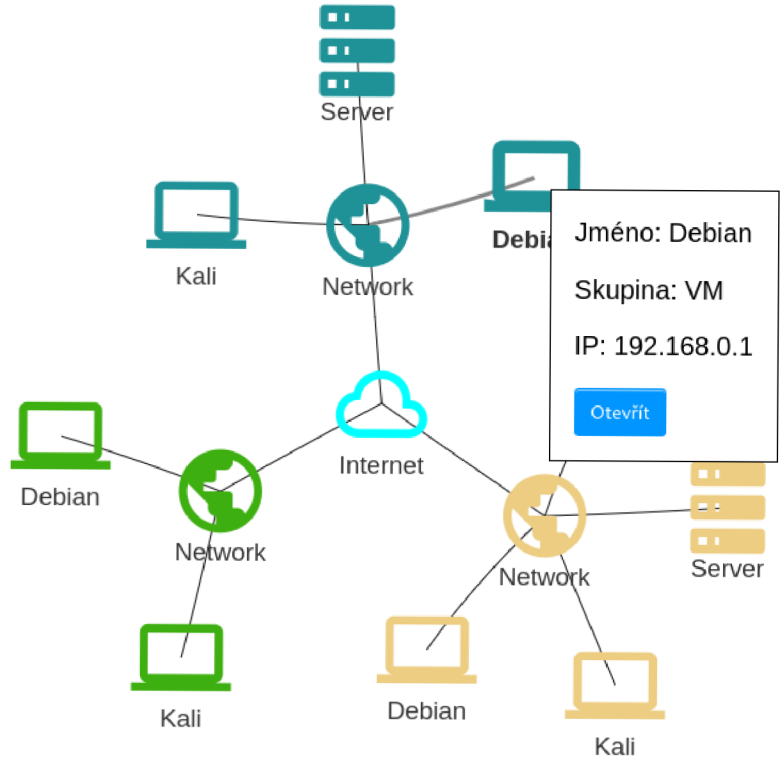
Nakonec byly přidány funkce pro eventy kliknutí na tlačítko otevřít, který uživatele přesměruje na příslušnou stránku, a pro kliknutí myší mimo komponentu vizualizace, která skryje kontextové menu.

Výpis 4.10: Grafické znázornění sítí

```
network.on('afterDrawing', function (ctx) {
  colorNodes(network, jsonData);
  if (drawstate == 0) {
    network.stabilize();
    drawstate++;
  }})
function colorNodes(canvas, dataset) {
  for (let nodes of dataset.nodes) {
    let randomNum;
    let nodeId = nodes.id;
    let chosenColor;
    let netId = nodes.netId;
    let currentNode = canvas.body.nodes[nodeId];
    if (nodes.group == 'Internet') {
      currentNode.setOptions({icon:{color: 'cyan'}})
      continue;
    }
    else if (nodes.group == 'Router') {
      currentNode.setOptions({icon:{color:'black'}})
      continue;
    }
    else if (nodes.netId == undefined) {
      currentNode.setOptions({icon:{color:'black'}})
      continue;
    }
    if (!colorMap.has(netId)) {
      colorMap.set(netId, chosenColor);
      randomNum = Math.floor(Math.random() * colors.length)
      chosenColor = colors[randomNum];
      colorMap.set(netId, chosenColor);
    } else {
      chosenColor = colorMap.get(netId)
    }
    currentNode.setOptions({
      icon: {
        color: chosenColor
      }
    })
  }
}
```



Obr. 4.2: Prvotní návrh vizualizace



Obr. 4.3: Finální verze implementované vizualizace

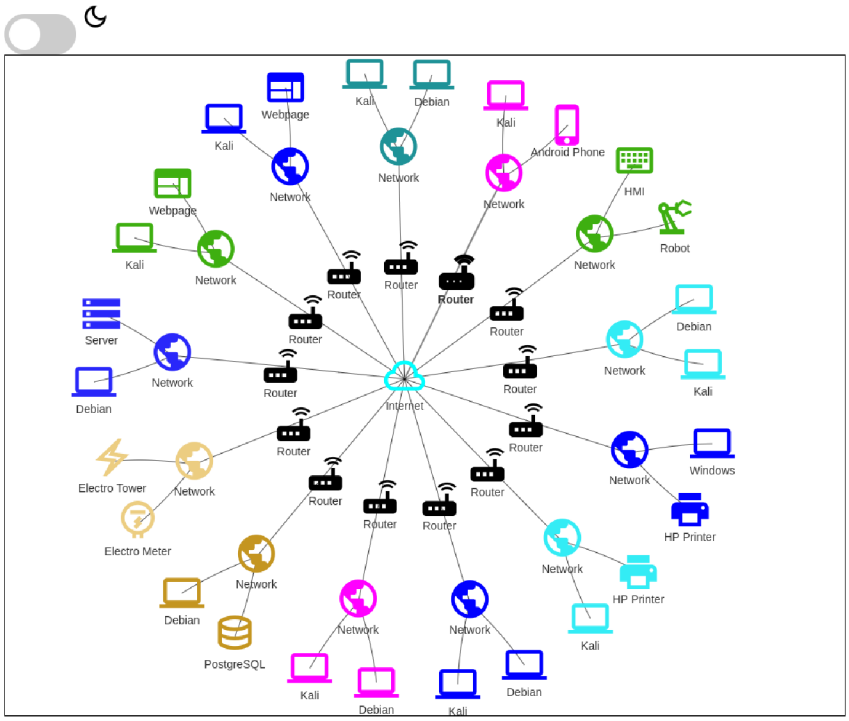
4.4 Finální úpravy

Na základě provedeného testování implementovaného řešení byly provedeny následující finální úpravy:

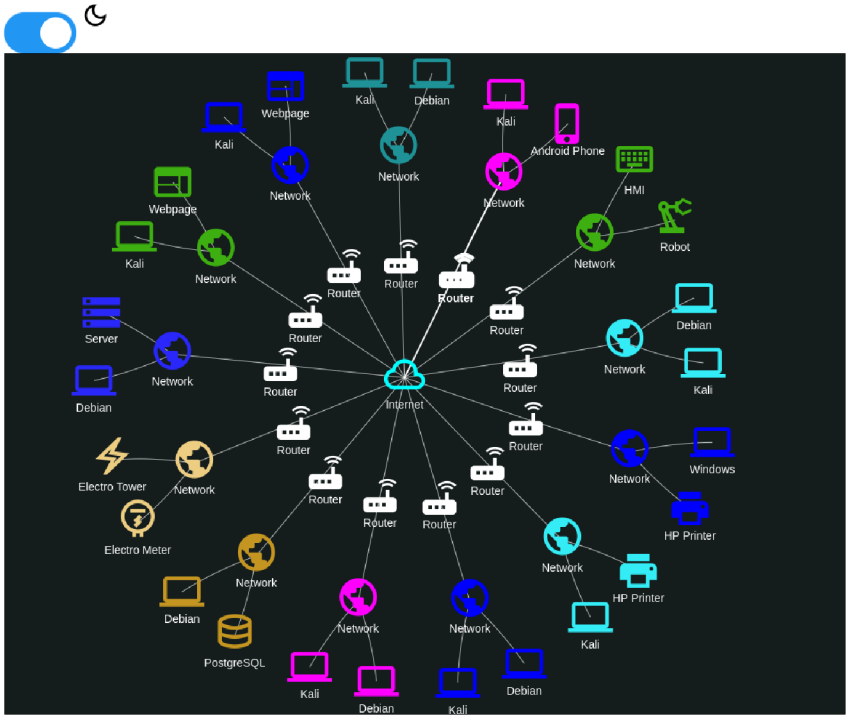
- **Změny vizuálního znázornění jednotlivých sítí** – vlivem větší náročnosti a nižší přehlednosti při vizualizaci většího množství zařízení byl způsob znázornění jednotlivých sítí přepracován na použití menšího množství pevně stanovených barev, které jsou náhodně přiřazeny samotným sítím. Ty jsou dále místo podbarvení znázorněny pomocí zbarvení ikon (viz obrázek 4.4).
- **Přechod na ikony Material Design Icons** – z důvodu nedostatečného množství vhodných ikon dříve používaným FontAwesome jsou nyní používany ikony a symboly knihovny Material Design Icons.
- **Změny prvků kontextového menu** – přidání možnosti zobrazení využitého portu a url, okno samotné se přizpůsobuje velikosti jeho obsahu.
- **Možnost přepínání barevného schéma vizualizace** – cílem je usnadnit uživatelům prohlížení aplikace v různých světelných podmínkách. Pomocí tlačítka nacházející se nad canvasem vizualizace označeného měsíčkem lze přepínat mezi barevným schématem se světlejšími barvami ikon a tmavější barvou pozadí a tmavými barvami ikon a světlou barvou pozadí (viz obrázek 4.5).
- **Možnost vizualizace nových prvků** – v rámci implementace byla přidána možnost vizualizace dalších druhů zařízení, které se mohou ve virtuálních scénářích platformy BUTCA objevit. Seznam všech zařízení zobrazitelných skriptem je do dostupný v tabulce 4.1.
- **Změny provedené pro zvýšení uživatelského zážitku** – stylování elementů, celkové usnadnění vytváření vlastní vizualizace přesunem funkcionalit vizualizace z eventů do funkcí, oprava drobných chyb ve fungování skriptu.

Název	Typ	Ikona
Internet	Centrální bod	
Síť	Centrální bod	
Router	Síťové zařízení	
Desktop	Klientské zařízení	
Tiskárna	Klientské zařízení	
Databáze	Komponenta	
Webová stránka	Komponenta	
Server	Server	
Mobilní telefon	Klientské zařízení	
Ovládací panel	Průmyslové zařízení	
Robotická paže	Průmyslové zařízení	
Rozvodna	Energetické zařízení	
Elektroměr	Energetické zařízení	
Čistírna odpadních vod	Průmyslové zařízení	

Tab. 4.1: Seznam podporovaných zařízení



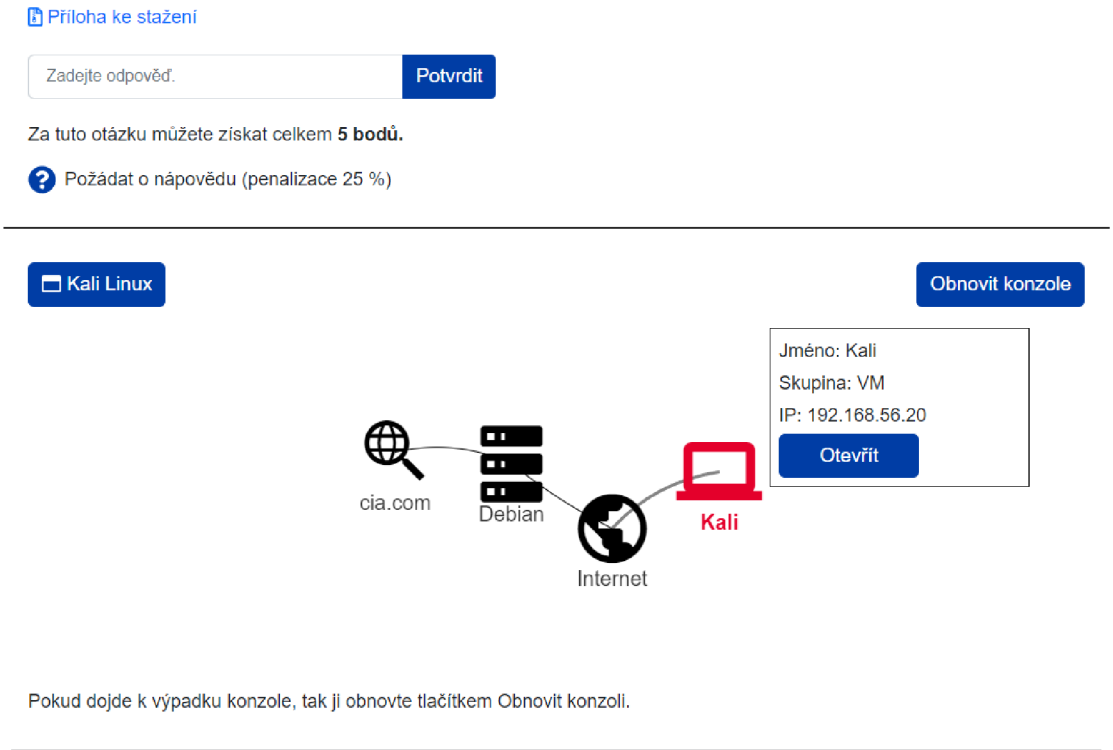
Obr. 4.4: Ukázka vizualizace většího množství zařízení



Obr. 4.5: Ukázka vizualizace většího množství zařízení v režimu „Dark Mode“

4.5 Integrace vlastního řešení do platformy BUTCA

Výstupem bakalářské práce je integrace vytvořeného řešení do platformy BUTCA. Integraci vlastního řešení lze vidět na obrázku 4.6, kde byla integrována do scénáře Dovolená snů. Dosavadním řešení přístupu ke konzoli virtuálního stroje bylo tlačítko `Kali Linux`, které je v rámci bakalářské práce nahrazeno vizualizací topologie daného scénáře, která také umožňuje přístup k jednotlivým uzlům a základní informace o nich. Poskytované řešení je snadno rozšiřitelné, umožňuje vizualizaci uzlů i mimo samotné virtuální prostředí, jako jsou například webové stránky (viz obrázek 4.6).



Obr. 4.6: Integrace vizualizace do platformy BUTCA

Závěr

Na základě stanovených cílů bakalářské práce bylo navrženo interaktivní zobrazení virtualizovaného prostředí, které bylo zasazeno do platformy BUTCA. V teoretické části byly nejprve rozebrány obecně CR platformy, včetně jejich možností využití, přínosů a architektury, kde byly také popsány principy jednotlivých vrstev, definovány základní principy virtualizace a kontejnerizace, a byla popsána také samotná platforma BUTCA. Následně jsou popsány technologie využité pro vytvoření implementace praktické části a na základně stanovených požadavků je provedena analýza JavaScript knihoven určených k vizualizaci dat, v rámci které byla pro implementaci vlastního řešení zvolena knihovna Vis.js.

V praktické části jsou nejprve ustanoveny základní požadavky na implementaci a je vytvořen teoretický grafický návrh, na jehož základě je následně vytvořena samotná implementace v jazyce JavaScript s použitím zvolené vizualizační knihovny. Implementace je nadále rozšířena o funkcionality související s jejím využitím jako prostředek interakce uživatele s platformou BUTCA, jako je zobrazování vlastností uzlů a pokročilé stylování. Důraz je kladen zejména na přenositelnost a přívětivý požitek uživatele. Úspěšnou integrací vlastního řešení do platformy BUTCA bylo dosaženo všech stanovených cílů bakalářské práce.

Literatura

[1] SHAH, S.; MEHTRE, B. M.; CHU, B. T. B.; JONES, M. SHARIF; UDDIN, M. H; Mohammed, M. A. A literature review of financial losses statistics for cyber security and future trend. In *World Journal of Advanced Research and Reviews*, **15**(1), s. 138–156. ISSN 2581-9615.

[2] STODŮLKA, T.; FUJDIAK, R. *Budování Cyber Range platformy s technologií cloud computingu.* Brno: Vysoké učení technické v Brně, 2022, s. 7–25. ISBN 978-80-214-6064-5.

[3] TAYLOR, H. What is a cyber range? *Cyber Security Guide* [online]. [cit. 2022-5-12]. Dostupné z: <`https://cybersecurityguide.org/resources/cyber-ranges/`>

[4] LOSHIN, P. SCADA (supervisory control and data acquisition) *WhatIs.com* [online]. [cit. 2022-11-12]. Dostupné z: <`https://www.techtarget.com/whatis/definition/SCADA-supervisory-control-and-data-acquisition`>

[5] Industrial Control System *TrendMicro* [online]. [cit. 2022-11-12]. Dostupné z: <`https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system`>

[6] What is virtualization?. *Opensource.com* [online]. [cit. 2022-20-11]. Dostupné z: <`https://opensource.com/resources/virtualization`>

[7] What are Linux containers?. *Opensource.com* [online]. [cit. 2022-20-11]. Dostupné z: <`https://opensource.com/resources/what-are-linux-containers`>

[8] HTML: HyperText Markup Language. *Mdn web docs* [online]. [cit. 2022-7-11]. Dostupné z: <`https://developer.mozilla.org/en-US/docs/Web/HTML`>

[9] Document Object Model (DOM). *Mdn web docs* [online]. [cit. 2022-8-11]. Dostupné z: <`https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model`>

[10] HTML Canvas Graphics. *W3 schools* [online]. [cit. 2022-8-11]. Dostupné z: <`https://www.w3schools.com/html/html5_canvas.asp`>

[11] WebGL: 2D and 3D graphics for the web. *Mdn web docs* [online]. [cit. 2022-8-11]. Dostupné z: <`https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API`>

[12] HTML SVG Graphics. *W3 schools* [online]. [cit. 2022-8-11]. Dostupné z: <https://www.w3schools.com/HTML/html5_svg.asp>

[13] CSS: Cascading Style Sheets. *Mdn web docs* [online]. [cit. 2022-8-11]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/CSS>

[14] MAZINANIAN, D. TSANTALIS, N. An empirical study on the use of CSS preprocessors. In *2016 IEEE 23rd international conference on Software Analysis, Evolution, and Reengineering (SANER)* 2016 s. 168–178. ISBN 978-1-5090-1855-0.

[15] JavaScript. *Mdn web docs* [online]. [cit. 2022-6-11]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

[16] ECMA-262. *ecma INTERNATIONAL* [online]. [cit. 2022-6-11]. Dostupné z: <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>

[17] D3.js. *Data-Driven Documents* [online]. [cit. 2022-18-10]. Dostupné z: <https://d3js.org/>

[18] Cytoscape.js. *Cytoscape* [online]. [cit. 2022-18-10]. Dostupné z: <https://cytoscape.org/>

[19] Cytoscape User Manual. *Cytoscascape User Manual* [online]. [cit. 2022-18-10]. Dostupné z: <http://manual.cytoscape.org/en/stable/>

[20] Vis.js. *Vis.js* [online]. [cit. 2022-19-10]. Dostupné z: <https://visjs.org/>

[21] The Keylines Toolkit. *Cambridge Intelligence* [online]. [cit. 2022-19-10]. Dostupné z: <https://cambridge-intelligence.com/keylines/>

[22] Sigma.js. *Sigma.js* [online]. [cit. 2022-19-10]. Dostupné z: <https://www.sigmajs.org/>

[23] Sigma.js. *GitHub* [online]. [cit. 2022-19-10]. Dostupné z: <https://github.com/jacomyal/sigma.js/blob/main/README.md>

[24] VivaGraphJS. *GitHub* [online]. [cit. 2022-19-10]. Dostupné z: <https://github.com/anvaka/VivaGraphJS>

[25] Chart.js. *Chart.js* [online]. [cit. 2022-19-10]. Dostupné z: <https://www.chartjs.org/docs/latest/>

[26] Two.js. *Two.js* [online]. [cit. 2022-19-10]. Dostupné z: <https://two.js.org/>

[27] Go.js. *Go.js* [online]. [cit. 2022-19-10]. Dostupné z: <https://gojs.net/latest/index.html>

[28] ApexCharts.js– Modern and Interactive Open-source Charts. *ApexCharts.js* [online]. [cit. 2022-19-10]. Dostupné z: <https://apexcharts.com/>

[29] Beautiful HTML5 Charts & Graphs. *canvasJS* [online]. [cit. 2022-19-10]. Dostupné z: <https://canvasjs.com/>

[30] Dygraphs. *dygraphs* [online]. [cit. 2022-19-10]. Dostupné z: <https://dygraphs.com/>

[31] Highcharts®JS. *dygraphs* [online]. [cit. 2022-19-10]. Dostupné z: <https://www.highcharts.com/products/highcharts/>

[32] Introducing Ogma, the JavaScript library for large-scale graph visualization and interaction. *LINKURIOUS* [online]. [cit. 2022-19-10]. Dostupné z: <https://linkurious.com/blog/ogma-js-library-large-scale-graph-visualization/>

[33] yFiles. *yWorks* [online]. [cit. 2022-19-10]. Dostupné z: <https://www.yworks.com/products/yfiles>

Seznam symbolů a zkratek

BUTCA Brno University of Technology Cyber Range

CR Cyber range

CSS Cascading Style Sheets

CSV Comma Separated Value

CTF Capture the Flag

DOM Document Object Model

HTML Hypertext Markup Language

JS JavaScript

JSON JavaScript Object Notation

Less Leaner style sheets

SASS Syntactically Awesome Style Sheets

SCADA/ICS Supervisory Control And Data Acquisition/Industrial Control System

SVG Scalable Vector Graphics

TSV Tab-Separated Values

WebGL Web Graphics Library

XML Extensible Markup Language

Seznam příloh

A Příloha A

Výpis A.1: Kompletní nastavení vizualizace

```
const options = {
  autoResize: true, // Automatické přizpůsobení
  velikost vizualizace
  interaction: { // Možnosti interakce
    dragNodes: true, // Uživatel může pohybovat uzly
    dragView: false, // Uživatel může pohybovat oknem
    multiselect: false // Uživatel může v jednu chvíli
    vybrat pouze jeden uzel
  },
  layout: { // Možnosti rozližení
    improvedLayout: true // využití Kamada Kawai algoritmu
    pro prvnotnní vykreslení
  },
  physics: { // Možnosti fyzikálních interakcí
    enabled: true // Použití fyzíkálního modelu
    při pohybu uzlů
  },
  nodes: { // Společné možnosti pro všechny uzly
    shape: 'icon', // Typ ikony - umoží import externích ikon
    icon: {
      size: 80, // Základní velikost
      color: 'black' // základní barva
    },
    borderWidth: 2, // Šířka hrany
    font: {
      size: 20 // Velikost fontu popisku uzlu
    },
  },
  edges: { // Nastavení spojů mezi uzly
    arrows: { // šipky
      to: { // šipka na straně cílového uzlu
        enabled: false
      },
      middle: {  // šipka v prostřed
        enabled: false
      },
      from: { // šipka na straně výchozího uzlu
        enabled: false
      }
    },
```

```
41      endPointOffset: { // Případné odsazení šipek
42        from: 5,
43        to: 5
44      },
45      color: { // Nastavení barev spoje
46        color: 'black', // Základní barva
47        hover: 'black', // Barva při najetí myší
48        opacity: 100 // Průhlednost spoje
49      },
50      dashes: false, // Spoje přerušovanou čarou
51      hidden: false, // Skrytí spojů
52      hoverWidth: 1.5, // Zvýrznení spoje při najetí myší
53      label: undefined, // Popisek spoje
54      length: undefined, // Pevně definovaná délka
55      roztažení spoje
56      při fyzikálích simulacích
57      physics: true, // Fyzikální simulace spojů
58      selectionWidth: 2, // Zvýšení šířky spoje při jeho vybrání
59
60      shadow: { // Možností stínování spojů
61        enabled: false,
62        color: 'black',
63        size: 10,
64        x: 5,
65        y: 5
66      },
67      smooth: { // Pokud je zapnuto, spoje jsou vykreslovány
68      jako dynamické beziérovy křivky
69        enabled: true,
70        type: 'dynamic',
71        roundness: 0.5
72      },
73      width: 1, // šířka spoje
74    },
75    groups: { // Nastavení skupin, na základě kterých jsou uzlům přiřaze
76      Internet: { // Skupina pro "centrální uzel"
77        shape: 'icon', // Nastavení druhu ikony
78        icon: {
79          face: 'Material Design Icons', // Zvolení rodiny
80          ikon, která bude použita
81          code: '\u{F0163}', // Samotný kód ikony
82          color: 'cyan' // Defaultní barva
83        }
```

```
84      },
85      Server: {
86        shape: 'icon',
87        icon: {
88          face: 'Material Design Icons',
89          code: '\u{F048B}'
90        }
91      },
92      VM: {
93        shape: 'icon',
94        icon: {
95          face: 'Material Design Icons',
96          code: '\u{F0322}'
97        }
98      },
99      DB: {
100       shape: 'icon',
101       icon: {
102         face: 'Material Design Icons',
103         code: '\u{F1632}'
104       }
105     },
106     Network: {
107       shape: 'icon',
108       icon: {
109         face: 'Material Design Icons',
110         code: '\u{F01E7}'
111       }
112     },
113     Router: {
114       shape: 'icon',
115       icon: {
116         face: 'Material Design Icons',
117         code: '\u{F11E2}',
118         color: 'black'
119       }
120     },
121     Webpage: {
122       shape: 'icon',
123       icon: {
124         face: 'Material Design Icons',
125         code: '\u{F070F}'
126       }
```

```
127     },
128     Phone: {
129       shape: 'icon',
130       icon: {
131         face: 'Material Design Icons',
132         code: '\u{F011C}'
133       }
134     },
135     Printer: {
136       shape: 'icon',
137       icon: {
138         face: 'Material Design Icons',
139         code: '\u{F042A}'
140       }
141     },
142     HMI: {
143       shape: 'icon',
144       icon: {
145         face: 'Material Design Icons',
146         code: '\u{F15C4}'
147       }
148     },
149     Robot: {
150       shape: 'icon',
151       icon: {
152         face: 'Material Design Icons',
153         code: '\u{F1A1A}'
154       }
155     },
156     ElectroTower: {
157       shape: 'icon',
158       icon: {
159         face: 'Material Design Icons',
160         code: '\u{F0D3E}'
161       }
162     },
163     ElectroMeter: {
164       shape: 'icon',
165       icon: {
166         face: 'Material Design Icons',
167         code: '\u{F1A58}'
168       }
169     },
```

```
170       Water: {
171         shape: 'icon',
172         icon: {
173           face: 'Material Design Icons',
174           code: '\u{F1849}'
175         }
176       }
177     }
178   }
```

B Obsah elektronické přílohy

- `/` ..... Kořenový adresář přiloženého archivu
  - `DataSets` ..... Předpřipravené ukázkové soubory dat
    - `allDevices.json` ..... Vizualizace zobrazuje všechny možné druhy zařízení
    - `fullEnviroment.json` ..... Kompletní topologie scénáře
    - `smallNetwork.json` ..... Zobrazení jednoho prostřední scénáře Dovolená snů
  - `showcase` ..... Složka pro ukázkové soubory
    - `showcaseapp.js` ..... Upravená aplikace pro vizualizaci více souborů na jednu stránku
    - `vishowcase.html` ..... Upravená stránka pro vizualizace více topologií
  - `JavaScript` ..... Logika aplikace
    - `app.js`
  - `Styles` ..... Použité kaskádové styly
    - `style.scss`
    - `style.css` ..... Soubor vygenerovaný ze souboru `style.scss`
    - `style.css.map` .... Vygenerovaný soubor pro propojení `.css` a `.scss` souboru
  - `index.html` ..... Zobrazovaná webová stránka
  - `README.md` ..... Soubor obsahující návod k použití aplikace