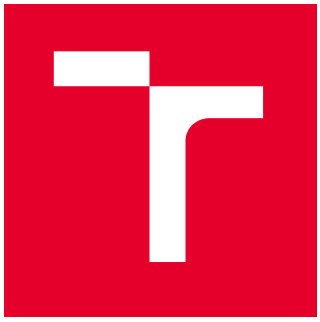
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2022 Petr Kuřina



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

APLIKACE PRO ZNÁZORNĚNÍ STRUKTURY TESTOVANÉHO PROSTŘEDÍ

APPLICATION FOR ILLUSTRATING THE STRUCTURE OF THE TESTED ENVIRONMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Petr Kuřina**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Karel Kuchař**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Petr Kuřina ***ID:*** 211561

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Aplikace pro znázornění struktury testovaného prostředí

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je návrh a implementace nástroje pro grafické znázornění struktury testovaného prostředí v jazyce JavaScript. Data pro zpracování aplikací jsou výsledky provedeného bezpečnostního testování (např. průzkum topologie sítě nástrojem nmap) studentem. Získaná data budou následně zpracována do grafické podoby a prezentována formou dynamické webové stránky s definovaným obsahem. Webová aplikace bude umožňovat export a import získaných informací a podporovat práci s databází. V rámci teoretické části se student zaměří zejména na nastudování problematiky práce s programovacím jazykem Javascript, frameworkem Vue JS a vytvoří grafický návrh vizualizace struktury prostředí. V praktické části se student nejprve zaměří na získávání dostupných informací v rámci bezpečnostního testování a vytvoří experimentální pracoviště pro vývoj aplikace. Následně bude návrh implementován a funkčnost aplikace otestována na experimentálním pracovišti.

DOPORUČENÁ LITERATURA:

[1] HANCHETT, Erik a Benjamin LISTWON. Vue.js in Action. Manning Publications, 2019, 375 s. ISBN 9781617294624.

[2] KIRKBRIDE, Philip. Network Scanning. Basic Linux Terminal Tips and Tricks [online]. Berkeley, CA: Apress, 2020, 2020-08-05, , 119-146. DOI: 10.1007/978-1-4842-6035-7_7. ISBN 978-1-4842-6034-0.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Karel Kuchař

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

ABSTRAKT

Tato bakalářská práce se zabývá vytvořením aplikace pro znázornění struktury testovaného prostředí. V teoretické části jsou popsány nástroje, s kterými se v praktické části pracuje, je to zejména programovací jazyk JavaScript, framework Vue.js a penetrační testování obecně. V praktické části jsou prezentovány výsledky testování topologie sítě, která byla prováděna nástrojem Nmap. Cílem praktické části je vytvořit aplikaci, která bude uživateli srozumitelně demonstrovat výsledky testování.

KLÍČOVÁ SLOVA

Javascript, penetrační testování, virtuální stroj, Vue.js, webová aplikace

ABSTRACT

This bachelor work deals with the creation of an application for the representation of the structures of the tested environment. The theoretical part describes the tools that are processed in the practical part, they are mainly the JavaScript programming language, Vue.js framework and penetration testing in general. The practical part presents the results of network topology testing was performed by Nmap tool. The aim of the practical part is to create an application that will clearly explain the results of testing to the user.

KEYWORDS

Javascript, penetration testing, virtual machine, Vue.js, web application

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Petr Kuřina
VUT ID autora:	211561
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Aplikace pro znázornění struktury testovaného prostředí

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 mé bakalářské práce panu Ing. Karlovi Kuchaři za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Dnešní doba nabízí nejrůznějším firmám a uživatelům možnost otestovat zabezpečení jejich systému. Internet a další technologie s ním spjaté už zcela jistě nejsou tou platformou, kterou kdysi bývali. Jsme svědky mnoha bezpečnostních neúspěchů v oblastech zabezpečování systémů. Nacházíme se v době, kdy internet, jeho technologie, a s nimi spjatá fyzická osoba, se stávají více zranitelnými. Počítačová technika se neustále vyvíjí, a s ní by měl jít ruku v ruce i vývoj zabezpečení systémů. V sázce jsou totiž data, která by mohla vést k poškození klienta. Nynější webové aplikace tedy musí ustát daleko větší tlak a eliminovat tak odtajnění či zneškodnění důležitých dat. Testování zabezpečovacích systémů je vhodné provádět opakovaně, zdokonalovat jejich mechanismy, případně aktivně podnikat kroky vedoucí k jejich nápravě.

Cílem bakalářské práce je návrh a implementace nástroje pro grafické znázornění struktury testovaného prostředí v jazyce JavaScript. Práce se zaměřuje zejména na skenování portů nástrojem Nmap. který pracuje na stroji Windows10. Tímto zařízením je následně testována daná síť. Výsledná data jsou zpracována do grafické podoby a jsou prezentována pomocí webové stránky, která je psaná v programovacím jazyce JavaScript, ve frameworku Vue.js.

V první kapitole je popsán programovací jazyk JavaScript a framework Vue.js a jeho komponenty, např. jádro, Vue-CLI nebo Vuex, které jsou využity k vytvoření webové aplikace, která prezentuje strukturu testovaného prostředí. V ní je možné strukturu upravovat, přidávat nové prvky, např. router nebo switch. Kapitola dále obsahuje návrh experimentálního prostředí a výstupu aplikace.

V druhé kapitole je popisováno penetrační testování, jeho metodiky a dělení testů, postupné fáze testování, typy, nástroje určené pro penetrační testování, strategie a oblasti.

Třetí kapitola se věnuje zejména programovému návrhu aplikace, kde jsou stanoveny požadavky k funkčnosti a jednotlivé komponenty a implementace aplikace.

Ve čtvrté kapitole jsou prezentovány výsledky testování, pro které byl vybrán nástroj Nmap, vhodný zejména ke skenování zařízení v síti, TCP portů a zjištění OS. Tyto výsledky jsou prezentovány na webové stránce, kde si uživatel vybírá ze tří typů skenů a po následném výběru je vizualizována daná topologie.

1 Využití programovací jazyku Javascript

1.1 JavaScript

JavaScript je multiplatformní, objektově orientovaný programovací jazyk, který byl přejat z jazyků C/C++/Java, ale je od nich výrazně odlišen. Patří mezi nejpoužívanější a nejrozšířenější programovací jazyky. Byl vytvořen v roce 1995 a jeho autorem je americký programátor Brendan Eich. Původně měl sloužit jako skriptovací jazyk uvnitř prohlížeče, dále k implementaci skriptů, které měly zlepšovat webové aplikace ze strany klienta. V dnešní době je však nejvíce využíván jako programovací jazyk pro webové stránky nebo pro mobilní, serverové a desktopové aplikace [1]. Ekosystémem JavaScriptu je sbírka softwarových balíčků, knihoven a dalších zdrojů, které usnadňují vývoj při jejich vzájemné integraci. Tyto nástroje jsou vytvořeny různými vývojáři a poskytovateli – například knihovna React je poháněna Facebookem, zatímco Angular framework vytvořil Google a Vue.js byl navržen nezávislým vývojářem. Přidání interaktivního chování na webové stránky JavaScript umožňuje uživatelům interakci s webovými stránkami. Věci, které s JavaScriptem na webové stránce lze dělat je mnoho, neexistují téměř žádná omezení – zde je jen několik příkladů: kliknutím na tlačítko se zobrazí nebo skryjí další informace, změní se barva tlačítka, když na něj uživatel najede myší, přiblížení nebo oddálení obrázku, zobrazení časovače nebo odpočítávání na webu, přehrávání zvuku a videa na webové stránce. 85% webových stránek běží na JavaScriptu, což zahrnuje více než 6 miliard stránek po celém světě. Podporu JavaScriptu má celkem 52 miliard webových stránek. Vzhledem k tomu, že JavaScript dnes běží prakticky na každém druhu výpočetního zařízení, včetně iPhonů, telefonů Android, Apple Mac, Microsoft Windows, Linuxu a chytrých televizí, mohlo by to snadno vysvětlit vysoké celosvětové používání JavaScriptu [2]. Některé JavaScriptové frameworky mění způsob, jakým vývojáři vytvářejí projekty. Patří mezi ně React, Angular, Vue.js. Trio frameworků, které v současnosti patří k nejvíce využívaným v tomto odvětví [3].

Mezi nejvetší výhody patří výrazně menší zátěž serveru, kde webové stránky pracují. Další výhodou je rychlost. JavaScript je na straně klienta velmi rychlý a lze ho spustit ihned v prohlížeči. Uživatel odešle požadavek na server, server zpět zašle HTML a skript, který je následně zpracován prohlížečem. Zde je patrný rozdíl mezi JavaScriptem a další populární technologií určenou pro tvorbu webu PHP, kdy stránka přichází stažená již ze serveru. Díky tomu, že u JavaScriptu server není zbytečně zatěžován, jsou pak weby a aplikace svižnější [4]. K výhodám se řadí taktéž jednoduchost a popularita, jelikož je snadný na implementaci a je používán téměř kdekoliv na webu. K přednostem JavaScriptu patří flexibilita, jelikož umožňuje funkční i objektově orientované programování.

K nevýhodám JavaScriptu se řadí zejména zabezpečení na straně klienta. Kód se spouští v počítači uživatelů a proto může být v ojedinělých případech zneužit ke škodlivým účelům. To vede některé uživatele k zakázání JavaScriptu ve svém počítači. Další nevýhodou je podpora prohlížeče. JavaScript může být někdy odlišně interpretován různými prohlížeči, kvůli tomu je obtížnější psát kód napříč prohlížeči. Škodlivé kódy JavaScript se obtížně odhalují, protože dokážou překódovat škodlivý kód metodami zmatku, jako je logická operace, segmentace řetězců a komprese řetězců. Navíc iterace verze JavaScriptu také ztěžuje detekci [5].

1.2 Framework Vue.js

Jedná se o pokrokový JavaScriptový framework, lze tedy aplikaci rozdělit na části a vyvíjet nezávisle. Tento framework je reprezentován malou knihovnou, která byla vytvořena Evanem You [6]. Využívá se např. k tvorbě SPA (Single Page Application) nebo je použit k vytváření uživatelských rozhraní (UI-User Interface) [7]. Zaručuje možnost automatického testování. Lze tedy vytvářet celou aplikaci v jednom frameworku. Ve Vue je totiž obsažen kompletní ekosystém (jádro, Vuex, Vue-router, Vue CLI a další komponenty). Vue.js se objevilo poprvé s verzí 0.9 s názvem Animatrix, která byla prvním vydáním Vue v únoru 2014. Všechna následující vydání dostaly název Vue. Příklady zahrnují verzi 1.0 (následně Vue 1) s názvem Evangelion, verzi 2.0 (následně Vue 2) s názvem Ghost in the Shell a nejnovější verzi 3.0 (následně Vue 3) s názvem One Piece [8].

Výhodou Vue.js je zejména jednoduchost. Vychází totiž z hlavní myšlenky vývoje, která se snaží o dosáhnutí co nejlepších výsledků s co nejmenším úsilím, díky čemuž by uživatel mohl psát program pomocí několika řádků. Vue.js je velmi vhodný pro práci s komponentami, jelikož komponenty s jedním souborem mohou ukládat všechny kódy, jako je HTML, CSS a JavaScript, do jednoho souboru. Díky jeho výborné flexibilitě, je hojně využíván vývojáři pro jednoduchou dostupnost jeho funkcí. Vue.js využívá virtuální DOM, ten umožňuje, aby byl skutečný DOM reprezentován jako objekty JavaScriptu, se kterými lze inteligentně manipulovat a aktualizovat je, aniž by bylo nutné znovu načítat webovou stránku [8].

Reaktivita

Uvnitř tagu <script> je datová funkce, ze které mohou vývojáři definovat data pro komponentu. Následně se objekt vrácený touto funkcí stává reaktivními zdroji komponenty, tedy znamená to, že kdykoli se změní vlastnost v tomto objektu, komponenta spustí opětovné vykreslení a aktualizaci zobrazení stránky. Každá komponenta má svého watchera k monitorování změny dat pomocí setterů a getterů. Když je detekována změna, watcher spustí opětovné vykreslení, které aktualizuje pohled uživatele nejnovějšími daty. [3].

Jádro

Jádro slouží převážně k dynamické práci s UI. Obsahuje pouze nezbytnou funkcionalitu pro práci s rozhraním, proto je kompaktní, snadno se integruje s jinými technologiemi.

Vuex

Vuex je oficiální knihovna vyvinutá pro Vue.js aplikace a také slouží jako model správy stavu. K managementu všech komponentů aplikace používá centralizované úložiště a zajišťuje, že změny stavu jsou prováděny předvídatelným způsobem s odpovídajícími pravidly [9]. Komplexní velké aplikace využívající Vuex ke správě storu výrazně zlepší stabilitu a škálovatelnost projektu. Vuex propaguje jednosměrný tok dat, který zajišťuje předvídatelné stavové mutace a umožňuje snazší komunikaci mezi komponentami, což má za následek lépe udržovatelný kód ve středně velkých až velkých projektech za cenu více konceptů, které je třeba se naučit [8].

Vue-router

Vue-router se využívá ke směrování klientů. Důležitou funkcí je správné namapování komponentů na trasách, aby je mohl router správně vykreslit. V JavaScriptu zavoláním app.use(router) se získá přístup jako this.$router a také aktuální trasu jako this.$route uvnitř libovolné komponenty. K získání přístupu k routeru nebo trase uvnitř nastavení funkce, se zavolá funkce useRouter nebo useRoute. K vytváření odkazů v HTML se používá vlastní komponenta *router-link*. To umožňuje Vue-routeru změnit adresu URL bez opětovného načítání stránky, zvládnout generování adresy URL a také její kódování. Router-view v HTML zobrazí komponentu, která odpovídá adrese URL [10].

Vue CLI

Jedná se o úplný systém pro rychlý vývoj Vue.js, pomocí kterého můžete spouštět příkazy Vue v terminálu. Je to výkonný systém pro rychlý vývoj Vue JS, můžete vytvořit nový projekt Vue a prototypovat svou aplikaci při sestavování jedním příkazem. CLI je velmi rozsáhlé, dobře se integruje s nástroji Vue build, takže ušetří čas nad konfigurací, a pokud se tak uživatel rozhodne, CLI také umožní zvládnout konfiguraci bez vysunutí. Vue CLI má také volitelné GUI, které lze použít pro vytváření a správu projektů [8].

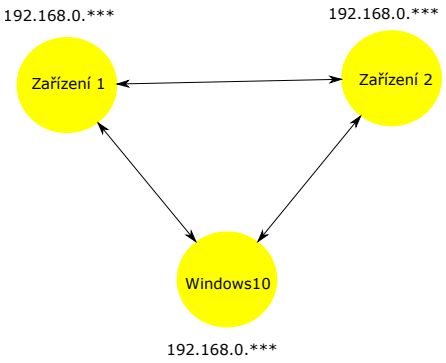
Vuetify

Jedná se o kompletní framework uživatelského rozhraní založený na Vue.js. Poskytuje mnoho nástrojů k vytvoření přibližně takového projektu, jaký si daný vývojář představuje. Framework je přehledný a lze se s ním poměrně rychle naučit pracovat [11]. K jeho výhodám patří velká komunita, díky ní je dobře zdokumentován a existuje mnoho řešení, které jsou dostupné online a mohou sloužit jako inspirace.

1.3 Návrh aplikace

Aplikace bude vytvořena pomocí programovacího jazyka JavaScript. Další důležitou částí bude využití frameworku Vue.js, který dokáže dynamicky měnit vzhled webové stránky pokaždé, co je změněna vlastnost objektu.

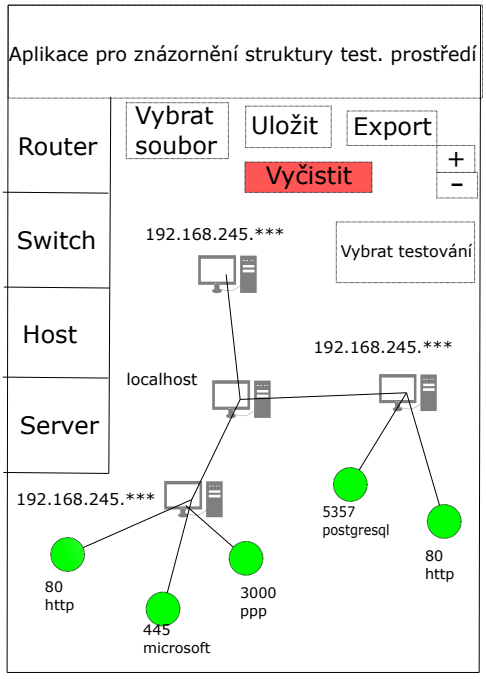
Testovací prostředí se bude odehrávat ve virtualizačním programu VMware Workstation Player 16, který uživateli umožňuje spustit více virtuálních strojů na jednom počítači. Pro účely otestování topologie sítě viz obr. 1.1, budou použity virtuální stroje Kali Linux 2020.3, Windows 10 a Ubuntu 20.04.1. Všechny stroje budou zapojeny do virtualizační sítě VMware Network Adapter VMnet8, která má rozsah IP adresy 192.168.245.0/24. Testování bude realizováno na virtuálním stroji Windows10. K testování bude využita aplikace Nmap. Jedná se o bezpečnostní skener, který dokáže uživateli ukázat, jaké porty jsou na skenovaném stroji otevřené, zavřené nebo filtrované. Nmap byl vybrán k testování z toho důvodu, že je velmi známý a oblíbený u uživatelů, dále je rychlý a jednoduchý na ovládání.



Obr. 1.1: Návrh experimentálního prostředí.

Na obr. 1.1 je zobrazen vytvořený návrh experimentálního prostředí, který obsahuje tři virtuální stroje, jenž budou testovány. Jelikož se jedná pouze o návrh experimentálního prostředí, je poslední trojčíslí IP adresy nahrazeno symbolem ***.

Následně byl vytvořen i návrh výstupu aplikace, viz 1.2, pro lepší představu prezentace výsledků.



Obr. 1.2: Návrh výstupu vytvořené aplikace.

Obr. 1.2 zobrazuje ukázku návrhu výstupu aplikace, kde je skenovaná struktura sítě jprezentována na webové stránce. Otevřené porty budou značeny kolečkem a budou u nich zobrazena čísla daného portu a služba na nich běžící. Pro ukázku byly volné porty a jejich služby náhodně zvoleny. Uživatel bude mít možnost importovat

vlastní výsledky skenování, které budou muset splňovat předem definované požadavky, aby mohly být vizualizovány. K zachování prvků na webové stránce bude sloužit tlačítko *Uložit*, které stránku uloží do databáze Localstorage a při následném obnovení bude rozložení zachováno. Naopak tlačítko *Vyčistit* zajistí vymazání celého obsahu. K přiblížení a oddálení rozložení poslouží tlačítka + a -. Pomocí možnosti *Vybrat testování* dojde k výběru jednoho ze tří testování, následně se zvolí jedna z možností, která zahájí skenování. Po dokončení skenování budou výsledky vizualizovány a uživatel bude mít možnost upravovat jednotlivé prvky. Mezi hlavní úpravy bude patřit smazání prvku, vložení odkazu nebo přejmenování. Následně bude možné napsat poznámku k jednotlivým prvkům a jejich zobrazení bude umožněno po najetí myši na objekt. Strukturu bude možné dále upravit přidáním dalších prvků, jako je router, switch, host a server. Topologie bude zobrazena z pohledu localhosta, ze kterého budou vést následné vazby do dalších zařízení.

2 Penetrační testování

Penetrační testování je uskutečňováno za účelem posouzení bezpečnosti daného systému. Jedná se o oprávněný pokus o proniknutí do dané IT infrastruktury. Testování probíhá bez jakékoliv spolupráce s objednavatelem, je vykonáváno buď samostatným testerem nebo bezpečnostním týmem, díky němuž se dosáhne přibližnému útoku, který by byl proveden hackerem. Vyžaduje povolení od vlastníka systému. Penetrační testování by se mělo provádět pravidelně, s cílem včas zareagovat na případnou hrozbu a zvýšit tak bezpečnostní standard. Hledají se slabá místa, která jsou zranitelná. Pokud je zranitelnost nalezena, je sjednáno její napravení [12].

2.1 Metodiky

Není žádný přesný návod na to, jak správně testovat a postupovat při penetračním testování. Každý tester má jiné zkušenosti, myšlení atd., proto se začaly vytvářet metodiky, dle kterých lze postupovat při testování. Mezi nejznámější patří OSSTMM nebo OWASP.

Open Source Security Testing Methodology Manual

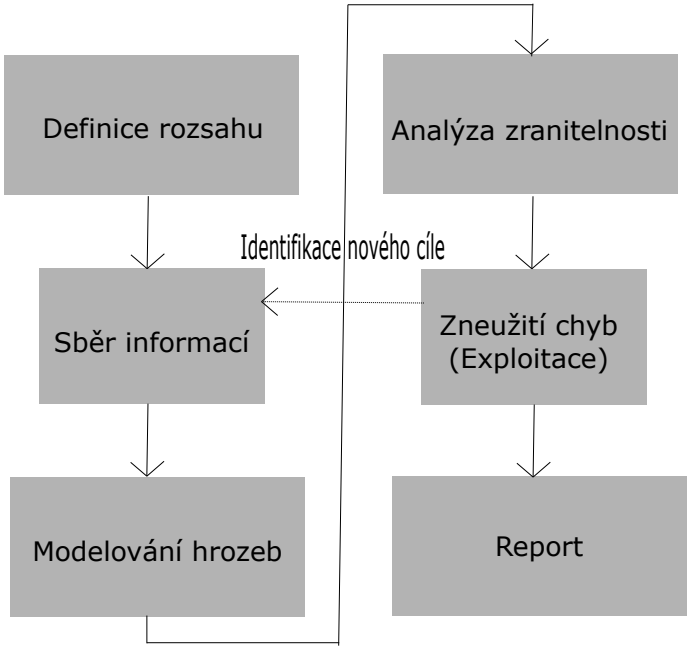
OSSTMM je metodika, vedená institutem ISECOM, která má více než 200 stran a je volně dostupná pro uživatele. Obsahuje konkrétní popis kroků při penetračním testování a jejich cílů. Příručka je aktualizována každých šest měsíců, aby zůstala relevantní pro aktuální stav testování zabezpečení. Primárním účelem manuálu je poskytnout vědeckou metodologii pro přesnou charakterizaci provozní bezpečnosti prostřednictvím zkoumání a korelace výsledků testů konzistentním a spolehlivým způsobem. Jako open-source projekt umožňuje každému testerovi přispívat nápady na provádění přesnějších, použitelnějších a účinnějších bezpečnostních testů. Dále umožňuje volné šíření informací a duševního vlastnictví [13].

Open Web Application Security Project

OWASP je nezisková organizace, která se zaměřuje na zlepšení zabezpečení softwaru, podporuje organizace ve vývoji, nákupu a údržbě důvěryhodných aplikací. OWASP nabízí mnoho nástrojů, návodů a testovacích metodik pro kybernetickou bezpečnost pod licencí open source, konkrétně OWASP Testing Guide [14]. Tato organizace vytvořila bezplatný standard OWASP Top 10, který vzdělává vývojáře, organizace atd. o důsledcích nejvýznamnějších slabin v zabezpečení webových aplikací a je opakovaně vydáván po třech letech. Poskytuje základní techniky pro ochranu proti těmto zranitelnostem, které patří mezi největší hrozby.

2.2 Postupné testovací fáze

Obecné metodiky nejsou zcela jednotné, vždy ale vychází ze základní struktury, která je stejná. Počet těchto fází je v rozsahu od čtyř do sedmi 2.1. Záleží na požadavcích, zkušenostech i představách objednavatele a testera.



Obr. 2.1: Postupné fáze penetračního testování.

Definice rozsahu

Základ je vymezit prioritní cíle útoku, které jsou závislé na požadavcích druhé strany a odborném odhadu. Je zde kladen důraz na přesnost, aby penetrační tester věděl, které systémy, technologie nebo IP adresy jsou obsahem testování. Dále je třeba se vyjádřit k tomu, co testování nebude obsahovat, aby nedošlo k narušení provozu aplikace. Mezi obsah této fáze se řadí i identifikace významných aktiv společnosti, jako je např. databáze klientských údajů, která patří do testovaného systému [15].

Sběr informací

Cílem této fáze je zjistit co nejvíce důležitých informací o daném systému a jeho prostředí, ty jsou poté využity pro další fáze testování. Pro získání těchto informací se využívají veřejně dostupné zdroje (např. Google, Whois) nebo se používají specializované nástroje, jenž skenují porty (např. Nmap, Zenmap). Více se využívá přímá interakce s testovaným systémem, protože veřejné zdroje jsou ve většině případů méně přesné [15].

Modelování hrozeb

Po získání informací o systému začíná expertní tým sestavovat vzor útoku podle definovaných kritérií. Pracuje se s databázemi zranitelností a specializovanými nástroji, jako je např. Nessus (skenner zranitelnosti) nebo SecurityFocus (databáze zranitelností) [15].

Analýza zranitelnosti

Analýza zranitelnosti slouží ke kompletnímu zhodnocení bezpečnosti aplikace. Pomocí manuálních a automatizovaných nástrojů jsou odhalovány chyby nebo slabá místa v systému, což může představovat hrozbu v tom, že bude útočník schopen vniknout do systému. Velkou roli v detekci zranitelností zde hraje znalost a zkušenost testera, která je pro celé testování klíčová [15].

Zneužití zranitelnosti

Zneužití zranitelnosti se označuje také jako exploitace. Jedná se o využití dané zranitelnosti jako takové, která může vést k narušení bezpečnosti celého systému. Zneužití zranitelnosti se provádí na slabých místech, které byly zjištěny v předchozí fázi. Při odhalení nového cíle, je potřeba se vrátit ke sběru informací a opakovat exploitaci. Pokud je exploitace dané služby úspěšná, může odhalit cestu k další službě, o které se předtím nevědělo. Tester se snaží zejména o získání přístupu do neautorizovaných částí systému a citlivých dat z databází. Nejpoužívanější nástroj je Metasploit Framework, ve kterém je obsaženo přes 1200 exploitů pro nejznámější operační systémy, všechny exploity jsou navíc ověřeny [15].

Report

Report je velmi významná fáze testování, dochází zde k rekapitulaci a odevzdání výsledků penetračních testů klientovi. Report bývá většinou rozdělen na dvě části. V první části jsou shrnuta rizika systému a s nimi spojená možná ohrožení aktiva

společnosti. Druhá část je zaměřena více na technickou stránku a jsou v ní detailně popsány jednotlivé zranitelnosti, úspešné útoky nebo průběh testů. Cílem reportu je druhé straně prezentovat závěrečné výsledky tak, aby došlo ke zlepšení zabezpečenosti systému. Zpráva by se měla odevzdat co nejdříve, čas je důležitým faktorem, výsledky totiž ztrácí na hodnotě, pokud jsou zpracovány po více než půl roce. Veškerá komunikace obou stran by měla být šifrována RSA klíči, aby nebyla zneužita případným útočníkem [15].

2.3 Typy testování

Otestovat software lze třemi způsoby, manuálně, automatizovaně nebo semi-automatizovaně. Je na klientovi, který typ si zvolí. Všechny způsoby mají své klady a zápory, takže je důležité zvolit ten správný typ, aby nedošlo k neúplné kontrole.

Automatizované testy

Jsou prováděny testerem, kterému stačí naučit se s testovacími nástroji pracovat, následně software provede test a poté tester musí umět správně vyčíst výsledky daných testů. Je zapotřebí znát vstupy a výstupy. Využívají se především při opakovaném testování, kdy je potřeba zkontrolovat velké množství generovaných dat a spustit větší množství testů. K výhodám patří menší časová náročnost provedení testu, dále pak větší počet opakování testů nebo otestovaní výrazně vyššího množství vstupů a výstupů. Nevýhody automatizovaného testování jsou především v kvalitě testu, protože nemusí vždy otestovat všechna zranitelná místa.

Manuální testy

Manuální testy jsou prováděny testerem, který provádí vše ručně a díky tomu dokáže vytvořit testy na míru podle určitých podmínek. Jsou obtížněji detekovatelné než automatizované testy. Tester nemusí využívat žádný ze specializovaných nástrojů, které jsou potřeba zejména při automatizovaném testování. Manuální testy jsou využívány zejména k detekci logických chyb v systému nebo aplikaci, dále v případech, kdy je potřeba lidské úvahy a ohodnocení. Dále jsou používany při vývoji nových aplikací. Mezi výhody manuálního testování patří důkladnější testování, jsou tedy schopné odhalit větší škálu případných slabých míst systému. K nevýhodám se řadí delší doba provedení testu, podrobnější znalosti problematiky a využití specializovaných nástrojů.

Semi-automatizované testy

Jedná se o kombinaci automatizovaných a manuálních testů, kdy se tester snaží využít výhod již dvou zmíněných testů. Testování je sice automatizované, ale je současně potřeba i tester, který obstarává manuální práci. Tester spouští a spravuje automatizované testování a eventuálně vyhodnocuje jeho výstupy.

2.4 Nástroje pro penetrační testování

Testeři využívají specializované nástroje, které jsou vytvořeny buď na míru nebo pro obecnější testy. Je mnoho druhů těchto nástrojů, mezi nejpoužívanější patří skenery.

Metasploit Framework

Jedná se o open-source nástroj, který vyvíjí a používá exploity. Testuje slabiny v operačních systémém a aplikacích. Tento penetrační testovací nástroj je založen na konceptu „využívaní“. Spustí sadu kódů na testovacím cíli vytvořením rámce pro penetrační testování. Je k využití na systémech Linux, Apple Mac OS X a Microsoft Windows.

Nessus Vulnerability Scanner

Nessus je nástroj pro penetrační testování a vzdálený bezpečnostní skener, obvykle běží na jednom počítači, kde skenuje služby nabízené vzdáleným strojem. Tento nástroj umožňuje uživateli skriptovat a spustit konkrétní kontroly zranitelnosti. Nessus je určen zejména k vyhledávání zranitelností, jako je například nesprávná konfigurace systému nebo účty s chybějícími hesly. Jedná se o nejpoužívanější skener zranitelnosti na světě, který se používá ve více než 75 000 organizacích celosvětově [16]. Je využíván na různých platformách a OS. K jeho výhodám se řadí velká přehlednost a možnost uložit dané testování ve více možných formátech, jako je LaTeX nebo PDF.

Network Mapper

Network Mapper zkráceně Nmap, jedná se o bezpečnostní skener portů. Za účelem rozvíjet síťové služby a mapy sítě, jsou Nmapem odesílány speciálně vytvořené pakety do definovaného cíle a poté se analyzují odpovědi. Tento nástroj podporuje skenování různých typů protokolů a většiny stávajících systémů. Zjišťuje typ operačního systému na skenovaném zařízení (OS fingerprinting). Pomocí bezpečnostního skeneru lze odhalit, na kterém portu aplikace naslouchá. Nmap se využívá zejména k hledání hostitelských počítačů a služeb na počítačové síti. Je hojně rozšířený u uživatelů s OS Linux [17].

Zenmap

Vychází z Nmapu, jedná se totiž o jeho GUI (grafické uživatelské rozhraní). Zenmap je multiplatformní aplikace, která je open-source a zdarma. Tato aplikace se snaží usnadnit začátečníkům používání Nmapu a zároveň poskytnout zkušeným uživatelům pokročilejší funkce. Při častém skenování je lze ukládat jako profily, aby jej mohl uživatel opakovaně snadno spouštět. Výsledky skenování lze uložit a zanalyzovat později. Ty se navzájem porovnají s předchozími a zjišťuje se, zda se liší. Poslední výsledky ze skenovaní se ukládají do databáze, kterou je možno prohledat.

Burp Suite

Burp Suite je integrovaná platforma provádějící testovací zabezpečení webových aplikací. Patří do skupiny bezpečnostních skenerů. Umožňuje jak průzkum, tak i změnu komunikace mezi serverem a webovým prohlížečem. Jeho různé nástroje hladce spolupracují na podpoře celého procesu testování, od počátečního mapování a analýzy povrchu útoku aplikace, až po hledání a zneužití slabých míst zabezpečení.

Wireshark

Wireshark [18] je přední, široce používaný analyzátor síťových protokolů. Demonstruje uživateli komunikaci v jeho síti. Díky tomu je standardem v mnoha komerčních, ale i neziskových firmách, dále pak ve vládních agenturách a vzdělávacích institucích [19]. Lze jej využívat multiplatformně. Je neustále vylepšován, díky čemuž může být provedena hluboká inspekce stovek protokolů. Může se využít offline analýza nebo živý záznam. Zachycená síťová data je možné procházet skrze GUI nebo pomocí obslužného programu TShark v režimu TTY. Podporuje dešifrování mnoha protokolů, včetně IPsec, ISAKMP, Kerberos, SNMPv3, SSL / TLS, WEP a WPA / WPA2.

2.5 Obecné dělení testů

Externí testování

Útok se provádí mimo organizaci, tedy z vnější strany testované sítě. To pomáhá určit, jak se dostane vnější útočník do sítě a jak až daleko může získat přístup při jednom převzetí kontroly.

Interní testování

Testovací útok běží ve vnitřní straně sítě jako interní uživatel, který má běžné přístupové oprávnění. Má napodobit potencionálního útočníka, který nějakým způsobem získal přístup do vnitřní sítě.

2.6 Strategie testování

Je důležité určit správný způsob testování systému, ten se odvíjí od stanovení dané strategie dodavatelem. Testeři mají následně za úkol vybrat efektivní typy testů a nástrojů, které budou využívat. Zvolení vhodného testu je naprosto klíčové, aby se dosáhlo největší účinnosti otestování. Nesprávné zvolení strategie může znamenat, že dodání výsledků testů se znatelně opozdí nebo výrazně zvýší cenu za testování systému. Jsou známy tři úrovně znalostí o testovaném systému:

Black-box testy

Na testovaný systém se nahlíží jako na tzv. černou skříňku, kde není známa vnitřní infrastruktura. Do této metody spadají útočníci, kteří ví o systému jen veřejnou informaci, jakou je třeba doménové jméno serveru, tu následně dále prozkoumávají a testují. Tester by tedy neměl mít žádný přístup ke zdrojovým ani binárním kódům, jsou zde známy vstupy a potenciální výstupy. Díky vstupním datům může tester očekávat, jak se systém bude chovat a jaké bude mít výstupy. Umožňují získat informaci o tom, zda byl test úspěšný. Tento typ testů závisí na testovacích scénářích, které si může tester vytvořit nebo mu je může někdo poskytnout. K testování jsou využity automatizované nástroje nebo je prováděno manuálně. Hlavní výhodou je efektivita při použití na velkých systémech. K dalším výhodám patří pomoc odhalit jakékoliv nejasnosti nebo nesrovnalosti ve specifikovaných požadavcích. Mezi nevýhody patří nežádoucí chování aplikace, protože systém může provádět akce, o kterých testeři neví, jelikož se neobjeví na výstupu. Kvůli neznalosti se kód netestuje, neví se tedy, zda je aplikace správně napsána [20, 21].

White-box testy

U tohoto typu testování jsou testerovi k dispozici všechny potřebné informace o systému, tedy jeho vnitří infrastruktura. Podrobné informace mohou odhalit případné zranitelnosti dříve a lépe zanalyzovat daný systém. Testeři tak znají topologii sítě i zdrojové kódy aplikací, navíc je nezbytné vědět o implementaci aplikace. Po porozumění dostupného kódu udělá tester analýzu, která se provádí buď manuálně nebo automatizovaně. Využívá se pro ladění kódu, případně nežádoucího kódu nebo k zjištění, jak je systém zabezpečený před neautorizovaným přístupem. White-box testování je využíváno zejména webovými službami, které jsou ve start-up fázi. K hlavním výhodám se řadí detekce nežádoucího kódu aplikace a rychlost odhalení chyb, díky analýze zdrojového kódu. Další výhodou je znalost vnitřních částí testovaného softwaru. Nevýhody White-box testování jsou dány velkou znalostní náročností programovacích jazyků, testovacích nástrojů a systému. Kvůli zkušenějším testerům a specializovaným nástrojům, jako je třeba debugger, jsou i větší náklady [20, 22].

Grey-box testy

Jedná se o kombinaci dvou předešlých typů. Jsou známy pouze základní vnitřní data o systému, díky čemuž se tester snaží maximálně využít a získat co nejvíce informací o systému a algoritmus, podle něhož může být navrhnut ideální testovací scénář. Test probíhá spíše z hlediska útočníka, jako je tomu u Black-box testování, ale v tomto případě má tester znalosti o implementaci aplikace. Případně se test aplikace provádí z pohledu uživatele. Grey-box testování využívá White-box k detekci zranitelností a Black-box k ověření, zda je zranitelnost využitelná při útoku. Mezi hlavní výhody patří kombinace výhod White-box a Black-box technik a znalost algoritmu, která vede k sestavení správného testovacího scénáře. Nevýhodami jsou nedostupnost testera ke zdrojovému kódu aplikace, nelze tedy otestovat všechna potřebná místa, následně nižší kvalita kódu, jako u Black-box testů, není tedy jasné, zda je aplikace správně napsána.

2.7 Oblasti testování

Social engineering

Jde o popis útoku, který zcela závisí na lidské chybě. Shromažďuje cenné a citlivé informace pomocí psychologické manipulace k oklamání legitimních uživatelů. Tento druh útoku je velmi nebezpečný, protože chyby uživatelů jsou méně předvídatelné. Toto testování pomáhá organizaci vyhodnotit jejich dodržování organizačních zásad a zaměstnaneckých postupů. Pomáhá také při zlepšování bezpečnostních školení, která jsou poskytované zaměstnancům. Nejvyužívanějšími technikami sociálního inženýrství jsou Phising a Pretexting, které je mnohdy složité odhalit pro běžné uživatele.

Testování webových aplikací

Testování zabezpečení webu je testování obchodní logiky, ověřování vstupů, výstupního kódování, autentizace a autorizačních problémů k vyhnutí se běžným zranitelnostem, jako jsou SQL Injections, Cross-site Scripting (XSS) a další. Testování zabezpečení lze provádět pomocí kombinací manuálního testování, prostřednictvím bezpečnostích testerů a automatických nástrojů určených k testování zabezpečení webových aplikací, které se spolu vzájemně doplňují. Aby bylo možné provádět testování bezpečnosti webových aplikací manuálně, testeři by měli mít přehled jak v HTTP protokolu a databázi OWASP Top 10 zranitelností, tak i v dalších známých zranitelnostech [23].

3 Programový návrh aplikace

Aplikace bude vytvořena frameworkem Vue.js. Framework je využíván např. k tvorbě SPA (Single Page Application) nebo k vytváření uživatelských rozhraní (UI - User Interface) [7]. Vue.js je vhodný pro práci s komponentami, jelikož komponenty s jedním souborem mohou ukládat všechny typy kódů, jako je HTML, CSS a JavaScript, do jednoho souboru s příponou vue.

3.1 Stanovení požadavků

K hlavním požadavkům funkčnosti aplikace patří grafické znázornění struktury testovaného prostředí pomocí JavaScriptu, tedy vizualizování topologie sítě a zobrazení jednotlivých prvků s odpovídajícími informacemi. Prostředí je možné přibližovat a oddalovat, jednotlivé prvky dále i přesouvat. Prvky je možné upravovat, po kliknutí pravým tlačítkem lze prvek smazat, přejmenovat, vložit odkaz, přidat popisek nebo přidat vazbu mezi prvky. Najetím myši na prvek jsou zobrazeny poznámky o aktuálním prvku. Tuto strukturu je možné importovat i exportovat v souborech s příponou .json. Celé prostředí je dále možné smazat kliknutím na tlačítko *Vyčistit*. Testování je prováděno na základě aplikace Nmap, kterou musí mít uživatel nainstalovanou na svém zařízení, aby mohlo být umožněno testování daného prostředí. Uživatel si zvolí ze tří předem definovaných možností testování právě jednu možnost. Ta je následně vizualizována na webové stránce.

3.2 Komponenty

Komponenty se řadí mezi nejdůležitejší prvky frameworku Vue.js. Aplikace se skládá ze dvou komponentů, které si předávají data a komunikují spolu. Patří sem komponenty *Topo.vue* a *ContextMenu.vue*. Hlavním komponentem je zde *Topo.vue*, který zajišťuje rozložení stránky, metody volání pravého menu a tlačítek. Tato komponenta má v sobě naimportovánu komponentu *ContextMenu.vue*, která obsahuje menu pravého tlačítka objektu, jako je vazba, popisek nebo odkaz. Jedná se o jednodušší komponentu, která pouze vypomáhá hlavní komponentě *Topo.vue*. Jak již bylo řečeno, hlavní komponentou je zde *Topo.vue*, jelikož obstarává co nejvíce práce. Je zde vytvořeno menu objektů, poté pomocí skriptu Context-Menu z komponenty *ContextMenu.vue* je vytvořena mapa topologie a rozmístěna tlačítka mapy. Následně jsou naimportována tzv. MessageBox, sada modálních boxů, která informuje o dotazech uživatele. Pokud uživatel chce například prvek odstranit a klikne na tlačítko Odstranit, MessageBox se ho poté dotáže, zda chce opravdu daný prvek odstranit. Následně je zde naimportována knihovna prvků nodeData.js, v níž jsou obsaženy

všechny prvky, které lze manuálně přidat do prostředí. Jsou zde vytvořeny metody reakcí na akce myši, např. reakce na posouvání objektů, na vytvoření vazeb nebo otevírání url odkazů. Dalšími metodami jsou již zmíněné MessageBox a k nim příslušné akce, jako je např. export nebo smazání topologie. Poté je využit nástroj computed, který vrací pozice uzlů. V konečné fazí komponenty *Topo.vue* je metoda mounted(), která přistupuje k reaktivním prvkům.

3.3 Složky

Mezi hlavní složky projektu patří src, public a node_modules. K nim byla vytvořena složka *server*, ve které je obsažen balíček Node-Nmap. Tento balíček umožňuje NodeJs aplikaci propojit se s funkcemi Nmapu, dále vyžaduje mít nainstalován Nmap na daném zařízení a být dostupný pro běžící Node aplikaci. Součástí složky src je složka *code*, která obsahuje backend dotazy. Tyto dotazy se týkají všech tří typů skenů. Backend naslouchá na adrese localhost:3000. Podle typu skenů je rozdělen na it localhost:3000/quick-scan, it localhost:3000/scan-ip-ports, it localhost:3000/scan-op-ports. Obsahuje knihovnu Axios, která je založená na XMLHttpRequestech, jenž je využívána k volání AJAXu a která usnadňuje získávání dat z externích zdrojů (například API) [24].

3.4 Implementace programu

Program se skládá ze dvou částí. Tou první je frontendová aplikace, která je tvořená v JavaScriptu pomocí frameworku Vue.js, běžící na adrese localhost:8080. Druhou částí je backend, který je tvořený Node.js serverem, běžícím na adrese localhost:3000. Node.js server vystavuje endpointy připravené pro typy skenu. Využitá knihovna pro komunikaci s Nmapem vrací po skenu dotazovaný host a jeho síťové prvky. V takové struktuře backend vrátí data na frontend, kde se vizuálně vygeneruje jejich struktura.

Frontend

Nejprve je potřeba mít nainstalované prostředí Node.js, které umožňuje spouštět JavaScript mimo webový prohlížeč. V tomto případě byla využita verze 16.14.0 viz. [27]. Následně byl spuštěn Příkazový řádek (cmd), pomocí něj byl zadán příkaz k nainstalování balíčku *@vue/cli*, který je schopen rychle vytvořit nový projekt pomocí příkazu *vue create*. Poté byl projekt vytvořen pomocí příkazu *vue create app_topo*. Další krokem bylo zvolení potřebných funkcí projektu. Z defaultního nastavení byl odstraněn *Linter/ Formatter*. Transpiler Babel je využit jako překladač

zdrojových kódů mezi novými a starými standardy. Zajišťuje tedy funkcionalitu i na starších zařízeních. Verzí tohoto Vue.js projektu se stala verze 2.x, jelikož verze 3.x ještě zcela není rozšířená a nepodporuje zatím určité funkce, např. Vuetify. V konečné fázi vybírání se zvolily možnosti *In dedicated config files* a *No*. Následně byl projekt úspěšně vytvořen s veškerým potřebným nastavením viz obr. 3.1.

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel
? Choose a version of Vue.js that you want to start the project with 2.x
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? No
```

Obr. 3.1: Ukázka přednastavení projektu v příkazovém řádku.

Následně se v Terminále změnil adresář příkazem *cd app_topo*, finálním příkazem *npm run serve* se projekt spustí a je k dispozici buď lokálně nebo na síti viz obr. 3.2.

```
App running at:
- Local:   http://localhost:8080/
- Network: http://192.168.245.128:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Obr. 3.2: Zobrazení příslušných odkazů kde lze aplikaci spustit

V programu je využita knihovna Element-UI, která zajišťuje interaktivitu UI. Patří sem např. dialogová okna, tlačítka, záhlaví nebo kontejnery.
V prvním kroku je třeba vytvořit komponentu Topo.vue, kde bude řešená vizuální stránka aplikace a další součásti potřebné k vytvoření aplikace (tlačítka, záhlaví s názvem aplikace nebo kontextové menu). Především je důležitá tvorba plochy, kde bude znázorněna výsledná topologie skenování. K tvorbě plochy je využit nástroj SVG. Tento nástroj je frontedovými vývojáři využívám k tvorbě dynamických prvků, které mohou disponovat libovolnou velikostí, v závislosti na volbě uživatele. Je vytvořen <svg> kontejner, který řeší vizuální interakci výsledků skenování.

Dále jsou vytvořena tlačítka k exportu a importu scanu, k uložení, vyčistění plochy, pro přiblížení a oddálení a pro výběr testu. Po kliknutí na tlačítko *Výber testu* se zobrazí dialogové okno s výběrou nabídkou. Tato nabídka obsahuje tři možnosti skenování. U každého skenu je předem určená IP adresa, kterou si může uživatel upravit a následně spustit sken. Pokud uživatelem zvolená IP adresa není v síti, je vizualizován pouze localhost, který bude přítomen v každém výsledku skenování. V opačném

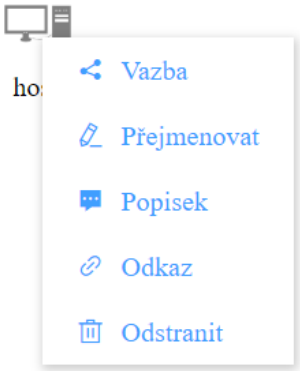
případě je zobrazena spolu s localhostem a případnými dalšími adresami, v závislosti na rozsahu. Podle zvoleného typu skenování jsou graficky znázorněny různé struktury, např. topologie se zařízeními s otevřenými porty, jenž jsou zobrazeny barevným kolečkem. Barva je přiřazena náhodnou funkcí. Modrá barva reprezentuje vazbu mezi skenovanými hosty a červená naopak značí vazbu mezi hostem a portem. Pod kolečkem je zobrazeno příslušné číslo odpovídajícího portu a služba na něm běžící. Dále je možné zobrazit i OS zařízení, pokud jej skener odhalí. Toto označení je napsáno pod ikonou hosta. S prvky lze manipulovat a přesouvat je na libovolnou pozici. O to se stará metoda *moveAndLink*, jenž řeší reakci myši na pohyby prvků. Uložení struktury do databáze zajišťuje metoda *saveTopo()*, která ukládá data do řetězce json. Následně je zobrazeno pop-up okno, které oznamuje o uložení do databáze LocalStorage viz 3.1.

Výpis 3.1: Výpis kódu metody saveTopo()

```
saveTopo() {
      localStorage.topoNodes = JSON.stringify(this.topoNodes);
      localStorage.topoLinks = JSON.stringify(this.topoLinks);
      MessageBox.alert("Uloženo do localStorage", "Uložení Topologie", {
        type: "info",
        confirmButtonText: "Ok",
      });
```

Ke smazání celé topologie slouží tlačítko *Vyčistit*, které je definováno metodou clearTopo(). Po kliknutí se objeví pop-up okno, které se uživatele dotazuje, zda chce opravdu smazat celou topologii. Pokud ano, topologie je odstraněna z databáze LocalStorage. Dalším tlačítkem je Import, který převádí vybraný soubor s příponou .json na objekt JavaScriptu, aby mohl být vizualizovaný. Naopak tlačítko Export, převádí JavaScriptový objekt na datový řetězec v raw struktuře. V levém bočním menu se nachází výběr čtyř prvků, které lze přidat do struktury. Jednotlivé vložené prvky mohou mezi sebou tvořit vazby. Tyto prvky simulují určitá síťová zařízení.

Druhá komponenta ContextMenu, která je importována do Topo.vue, obsahuje kontejner, který řeší správu kontextového menu prvků. Toto menu je vyvoláno případným kliknutím pravého tlačítka myši viz 3.3.



Obr. 3.3: Zobrazení kontextového menu

Kořenovým adresářem celé aplikace je App.vue, která má v sobě naimportováno komponentu Topo.vue. Obsahuje pouze tuto komponentu a slouží ke spuštění programu.

Dále je ve složce src vytvořena složka code, v které je soubor *requests.js*. Zde je naimportována knihovna Axios, která komunikuje s backendem. Tato knihovna zasílá asynchronní HTTP dotazy na endpointy. Je vytvořen dotaz na adresu localhost:3000, která nám v závislosti na asynchronní funkci vrací odpověď ve formě dat, o které bylo požádáno, např. /quick-scan.

Backend

Jak již bylo zmíněno, s prací backendu je spojena knihovna Axios. Je zde Node server, který využívá knihovnu Axios navázanou na Nmap. Tento server vystavuje endpointy mezi backendem a frontendem. Vystavení endpointů je zprostředkováno pomocí API, která umožňuje interakci mezi dvěma systémy, frontendem a backendem. API endpoint představuje konec komunikačního kanálu mezi systémy [25]. Komunikace mezi systémy funguje pomocí *požadavků* a *odpovědi*. Server se spustí příkazem *npm run server*, pokud vše proběhlo v pořádku, je v Terminalu napsáno: *Server naslouchá na portu 3000* viz obr. 3.4.

```
PS C:\Users\petr.kurina\Documents\App_BP_final\App_Topo-master> npm run server

> easy_topo@0.1.0 server
> node server

Server naslouchá na portu 3000
```

Obr. 3.4: Ukázka správného naslouchání portu

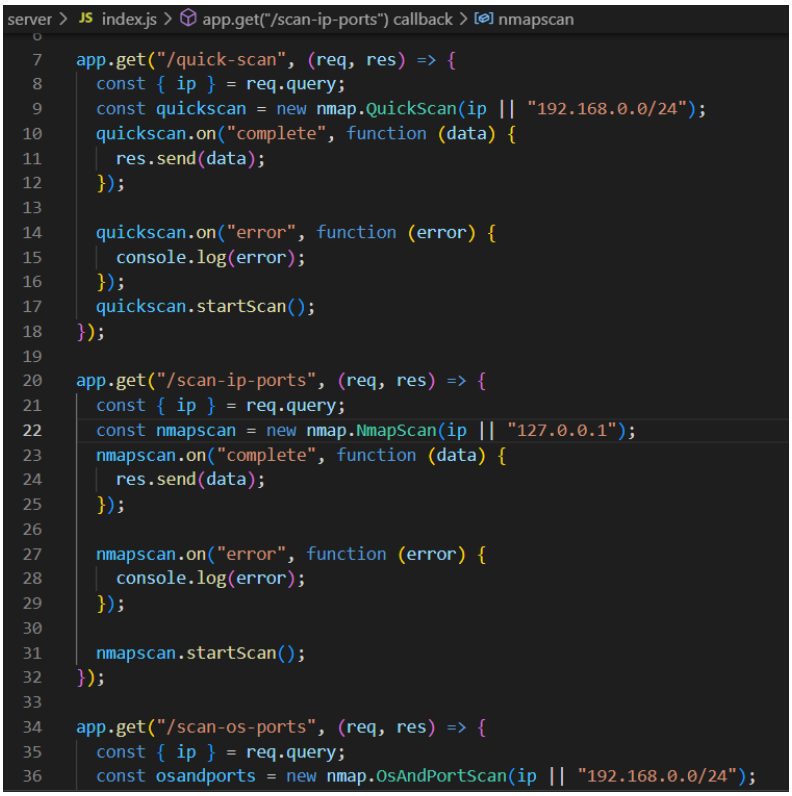
Zde viz obr. 3.5 je ukázka zobrazení odpovědi na serveru s koncovkou /quick-scan.

Obr. 3.5: Zobrazení odpovědi na serveru

4 Výsledky skenování

Cílem této kapitoly je prezentování výsledků otestovaných strojů, které byly získány za pomocí předdefinovaných skriptů nástroje Nmap. Bylo za potřebí stáhnout *npm nmap* balíček, který obsahuje několik skenů. Ze stroje Windows10 byla testována síť v rozsahu 192.168.0.0/24. Následně jsou výsledky prezentovány na webové stránce, která byla vytvořena v programovacím jazyce JavaScript ve frameworku Vue.js v programu Visual Studio Code, 64 bit. verze viz [26].

Pro skeny byly předefinovány rozsahy IP adressy viz obr. 4.1. Lze jej v souboru index.js upravit. Pro Quick scan a OS scan jsou zvoleny stejné rozsahy.

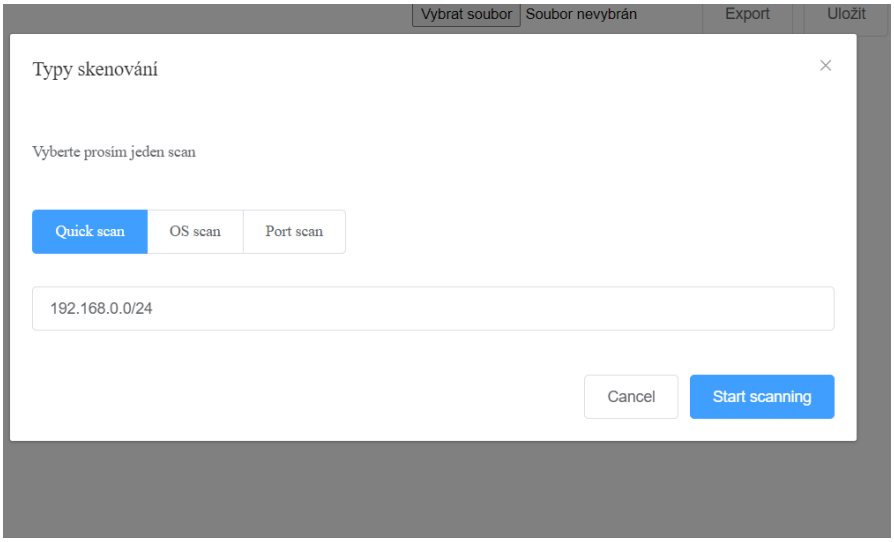


Obr. 4.1: Ukázka nadefinování rozsahu IP adres v souboru index.js

V testu Quick scan byl zvolen rozsah 192.168.0.0/24. Tento sken bude vracet všechny hosty, kteří jsou v dané síti aktuálně připojení bez dalších specifik. V testu Port scan byla zvolena adresa 127.0.0.1. a tento test vrací otevřené porty a služby na nich běžící. Posledním testem je OS scan, zde je zvolen rozsah 192.168.245.0/24. Sken vrací hosty s jejich otevřenými porty a službou na nich běžící. Lze zjistit i OS stroje, pokud sken pozná podle služeb o jaký OS se jedná.

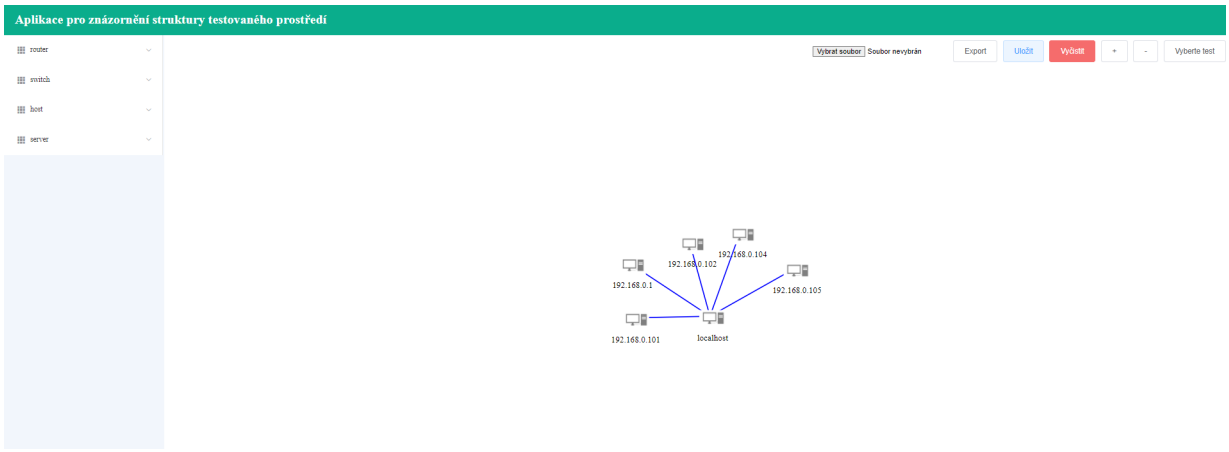
Uživatel si zvolí jeden ze tří skenů viz obr. 4.2 a může zadat libovolný rozsah IP adres nebo jednu konkrétní. Pokud IP adresa neleží v síti, je vrácena hodnota dané

IP a ikona localhosta. Pokud se IP adresa nachází v tomto rozsahu, je znázorněna topologie. Nutno podotknout, že localhost bude v topologii vždy a bude sloužit jako centrum. |



Obr. 4.2: Zobrazení možnosti spustit jeden ze tří skenů

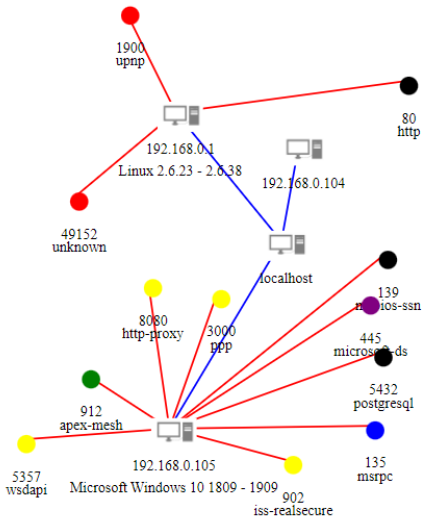
První vybraným skenen v tomto případě je Quick scan. Je to rychlá metoda skenování, která zobrazuje aktivní zařízení v síti bez dalších specifik jako jsou otevřené porty nebo OS systému. Okno pro zadání rozsahu nebylo nijak měněno a tak tedy rozsah zůstal na celou síť. Topologie je tedy zobrazena pouze jako struktura hostů v síti 4.3.



Obr. 4.3: Výsledky testu Quick scan v rozsahu 192.168.0.0/24

Z výsledků jde vidět, že skener odhalil celkem 5 zářízení v síti a localhosta, který bude v topologii pokaždé.

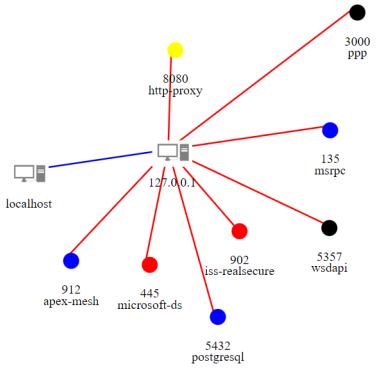
Druhým vybraným skenen je OS scan. Oproti ostatním skenům je to delší metoda skenování. Výsledky skenování vracejí hosty, kteří jsou v dané síti aktuálně připojeni, otevřené porty a k nim službu na nich běžící a snaží se o detekci OS stroje 4.4. V mnoha případech se OS nedaří odhalit, jelikož skener nepozná podle běžících služeb daný OS.



Obr. 4.4: Výsledky skenování sítě pouze 192.168.0.0/24 OS scanem

Skenování odhalilo celkem tři stroje a dvanáct portů. Výsledky testování o zařízení s IP adresou 192.168.0.104 nic specifického neodhalily, jen že je aktivní. Naopak u zařízení, na kterém běží toto testování, skener odhlalil devět otevřených portů. Lze si povšimnout portů 3000 a 8080, na kterých běží celá aplikace. Přesněji specikováno, na portu 3000 běží služba Point-to-point Protocol a na portu 8080 služba http-proxy.

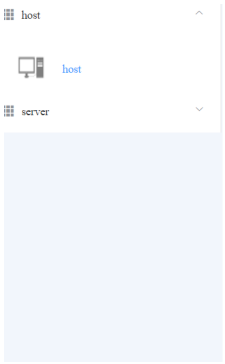
Posledním vybraným skenen je Port scan. Jedná se se o rychlé oskenování sítě, které zobrazí dostupné zařízení a jejich volné porty a služby, které se na daných portech nacházejí viz obr. 4.5.

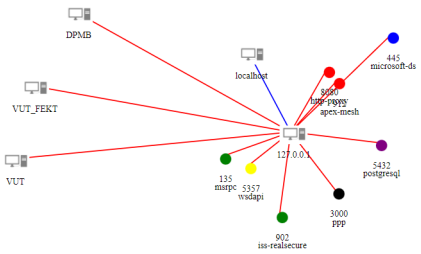


Obr. 4.5: Výsledky skenování sítě pouze pro 127.0.0.1 Port scanem

Z výsledků posledního skenování je patrné, že adresa 127.0.0.1 má osm otevřených portů. Z topologie lze vyčíst, že na portu 8080 běží služba http-proxy, protože frontend aplikace naslouchá na portu 8080. Následně na portu 3000 běží služba ppp, což je Point-to-point Protocol, kde pracuje server aplikace. Dále je možné si všimnout, že na portu 5432 běží služba postgresql.

Jako poslední bude prezentován výsledek testování Port Scanem, ke kterému jsou manuálně přiděláni 3 hosti s různými názvy.





Obr. 4.6:

Výsledek je totožný jako u Port Scanu, jen počet hostů byl navýšen o tři, u kterých jsou změněny jména z původního host.

Závěr

.

Tato bakalářská práce byla věnována návrhu aplikace pro znázornění struktury testovaného prostředí v programovacím jazyce Javascript. Teoretická část se zabývala popisem jazyka Javascript, frameworku Vue. Js. a hlavně návrhem výstupu webové aplikace. Obsah byl dále věnován penetračnímu testování, kde byly shrnuty důležité body a srovnány jednotlivé typy testování. Programový návrh aplikace byl vytvořen pomocí frameworku Vue.js. Tento framework je hojně využíván při práci s komponentami, kdy hlavní komponentou po App.vue je Topo.vue, která zajišťuje veškeré funkncionality programu. Další komponentou bylo ContextMenu, obsahující kontejner pro řešení kontextového menu prvků. Toto menu bylo vyvoláno pravým tlačítkem myši. Hlavními pilíři implementace programu jsou frontend a backend, které běží v programovacím jazyce Javascript a navzájem spolu kooperují. Backend je spojen s knihovnou Axios. Je zde Node server, který využívá knihovnu Axios navázanou na Nmap. Node server vystavuje endpointy, které jsou nachystané pro jednotlivé typy skenů. Využitá knihovna axios pro komunikaci s Nmapem vrací skenu dotazové hosty a jejich síťové prvky. Pro ukládání prvků slouží databáze LocalStorage. K implementaci frontendu bylo potřeba stažení modulu Node.js, který umožňuje spouštět JavaScript mimo prohlížeč. Dále pomocí příkazového řádku naistalován balíček @vue/cli a následně vytvořena aplikace. Stažení webové aplikace bylo zpřístupněno pomocí uložiště na webové stránce viz. [28]. Pro zpřehlednění aplikace bylo dále doporučeno stažení Visual Studio Code a k detekci sítě bylo potřebné stažení aplikace Nmap, aby bylo možné využít předemdefinové skripty. Aplikace běží na adrese localhost:8080 a server na adrese localhost:3000. Po obdržení požadavků server odešle aplikaci potřebná data a komunikace je ukončena. Podle typu požadavku server odpovídá na třech adresách localhost:3000/scan-ip-ports, localhost:3000/quick-scan, localhost:3000/scan-os-ports Čtvrtá kapitola je věnována výsledkům skenování. Třemi skeny byly provedeny celkem čtyři skenování topologie. Jako první byl proveden Quick Scan, který odhalil pět aktivních zařízení v síti. Druhé skenování patřilo OS Scan, která taktéž testovalo celou síť 192.168.0.0/24. Zde bylo zaznamenáno nejvíc volných portů, celkem dvanáct. U zařízení 192.168.0.104 nebyl zaznamenám ani jeden volný port, naopak u na Windows10 celkem devět volných portů. K nim patří např. port 3000 se službou Point-to-Point Protocol nebo protocol 8080, k němuž patří služba http-proxy. Byly zjištěny i dva OS systémy zařízení. U Windows10 skener správně informoval o tom, že se jedná o Windows10 a OS Linux u zařízení 192.168.0.1. Třetí skenování patřilo pouze skenu portů, kde byl testován pouze Windows10, tímto skenem byl odhalen o jeden port méně, než tomu bylo u OS Scanu. Jedná se o port 139, kde běží služba netbios-ssn. Poslední sken byl Port Scan, jenž

skenoval taky pouze Windows10 a následně byly k němu manuálně přetáhnuti myší 3 hosti, VUT, VUT_FEKT, DPMB, jenž byli dohromady s Windowsem10 spojení vazbou. V příloze byl vytvořen manuál, který uživatel může využít při instalaci projektu a k rychlejšímu pochopení aplikace.

Co se týká doporučení, který ze skenů vybrat, tak bych zvolil OS Scan, který trvá sice déle než ostatní, ale za to má odpovídající výsledky. Pokud by byl ale prvně proveden Quick Scan, jímž by se zjistilo velké množství zařízení v síti, doporučil bych zvolit pouze konkrétní IP adresu a ne celý rozsah, jako tomu bylo ve dvou případech v této práci. Aplikace by měla být přínosná zejména pro nenáročné uživatele a menší domácí sítě, jelikož se jedná o první verzi.

.

Literatura

[1] ANDREASEN, Esben, Liang GONG, Anders MØLLER, Michael PRADEL, Marija SELAKOVIC, Koushik SEN a Cristian-Alexandru STAICU. A Survey of Dynamic Analysis and Test Generation for JavaScript. *ACM Computing Surveys* [online]. 2017, **50**(5), 1-36 [cit. 2020-11-01]. ISSN 0360-0300. Dostupné z: doi:10.1145/3106739

[2] How Many Website Use Javascript In The World?. *Spritely* [online]. 2022 [cit. 2022-05-25]. Dostupné z: https://www.spritely.net/how-many-website-use-javascript-in-the-world/

[3] NGUYEN, Tran. *APPLYING VUE.JS FRAMEWORK IN DEVELOPING WEB APPLICATIONS* [online]. Ammattikorkeakoulu, 2020 [cit. 2022-05-28]. Bakalářská práce. LAB University of Applied Sciences.

[4] KOĎOUSKOVÁ, Barbora. JAVASCRIPT PRO ZAČÁTEČNÍKY: CO TO JE A JAK FUNGUJE. *Rascasone* [online]. Praha: Barbora Koďousková, 2022 [cit. 2022-05-24]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky

[5] HUANG, Yunhua, Tao LI, Lijia ZHANG, Beibei LI a Xiaojie LIU. JSContana: Malicious JavaScript detection using adaptable context analysis and key feature extraction. *Science Direct* [online]. Chengdu: Elsevier, 2021 [cit. 2022-05-27]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167404821000420

[6] ULIČNÝ, Vít. VUE.JS: TVORBA SVIŽNÝCH WEBŮ A VÝVOJ SINGLE-PAGE APLIKACÍ. *Rascasone* [online]. Praha, 2021 [cit. 2022-05-24]. Dostupné z: https://www.rascasone.com/cs/blog/vue-js-lehky-framework-idealni-pro-tvorbu-webovych-komponent

[7] NELSON, Brett. *Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch*. Eagan, Minnesota, USA: Apress, 2018. ISBN 978-1-4842-3780-9.

[8] KUMPULAINEN, Tomi. *Web application development with Vue.js* [online]. Jyväskylä, 2021 [cit. 2022-05-28]. Bakalářská práce. JAMK University, Information and Communication Technology.

[9] XIONG, Shiyong a Keqin FANG. Research On Persistent Management Model of Component State. *IEEE Xplore* [online]. Guangzhou: IEEE, 2020 [cit. 2022-04-12]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/9103772

[10] Getting Started. *Vuesjs* [online]. [cit. 2022-05-28]. Dostupné z: https://router.vuejs.org/guide/

[11] What is Vuetify. *Vuetifyjs* [online]. 2022 [cit. 2022-04-11]. Dostupné z: https://vuetifyjs.com/en/introduction/why-vuetify/#what-is-vuetify3f

[12] SHEBLI, Hessa Mohammed Zaher Al a Babak D. BEHESHTI. A study on penetration testing process and tools. *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* [online]. IEEE, 2018, 2018, , 1-7 [cit. 2020-11-01]. ISBN 978-1-5386-5029-5. Dostupné z: doi:10.1109/LISAT.2018.8378035

[13] HERZOG, Pete. *The Open Source Security Testing Methodology Manual 3* [online]. 3.02. 2010 [cit. 2022-05-28]. Dostupné z: https://www.isecom.org/OSSTMM.3.pdf

[14] EKA PRATAMA, I Putu Agus a Anak Agung BAGUS ARYA WIRADARMA. Open Source Intelligence Testing Using the OWASP Version 4 Framework at the Information Gathering Stage (Case Study: X Company). *I. J. Computer Network and Information Security* [online]. 2019, **11**(7), 1-5 [cit. 2022-05-28]. Dostupné z: doi:10.5815/ijcnis.2019.07.02

[15] *Průběh penetračního testování* [online]. , 1 [cit. 2020-12-05]. Dostupné z: https://mbi.vse.cz/public/cs/obj/TASK-362

[16] Nessus Vulnerability Scanner. *Tenable* [online]. 2020 [cit. 2020-11-29]. Dostupné z: https://www.tenable.com/products/nessus

[17] Linux Distributions. *Nmap* [online]. [cit. 2022-05-24]. Dostupné z: https://nmap.org/book/inst-linux.html

[18] *Wireshark* [online]. 2020 [cit. 2020-11-30]. Dostupné z: https://www.wireshark.org/

[19] Wireshark, aplikace pro sběr a analýzu paketů v síti. *Ubunlog* [online]. [cit. 2022-05-24]. Dostupné z: https://ubunlog.com/cs/wireshark-una-aplicacion-para-la-captura-y-analisis-de-paquetes-en-la-red/

[20] NIDHRA, Srinivas a Jagruthi DONDETI. BLACK BOX AND WHITE BOX TESTING TECHNIQUES —A LITERATURE REVIEW. *International Journal of Embedded Systems and Applications* [online]. June 2012, **2012**(2), 29-50 [cit. 2020-11-19]. Dostupné z: DOI : 10.5121/ijesa.2012.2204

[21] ČERMÁK, Miroslav. *Black box test* [online]. 25. 12. 2008n. l., 25. 12. 2008, **2008**, 1 [cit. 2020-11-19]. Dostupné z: https://www.cleverandsmart.cz/black-box-test/

[22] ČERMÁK, Miroslav. *White box test* [online]. 23.12. 2008n. l., 23. 12. 2008, **2008**, 1 [cit. 2020-11-19]. Dostupné z: https://www.cleverandsmart.cz/white-box-test/

[23] AYDOS, Murat, Çiğdem ALDAN, Evren COSSSKUN a Alperen SOYDAN. Security testing of web applications: A systematic mapping of the literature. *Computer and Information Sciences*. 2021, 1-18

[24] ŘENČOVÁ, Vlasta. Lekce 5 - AJAX v React. *Itnetwork* [online]. Praha, 2021 [cit. 2022-05-30]. Dostupné z: https://www.itnetwork.cz/javascript/react/zaklady/ajax-v-reactu/

[25] API Endpoints - What Are They? Why Do They Matter?. *Smartbear* [online]. 2022 [cit. 2022-05-31]. Dostupné z: https://smartbear.com/learn/performance-monitoring/api-endpoints/

[26] Download Visual Studio Code. *Visual Studio Code* [online]. 2022 [cit. 2022-05-28]. Dostupné z: https://code.visualstudio.com/sha/download?build=stable&os=win32-x64-user

[27] Index of /download/release/v16.14.0/. *Nodejs* [online]. 2022 [cit. 2022-05-28]. Dostupné z: https://nodejs.org/download/release/v16.14.0/node-v16.14.0-win-x64.7z

[28] KUŘINA, Petr. Aplikace. *Sharepoint* [online]. 2022 [cit. 2022-05-31]. Dostupné z: https://vutbr-my.sharepoint.com/:u:/g/personal/xkurin01_vutbr_cz/EU14CKmOcxhFmDzANoxvz0QBPyaZ0hy_w?e=DaUyzS

[29] Downloads. *Nodejs* [online]. 2022 [cit. 2022-05-29]. Dostupné z: https://nodejs.org/en/download/

[30] Nmap - Download. *Nmap.org* [online]. 2022 [cit. 2022-05-30]. Dostupné z: https://nmap.org/download

[31] Allow CORS: Access-Control-Allow-Origin. *Webstore* [online]. 2022 [cit. 2022-05-30]. Dostupné z: https://chrome.google.com/webstore/detail/allow-cors-access-control/lhobafahddgcelffkeicbaginigeejlf

[32] *Service Name and Transport Protocol Port Number Registry* [online]. [cit. 2020-12-07]. Dostupné z: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Seznam symbolů a zkratek

CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IPsec	Internet Protocol security
ISECOM	Institute for Security and Open Methodologies
ISAKMP	Internet Security Association and Key Management
OS	Operating System
Nmap	Network Mapper
RSA	Rivest, Shamir, Adleman
SNMPv3	Simple Network Management Protocol version 3
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics
TCP	primární přenosový protokol – Transmission Control Protocol
TLS	Transport Layer Security
TTY	Teletypewriterl
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access 2

Seznam příloh

A Návod na spuštění aplikace

1) Aplikace je volně dostupná ke stáhnutí Code -> Download Zip), viz stránka [28] . 2) Následně je aplikace rozbalena a spuštěna v editoru zdrojového kódu, lze doporučit Visual Studio Code viz [26] pro lepší přehlednost. 3) Dále je potřeba mít nainstalováno prostředí Node.js, který umožňuje spouštět JavaScript mimo webový prohlížeč [29]. 4) Při otevření aplikace editorem zdrojového kódu, uživatel spustí Terminal, kde příkazem *npm i* se nainstalují/aktualizují všechny potřebné balíčky. Je důležité spouštět aplikaci z kořenové složky. 5) Je potřeba mít na svém zařízení stáhnutou aplikaci Nmap viz [30], aby bylo možné detekovat síť. 6) Ve složce it server a v souboru *index.js* je předem nadefinovaný rozsah IP adres, které budou skenovány, tento rozsah je možné změnit a rozhodnout, zda se bude jednat pouze o 1 stroj (např. xxx.xxx.xxx.100 nebo 127.0.0.1) nebo o celý rozsah sítě (xxx.xxx.xxx.0/24). 7) Po nadefinování se v Terminálu zadá příkaz *npm run serve*,který spustí frontend aplikace a v dalším Terminálu příkázem *npm run server* se spustí backend aplikace. 8) Následně bude vybrána jedna z možností, na které bude zobrazena webová aplikace 3.2. Na adrese localhost:3000 pracuje backend aplikace. 9) Po zobrazení webové stránky lze spustit jeden ze tří scanů. První je QuickScan – vyskenuje hosta/y (pokud je zadán rozsah 0/24) v síti bez dalších informací. Druhým je PortScan – ten vyskenuje volné porty na hostovi/hostech dle rozsahu a posledním je OsScan – který skenuje volné porty i OS skenovaných zařízení. Všechny výsledky lze exportovat v souboru s příponou .json nebo ukládat do databáze LocalStorage. 10) Dále je možné strukturu různě upravovat, přetáhnutím lze vkládat nové prvky a editovat je pomocí kliknutí pravého tlačítka. Topologii je možno i smazat tlačítkem „Vyčistit“. 11) Je možné i importovat soubor s příponou .json, ta ale musí mít stejnou strukturu jako jeden ze tří vybíraných skenů, aby ho bylo možné vizualizovat. 12) Při použití prohlížeče Google Chrome může dojít k nevykreslení mapy topologie, bude tedy potřeba stáhnout rozšíření prohlížeče: Allow CORS: Access-Control-Allow-origin viz.[31] a povolit jej.