VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Viktória Matušicová

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

WEBOVÁ APLIKACE PRO ZOBRAZENÍ KYBERNETICKÝCH ÚTOKŮ V LOKÁLNÍCH SÍTÍCH
WEB APPLICATION FOR DISPLAYING CYBER ATTACKS IN LOCAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Viktória Matušicová**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Yehor Safonov**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Viktória Matušicová ***ID:*** 211803

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Webová aplikace pro zobrazení kybernetických útoků v lokálních sítích

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je návrh a implementace webové aplikace, která bude provádět analýzu kybernetických útoků na L2 a L3 vrstvách v lokálních sítích. Grafické uživatelské rozhraní bude mít formu interaktivní webové stránky umožňující vhodnou analýzu incidentů. Záznamy událostí budou odesílány z detekční sondy Raspberry Pi do centrálního úložiště Elasticsearch pomocí nástroje Filebeat. Jelikož se bude jednat o přenos a zpracování citlivých dat, funkcionální stránka webové aplikace musí splňovat základní bezpečnostní požadavky. V teoretické části nastudujte problematiku počítačových útoků vyskytujících na L2 a L3 vrstvách. Proveďte analýzu a srovnání vhodných webových frameworků (např. Django, ASP.NET, Spring MVC), popište specifika požadavků na aplikaci a modely případů užití, navrhněte architekturu budoucího řešení a grafické rozhraní aplikace (př. použitím Vue, React, Angular, Bootstrap). V rámci praktické části proveďte zpracování logů, realizujte experimentální pracoviště (sestavené ze směrovače, Raspberry Pi, webového serveru a koncových zařízení) a implementujte grafické uživatelské rozhraní dle návrhu. Výsledné řešení otestujte na nejméně třech útocích simulovaných v experimentálním pracovišti.

DOPORUČENÁ LITERATURA:

[1] MIKOWSKI, Michael; POWELL, Josh. Single page web applications: JavaScript end-to-end. Manning Publications Co., 2013.

[2] RAVINDRAN, Arun. Django Design Patterns and Best Practices. Packt Publishing. ISBN 978-1-78398-664-4.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Yehor Safonov

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

ABSTRAKT

Informačná sféra sa neustále a najmä extrémne rýchlo rozvíja. S touto expanziou je samozrejme spojený aj nárast rizík využívania internetu jeho užívateľmi. Objavujú sa zraniteľné miesta a iné nebezpečenstvá, ktoré umožňujú neoprávneným osobám vniknúť do integrity chránených infraštruktúr.

Hlavným cieľom bakalárskej práce je vytvorenie nástroja, ktorý umožňuje správcovi systému prevádzať analýzu koncových staníc lokálnej siete. Pomocou webovej aplikácie si je správca schopný zobraziť všetky počítačové útoky vykonané na jeho počítačovej infraštruktúre a tak môcť zaviesť protiopatrenia, ktoré zabránia vzniku negatívnych dopadov na celú infraštruktúru.

Z teoretického hľadiska sa bakalárska práca zameriava na problematiku počítačových útokov na dátovej a sieťovej vrstve modelu ISO/OSI. Následne sa zaoberá štruktúrou zapojenia pracoviska a webovej aplikácie. V poslednej časti sa práca venuje návrhu danej webovej aplikácie a jej integrácii do experimentálneho pracoviska.

V praktickej časti je dôraz kladený na realizáciu pracoviska a webovej aplikácie. Praktická časť je rozdelená na dve implementačné fázy. V prvej fáze je experimentálne pracovisko zapojené v rámci lokálnej siete. Webová aplikácia sa v tejto fáze zameriava na vývoj serverovej časti – prácu s databázami.

V druhej fáze implementácie je experimentálne pracovisko prenesené do reálnej podoby. Do webovej aplikácie sú pridané rôzne grafické zobrazenia, filtrovanie a sekcia pre nastavenia užívateľa. Najväčší dôraz je kladený na bezpečnosť celej aplikácie – na prihlasovací systém, konfiguračné nastavenia servera a klienta.

Po zapojení experimentálneho pracoviska a následnom prepojení s webovou aplikáciou je vykonané testovanie funkčnosti celého riešenia pomocou troch rôznych počítačových útokov.

Na konci práce je vytvorený stručný záver a zhrnutie bakalárskej práce.

KĽÚČOVÉ SLOVÁ

detekčný systém, Elasticsearch, Filebeat, IDS, IPS, počítačový útok, Raspberry Pi, Surikata, webová aplikácia, webový framework.

ABSTRACT

The information sphere is constantly and rapidly developing. This expansion means an increase in the risks of the Internet use. Vulnerabilities and other threats are emerging that provide an opportunity for unauthorized users to penetrate the integrity of protected infrastructures.

The main goal of the bachelor's thesis is to create a tool that allows the system administrator to perform the analysis of end stations in the local network. With the help of the web application, the administrator is able to view all computer attacks performed on his computer infrastructure. That makes it possible for him to implement countermeasures which will improve performance and security of the entire infrastructure.

From a theoretical point of view, the bachelor thesis is focused on the issue of computer attacks on the data layer and network layer of the ISO/OSI model. Subsequently, it is focused on the structure of workplace involvement and web application. In the last part the work is focused on the design of the web application and its integration into the experimental workplace.

Emphasis during the practical part is placed on the implementation of the workplace and web application on the local network. The practical part is divided into two implementation groups. Initially the experimental workplace is implemented within the local network. Here the web application focuses on the development of the server side – working with databases.

In the second phase of implementation the experimental workplace is transferred into a real form. Following application features are added: various graphical displays, filtering and a section for user settings. Large emphasis is placed onto the security of the entire application – login system, server and client configuration settings.

After the experimental workplace is connected with the web application, the functionality of the entire solution is tested by three different computer attacks.

At the end of the thesis a brief conclusion and summary of the bachelor's thesis is established.

KEYWORDS

computer attack, detection system, Elasticsearch, Filebeat, IDS, IPS, Raspberry Pi, Suricata, web application, web framework.

MATUŠICOVÁ, Viktória. *Webová aplikace pro zobrazení kybernetických útoků v lokálních sítích*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 74 s. Bakalárska práca. Vedúci práce: Ing. Yehor Safonov

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Viktória Matušicová
VUT ID autora:	211803
Typ práce:	Bakalárska práca
Akademický rok:	2020/21
Téma závěrečnej práce:	Webová aplikace pro zobrazení kybernetických útoků v lokálních sítích

Vyhlasujem, že svoju záverečnú prácu som vypracovala samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autorka uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušila autorské práva tretích osôb, najmä som nezasiahla nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomá následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rada by som sa poďakovala vedúcemu mojej bakalárskej práce, pánovi Ing. Yehorovi Safonovi, najmä za čas, ktorý mi venoval, za odborné vedenie, pravidelné konzultácie, podnetné návrhy a pripomienky k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Informačná sféra sa neustále a najmä extrémne rýchlo rozvíja. S touto expanziou je samozrejme spojený aj nárast rizík využívania internetu jeho užívateľmi. Objavujú sa zraniteľné miesta a iné nebezpečenstvá, ktoré umožňujú neoprávneným osobám vniknúť do integrity chránených infraštruktúr s cieľom ukradnúť, poškodiť alebo zničiť informácie. Kvôli narastajúcim rizikám bolo nutné vytvoriť odbor informačnej sféry s cieľom ochrániť užívateľov a ich dáta – *Informačnú bezpečnosť.* [1]

Predmetom ochrany informačnej bezpečnosti sú informácie bez ohľadu na to, v akej forme sú uložené. Jej cieľom je zaistenie dôvernosti, dostupnosti, celistvosti, autentizácie, autorizácie a nepopierateľnosti informácií [2]. Kvôli obavám z bezpečnostných rizík vyhľadávajú súkromné spoločnosti čo najkvalitnejšie monitorovacie systémy svojej siete obsahujúcej ich citlivé dáta [2]. Z daného dôvodu je potrebné vytvoriť nástroj na zobrazenie stavu ich infraštruktúry v lokálnej sieti.

Hlavným prínosom bakalárskej práce je vytvorenie nástroja, ktorý by umožnil správcovi systému prevádzať analýzu koncových staníc lokálnej siete. Pomocou webovej aplikácie si je správca schopný zobraziť všetky počítačové útoky vykonané na lokálnej sieti, vďaka čomu je možné zaviesť protiopatrenia, ktoré zabránia vzniku negatívnych dopadov na celú infraštruktúru.

Bakalárska práca sa rozdeľuje na teoretickú a praktickú časť. Skladá sa z piatich kapitol, z ktorých každá predstavuje jeden celok logicky prepojený s ostatnými.

Prvá kapitola sa zaoberá problematikou počítačových útokov. Na začiatku sú vysvetlené základné pojmy nutné na porozumenie danej tematiky (časť 1.1). V ďalšej časti práce sú objasnené spôsoby detekcie a prevencie prienikov, ich rozdelenie a najznámejšie príklady detekčných systémov používaných v praxi (časť 1.2). Poslednou súčasťou prvej kapitoly je charakteristika počítačových útokov. Obsahuje popis najznámejších útokov na sieťovej a dátovej vrstve modelu ISO/OSI (časť 1.3), z ktorých sa vybrané z nich použijú pri testovaní výstupu praktickej časti pomocou detekčného systému zapojeného na pracovisku.

Druhá kapitola sa zaoberá charakteristikou riešenia. Riešenie je systematicky rozdelené na dva celky. V časti 2.1 je zobrazená architektúra celého pracoviska výslednej práce, obsahuje vysvetlenie jednotlivých komponentov a ich logické prepojenie. Kapitola 2.2 sa zaoberá architektúrou zapojenia webovej aplikácie a vysvetlením využitých komponentov v danej architektúre.

Tretia kapitola sa venuje návrhu budúceho riešenia webovej aplikácie. Na začiatku kapitoly je vykonaná analýza webovej aplikácie, ktorá sa skladá z tvorby modelu prípadu užitia (kapitola 3.1.1) a špecifikácie požiadaviek na jej funkcionalitu (kapitola 3.1.2). Obsahuje porovnanie dostupných najpoužívanejších frameworkov slúžiacich na tvorbu webových aplikácií, ich hlavné rozdelenie a zhrnutie ich

výhod a nevýhod v praxi (časť 3.2). Ďalšou súčasťou kapitoly je vykonanie prieskumu dostupných SIEM nástrojov z dôvodu získania prehľadu, akým spôsobom je vhodné incidenty zobraziť (kapitola 3.3). Kapitola obsahuje prvotný návrh riešenia vytvorený na základe tohto prieskumu (časť 3.4).

Implementácia praktickej časti práce je rozdelená do dvoch fáz. V rámci prvej fázy (kapitola 4.2) je vytvorený základ serverovej časti webovej aplikácie a podľa architektúry uvedenej v sekcii 2.1 je experimentálne pracovisko zapojené na lokálnej sieti.

V druhej fáze (kapitola 4.3) implementácie je webová aplikácia rozšírená integrovaním GUI (angl. *Graphical User Interface*). Aplikácia je rozdelená na tri sekcie (kapitola 4.3.1) – sekcia *Dashboard*, *Investigation* a *Settings*. Okrem tvorby GUI je dôraz kladený na implementovanie zabezpečenia aplikácie najnovšími bezpečnostnými technikami (kapitola 4.3.2). V závere kapitoly je experimentálne pracovisko zapojené na lokálnej sieti prenesené do reálnej podoby (kapitola 4.3.3).

Obsahom poslednej kapitoly, kapitoly piatej, je testovanie výsledného riešenia bakalárskej práce. Na jej začiatku je uvedený postup testovania (kapitola 5.1). Kapitola pokračuje vysvetlením priebehu jednotlivých útokov vybraných na overenie funkčnosti zapojenia experimentálneho pracoviska s webovou aplikáciou (kapitola 5.2).

Poslednou súčasťou práce je jej záver. Obsahuje zhodnotenie a analýzu dosiahnutých výsledkov.

1 Problematika počítačových útokov

Za počítačový útok sa v informatike označuje neoprávnený prístup na užívateľský počítač s cieľom ukradnúť, pozmeniť alebo zničiť užívateľské dáta. S danou problematikou úzko súvisí bezpečnosť na internete. Pojem bezpečnosť na internete je pomerne široký. Môžeme si pod ním predstaviť súbor opatrení, ktoré majú za cieľ znemožniť útočníkovi získanie neoprávneného prístupu k našim súkromným dátam. [3]

Na začiatku kapitoly sú vysvetlené definície pojmov potrebných na pochopenie z dôvodu ich využitia v ďalších kapitolách. Nadchádzajúca časť sa venuje spôsobom detegovania počítačových útokov a ich systémom.

1.1 Úvod do problematiky

V úvode kapitoly problematiky počítačových útokov je dôležité zoznámiť sa s používanými pojmami, ktoré úzko súvisia s tematikou. Pojmoslovie je v každom odbore významným prostriedkom zhodného chápania obsahu, aby pri čítaní danej práce nevznikli problémy s ich významom v texte. K základným pojmom patrí:

1. Administrátor (angl. *Administrator*) – správca siete, ktorý má spravidla najvyššie práva prístupu [4].
2. Antivírus (angl. *Antivirus*) – program slúžiaci na vyhľadávanie počítačových vírusov, liečenie napadnutých súborov, zálohovanie, obnovu systémových oblastí na disku [4].
3. Autentizácia (angl. *Authentication*) – proces overenia identity subjektu [3].
4. Autorizácia (angl. *Authorization*) – proces overenia oprávnení pre danú úlohu subjektu [3].
5. Databáza (angl. *Database*) – obsahuje súhrn dát podľa štruktúry, v ktorej sú popísané ich vlastnosti a vzťahy medzi entitami slúžiace pre iné aplikačné oblasti [4].
6. Firewall (angl. *Firewall*) – systém bezpečnostných opatrení, ktoré majú zabrániť neoprávnenému prístupu k počítaču, službám... Môže byť realizovaný ako hardware alebo software [3, 4].
7. Incident (angl. *Incident*) – nežiaduca udalosť, t.j. udalosť spojená s uskutočnením útoku [4].
8. IP adresa (angl. *IP Address*) – číslo, ktoré jednoznačne identifikuje zariadenie v počítačovej sieti [4].
9. MAC adresa (angl. *MAC Address*) – jednoznačný identifikátor zariadenia pridelený výrobcom [4].
10. Monitorovanie (angl. *Monitoring*) – nepretržitá kontrola, dozor s cieľom identifikovania zmien [4].

11. Paket (angl. *Packet*) – blok dát prenášaných v počítačovej sieti [4].
12. Podvrhnutie (angl. *Spoofing*) – činnosť s cieľom oklamať užívateľa pomocou falošnej identity [3].
13. Užívateľ (angl. *User*) – fyzická osoba, ktorá je na základe pridelenej roly oprávnená využívať služby informačného systému [4].
14. Útočník (angl. *Attacker*)– osoba, ktorá vykonáva aktivity bez oprávnenia [4].
15. Útok (angl. *Attack*)– incident iniciovaný útočníkom [4].
16. Zaplavenie (angl. *Flooding*) - nefunkčnosť prevádzky, ktorá umožní vyradenie ochranných mechanizmov z činnosti [4].
17. Zneužitie (angl. *Exploit*) – chyba v programe, ktorá povoľuje užívateľovi používanie programov nepovoleným spôsobom [3].
18. Zraniteľnosť (angl. *Vulnerability*) – slabé miesto zariadenia, ktoré môže byť zneužité [3].

1.2 Spôsoby detekcie a prevencie prienikov

Dôležitosť ochrany všetkých informačných systémov pred rôznymi druhmi počítačových útokov dnes už nie je potrebné zdôrazňovať. Detekcia útokov tak zohráva významnú úlohu pri zabezpečení siete. Je to proces monitorovania udalostí v počítačovom systéme a ich analyzovanie na príznaky možných incidentov.

V nasledujúcich častiach kapitoly sú rozobrané jednotlivé technológie určené na detekciu a prevenciu prienikov. Je podané vysvetlenie, čo systémy predstavujú a aké sú ich rozdiely.

1.2.1 Systém detekcie prienikov (IDS)

IDS (angl. *Intrusion Detection System*) môže byť definovaný ako súbor nástrojov a metód, ktoré nám pomáhajú identifikovať a hlásiť neautorizované sieťové aktivity. Deteguje škodlivé pakety v sieti a vytvára upozornenia, ale nemôže blokovať ich vstup zo siete. [5]

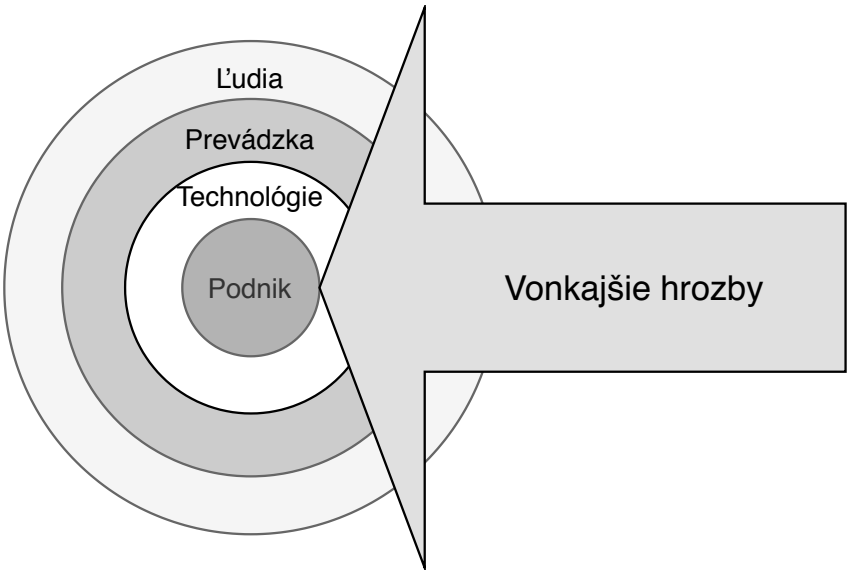
Uzlovo orientovaný systém detekcie prienikov HIDS (angl. *Host-Based Intrusion Detection System*) je typ IDS, ktorý vyžaduje softvér umiestnený na tomto systéme a môže skenovať aktivitu všetkých uzlových spojov. Zapíše udalosť do bezpečnostnej databázy a preverí, či sa tieto udalosti nezhodujú so záznamami závadných udalostí v databáze. [5, 6]

Sieťovo orientovaný systém detekcie prienikov NIDS (angl. *Network-Based Intrusion Detection System*) je typ IDS systému, ktorý sa obyčajne zaraďuje do siete sériovo a analyzuje sieťové pakety pomocou metódy, ako je napr. zrkadlenie portov. [5, 6]

1.2.2 Systém prevencie prienikov (IPS)

IPS (angl. *Intrusion Prevention System*) technológia je pomerne nová a stále sa vyvíjajúca. Svojím nastavením sa podobá IDS systému. IPS môže byť uzlovo orientovaný prevenčný systém HIPS (angl. *Host-Based Intrusion Detection System*), ktorý najlepšie pracuje v ochranných aplikáciách, alebo sieťovo orientovaný prevenčný systém NIPS (angl. *Network-Based Intrusion Detection System*). Užívateľské činnosti by mali zodpovedať činnostiam v preddefinovaných znalostných databázach. Ak nejaká činnosť nie je v zozname, IPS zabráni jej uskutočneniu. Logika IPS sa typicky aplikuje predtým, ako je vykonaná v pamäti. [6]

Detekcia prienikov, prevencia prienikov a firewall sú jednotlivými časťami celkového ochranného systému, ktorý je nainštalovaný na zariadení alebo systéme. Sú to rôzne druhy technológií, ktoré spolu dokážu zaistiť určitú bezpečnosť. To znamená, že sieť by mala mať niekoľko bezpečnostných vrstiev, ktoré dohromady vytvárajú celkovú bezpečnostnú stratégiu organizácie. Obrázok 1.1 zobrazuje štruktúru tzv. hĺbkového riešenia, ktoré ochraňuje sieť na viacerých úrovniach. [6]



Obr. 1.1: Hĺbková ochrana [6]

1.2.3 Význam IDS a IPS

IDS a IPS sú dôležité pre veľa organizácií, od malých firiem až po nadnárodné korporácie. Majú množstvo výhod [7]:

- Schopnosť vyrovnať sa s veľkým objemom dát.
- Schopnosť výstrahy takmer v reálnom čase, čo pomáha znížiť potenciálne poškodenie.

- Zabudovaná podpora pre súdne riadenie.
- Zabudovaná schopnosť ohlasovacích funkcií.
- Automatizovaná odozva, napr. odhlásenie užívateľa, ponuka automatizovaných skriptov.

Obe technológie majú svoje miesto v bezpečnostnom programe, pretože vykonávajú oddelené funkcie. V nasledujúcej tabuľke 1.1 sú zobrazené hlavné rozdiely medzi nimi.

Tab. 1.1: IDS verzus IPS [7]

IDS	**IPS**
Nástroj na detekciu a monitorovanie	Systém na kontrolu a prevenciu
Nemôže analyzovať šifrovanú prevádzku	Vhodnejší pre ochranné aplikácie
Výstrahu vydávajúci systém	Blokujúci systém
Potrebný ďalší systém na sledovanie výsledkov	Po nájdení hrozby sa databáza automaticky aktualizuje

Detekčné metódy IDS a IPS môžeme vo všeobecnosti rozdeliť na dva základné druhy – metódy založené na signatúrach a metódy založené na detekcii anomálií.

Detekcia útokov založená na signatúrach

Detekčný systém je založený na vyhľadávaní fixných znakov v pakete, ktoré sú typické pre určité druhy útokov. Mechanizmus obsahuje svoj vlastný slovník, ktorý sa automaticky dopĺňa. Ak pri komunikácii nájde značku, ktorá je zhodná so značkou v slovníku, vykoná opatrenia. Systém je veľmi jednoduchý na detekciu útokov, je aplikovateľný na všetky protokoly a umožňuje koreláciu[1] zraniteľností podľa vzorky. [8]

Detekcia útokov založená na anomáliách

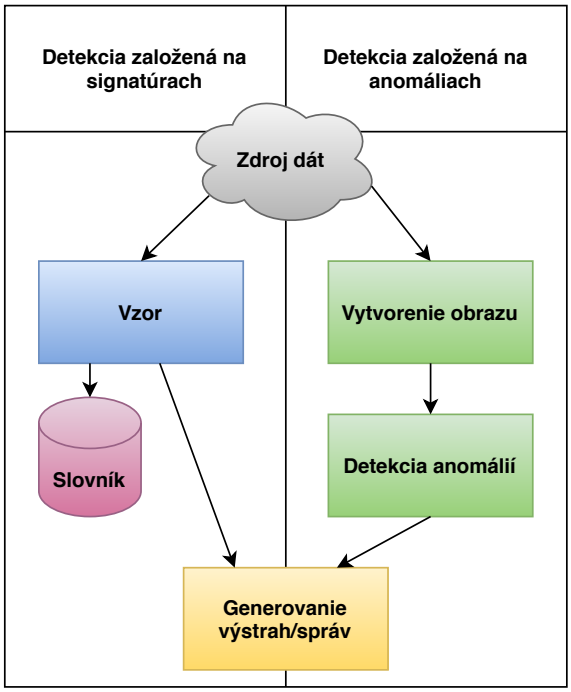
Kľúčový rozdiel medzi detekciou anomáliami a inými koncepciami je, že koncepcia založená na anomáliách nedefinuje iba činnosti, ktoré nie sú povolené, ale aj také, ktoré povolené sú [6].

Na obrázku 1.2 je znázornený princíp jednotlivých metód.

1.2.4 Detekčné systémy

Ako bolo vysvetlené v kapitole 1.2.1, systémy sieťovej detekcie prienikov (NIDS) slúžia na monitorovanie siete, identifikáciu a nahlásenie entít, ktoré sa pokúšajú

[1] Vzájomný vzťah medzi dvoma javmi, pojmami a pod.



Obr. 1.2: Princíp detekcie pomocou signatúr a anomálií [10]

narušiť dôvernosť, integritu alebo dostupnosť zdrojov. SolarWinds Security Event Manager, Snort, Surikata, Sagan a Kismet patria k populárnym technológiám, ktoré dané monitorovanie a detekciu útokov v sieti umožňujú. Uvedené technológie sú charakterizované v ďalších častiach kapitoly.

SolarWinds Security Event Manager

SolarWinds Security Event Manager (SEM) je považovaný za komerčný vysoko inteligentný prístup k detekcii hrozieb. Je založený na zhromažďovaní protokolov systému detekcie vniknutia do siete, vďaka čomu zhromažďuje informácie o typoch a množstvách jednotlivých útokov. Dané informácie sú ďalej integrované do protokolov infraštruktúry, a tým vytvárajú rozsiahlu sieť údajov, ktoré optimalizujú bezpečnostné systémy IDS. [11]

Pomocou SEM je možné identifikovať problémové zariadenia v sieti, použiť dáta na vytvorenie správ o hodnotení rizika a identifikovať hrozby pred ich vypuknutím. Je dostupný pre operačné systémy Windows, Unix, Linux a MacOS. [11]

Snort

Snort je bežne dostupná, bezplatná technológia systému IDS/IPS. Snort vie byť nakonfigurovaný v troch rôznych režimoch: režim *sniffer*, paketový *logger* a režim

detekcie sieťového prieniku. *Sniffer* mód číta pakety prichádzajúce zo siete a zobrazuje ich ako nepretržitý prúd na konzole. Režim detekcie sieťového prieniku je komplexný a povoľuje technológii Snort analyzovať sieťovú prevádzku pomocou definovaných pravidiel a následne podľa nich vhodne zareagovať. Posledný režim, paketový *logger*, zaznamenáva pakety na disk. [13]

Snort sa skladá z nasledujúcich komponentov [12]:

- *Packet Detector* – ukladá pakety zo sieťových rozhraní a pripravuje ich na spracovanie.
- *Prepocesor* – deteguje anomálie v hlavičkách paketu.
- *Detection Engine* – aplikuje pravidlá na pakety, najdôležitejší komponent.
- *Logging and Alerting Systems* – vytvára výstrahy.
- *Output modules* – spracováva výstrahy do výstupu.

Surikata

Zatiaľ čo bol Snort najpopulárnejší a najrozšírenejší systém, potreba Surikaty sa zvyšovala, pretože technológia Snort bola limitovaná kvôli jednovláknovej architektúre.

Surikata je bežne dostupný, bezplatný a robustný detekčný systém IDS/IPS založený na viacerých vláknach. Obsahuje rozsiahlu a výkonnú sadu pravidiel (obr. 1.3) a databázu slúžiacu na detekciu širokého spektra sieťových hrozieb [13].

```
root@ubuntu:~# nano /var/lib/suricata/rules/local.rules
drop tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%27"
; sid:100000011; )
drop tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%22"
; sid:100000012; )


alert tcp any any -> any 80 (msg: "AND SQL Injection Detected"; content: "and" ; nocas
e; sid:100000060; )
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected"; content: "or" ; nocase;
sid:100000061; )
```

Obr. 1.3: Príklad pravidiel určených na detekciu útoku *SQL Injection* [12]

Sagan

Sagan je voľne dostupný vysoko rýchlostný nástroj bežiaci na operačných systémoch Unix (napr. Linux), ktorý slúži na analýzu logov (časť 2.1.1) a koreláciu protokolov v reálnom čase. Je napísaný v jazyku C. Na zabezpečenie vysoko výkonnej analýzy protokolov a udalostí využíva viacvláknovú architektúru. Štruktúrou a pravidlami sa podobá na systémy Snort a Surrikata IDS/IPS, vďaka čomu je schopný korelácie

s údajmi Snort. Podporuje rôzne výstupné formáty analýzy, viacriadkové protokoly a časové upozornenie. [14]

Sagan povoľuje vykonávanie skriptov po detekcii hrozby, automatický firewall a varovanie. Podporuje viacero výstupných formátov. Nevýhodou daného nástroja je, že obsahuje príliš veľa funkcií, čo z neho pre nových užívateľov robí príliš komplikovaným. [11]

Kismet

Kismet je voľne dostupný a bezdrôtový IDS, čo znamená že sa sústredí na bezdrôtové protokoly ako Bluetooth a Wi-fi. Sleduje a odhaľuje neoprávnené prístupy a deteguje medzery v konfigurácii siete. Pomocou pluginov je možné rozšíriť funkčnosť webového užívateľského rozhrania prostredníctvom vylepšení na strane prehliadača a servera, ako napr. plugin Kismet (pridanie živého mapovania). [11]

Hlavnou nevýhodou nástroja Kismet je príliš veľká doba vykonávania prehľadávania sietí a jeho obmedzená podpora pre operačný systém Windows. [11]

1.3 Charakteristika počítačových útokov

Pred uvedením charakteristiky počítačových útokov na L2 a L3 vrstve je nutné vysvetliť, čo jednotlivé vrstvy znamenajú. Dané vrstvy sú súčasťou referenčného modelu ISO/OSI. Definuje hierarchickú architektúru, ktorá logicky rozdeľuje funkcie potrebné na komunikáciu medzi systémami.

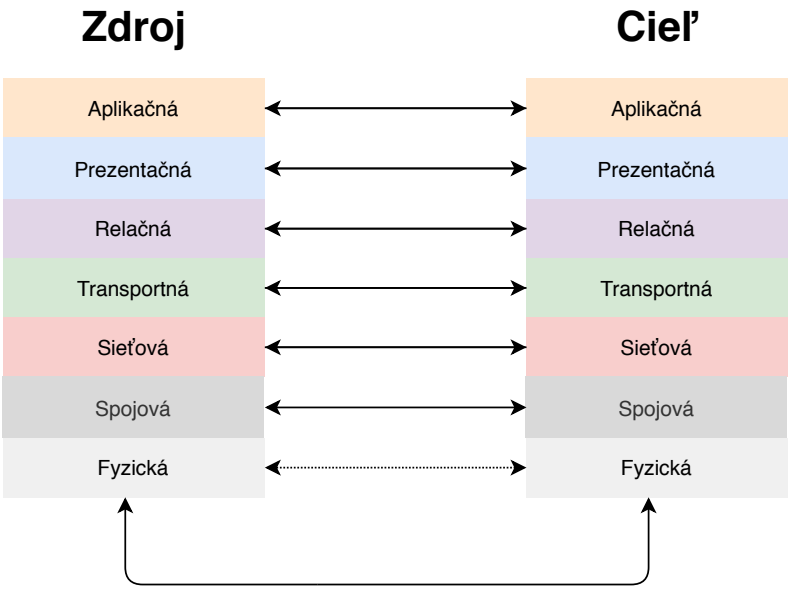
1.3.1 Referenčný model ISO/OSI

Referenčný model ISO/OSI je konceptuálny model, ktorý poskytuje rámec pre špecifikáciu a identifikáciu sieťových funkcií (obr. 1.4). Je veľmi užitočný pre vývoj sieťových procesov, a preto pochopenie základného konceptu modelu ISO/OSI je nevyhnuté pre porozumenie funkcií počítačových sietí. [15]

Model sa skladá zo siedmich vrstiev. Úlohou každej z nich je poskytnúť služby susedným vyšším vrstvám. Každá vrstva plní určité funkcie [15]:

1. **Fyzická vrstva (L1)** - špecifikuje zariadenia a protokoly potrebné k fyzickému zapojeniu sietí (konektory, prepínače apod.).
2. **Spojová vrstva (L2)** - poskytuje rozdelenie dát na transportné jednotky a spojenie medzi dvoma susednými systémami.
3. **Sieťová vrstva (L3)** - stará sa o smerovanie v sieti a adresovanie, poskytuje spojenie medzi dvomi systémami, ktoré spolu priamo nesúvisia.
4. **Transportná vrstva (L4)** - poskytuje spoľahlivý prenos dát s požadovanou kvalitou, prevádza transport adries na sieťové.

5. **Relačná vrstva (L5)** - umožňuje vytvorenie a ukončenie relácie medzi dvomi systémami a dohliada na synchronizáciu spojení.
6. **Prezentačná vrstva (L6)** - pretvára dáta do tvaru, aký používajú aplikácie.
7. **Aplikačná vrstva (L7)** - poskytuje prístup aplikačných procesov ku komunikačnému systému.



Obr. 1.4: Referenčný model ISO/OSI [15]

Nasledujúce časti 1.3.2 a 1.3.3 sú zamerané na charakteristiku útokov vyskytujúcich sa na spojovej a sieťovej vrstve.

1.3.2 Útoky na L2 vrstve

Pod názvom sekcie „Útoky na L2 vrstve“ sú predstavované útoky na spojovej (dátovej) vrstve referenčného modelu ISO/OSI (časť 1.3.1).

ARP Spoofing

Zneužitie protokolu ARP[2] (angl. *Address Resolution Protocol*) je technika umožňujúca útočníkovi vydávať sa v sieti za iný počítač.

Na každom smerovači musí byť po ceste IP paket opatrený MAC adresou. Smerovač s protokolom ARP zasiela požiadavku všetkým počítačom v sieti a očakáva MAC adresu príjemcu paketu. Smerovač MAC adresu uloží spolu s IP adresou do ARP tabuľky a stanovuje vďaka nej, kam sa majú zasielať pakety pre danú IP adresu. [16]

[2]Protokol slúžiaci na preklad IP adresy na adresu MAC.

Princípom útoku je neustále odosielanie MAC adresy útočníka na smerovač, ktorý si danú falošnú MAC adresu uloží do svojej tabuľky a pakety zasiela na ňu.

Nasledujúce metódy sú odporúčané kvôli prevencii a obrane pred ARP spoofing útokmi [17, 18]:

- **Filtrovanie paketov:** Filter skontroluje pakety prenášané sieťou. Blokuje pakety, ktoré majú konfliktnú zdrojovú adresu (pakety mimo lokálnej siete, ktoré zobrazujú zdrojové adresy z vnútra siete a naopak).
- **Použitie statickej ARP tabuľky:** Preložené IP adresy na adresy MAC môžu byť staticky uložené v lokálnej ARP tabuľke. V takomto prípade môžu byť počítače nastavené na ignorovanie všetkých paketov s ARP odpoveďou.
- **Použitie kryptografických sieťových protokolov:** Komunikačné protokoly TLS (angl. *Transport Layer Security*), SSH (angl. *Secure Shell*) a HTTPS (angl. *Hypertext Transfer Protocol Secure*) posilňujú prevenciu proti útokom šifrovaním údajov pred ich prijatím.

VLAN Hopping

Preskakovanie virtuálnej lokálnej siete VLAN (ang. *Virtual Local Área Network*) je metóda útoku, ktorej cieľom je získanie prístupu k prevádzke iných hostiteľov VLAN. Existujú dve metódy útoku, *Switch Spoofing* a *Double Tagging*.

Pri útoku **Switch spoofing** útočník odosiela pakety a pokúša sa pomocou smerovača dostať do zariadenia. Ak je úspešný, získa prístup do celej VLAN.

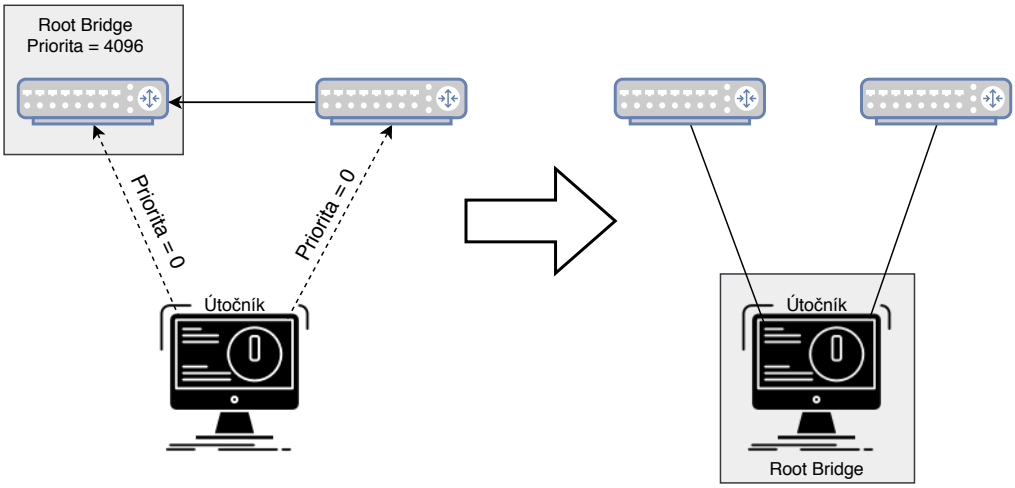
Pri útoku **Double tagging** útočník odosiela rámec s dvomi značkami. Keď ho smerovač prijme, prvá značka sa odstráni a druhá falošná získa útočníkovi prístup do VLAN. [19]

K hlavným odporúčaniam na prevenciu proti útoku VLAN hopping patrí [19]:

- Nepoužívať predvolené názvy VLAN, ako napr. `VLAN1`.
- Vložiť odpojené porty smerovača do nepoužívanej VLAN.
- Aktualizovať softvér.
- Predvoliť nepoužívanú VLAN pre všetky prijímané rámce.

STP Attack

STP (angl. *Spanning Tree Protocol*) je sieťový protokol slúžiaci na odstraňovanie nekonečných slučiek v sieti LAN. Protokol vytvára zo smerovačov topológiu a jednému smerovaču pridelí rolu, tzv. *Root Bridge*. Rola sa stanovuje na základe hodnoty priority. Najnižšia možná hodnota je 4096. Princíp útoku spočíva v zmanipulovaní root bridge smerovača tak, aby si myslel, že stanica útočníka má nižšiu hodnotu priority, a tým získa úlohu root bridge. Všetka prevádzka siete tak následne prechádza útočníkovým smerovačom (obr. 1.5). [20]



Obr. 1.5: Grafické zobrazenie útoku STP [21]

Na prevenciu proti STP útoku slúži mechanizmus zvaný *Root Guard*. Mechanizmus je povolený na všetkých smerovačoch, ktoré sa nemajú stať root bridge. Ide o odporúčané nastavenie pre smerovače, na ktoré sa pripájajú užívatelia, zariadenia a pod. [21]

CAM Overflow Attack

Smerovače pracujú pomocou tabuľky, kontextovo adresovanej pamäte, CAM (angl. *Context-Addressable Memory*). Tabuľka obsahuje MAC adresy spojených s portmi. Smerovač na základe MAC adresy vie, na ktorý port má poslať rámce. Pretečenie CAM je typ útoku, pri ktorom sa útočník pripojí na port smerovača a pomocou nástroja generuje tisíce náhodných MAC adries. Smerovač si dané adresy ukladá do tabuľky, a tým naplní svoju kapacitu. Smerovač nové adresy hostiteľov zahadzuje, čo spôsobí jeho zmenu na rozbočovač. Zmena umožní útočníkovi sieťový odposluch (časť 1.3.3) a vykonanie útoku *Man-in-the-Middle* (časť 1.3.3). [21]

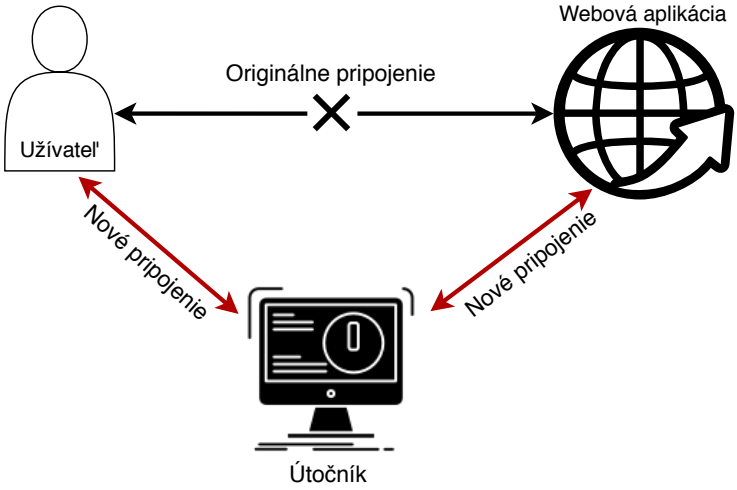
Prevencia proti danému útoku spočíva v bezpečnosti portov, tzv. *Port Security*. Port security na smerovači umožní jeho kontrolu nad ukladaním MAC adries. Ak sa na smerovač začne odosielať príliš veľa adries, port security smerovač vypne.

1.3.3 Útoky na L3 vrstve

V nadchádzajúcej časti kapitoly sú vysvetlené útoky spadajúce do sieťovej vrstvy referenčného modelu ISO/OSI (časť 1.3.1).

General MitM Attack

MitM (angl. *Man-in-the-Middle*) je druh aktívneho sieťového útoku, pri ktorom útočník postaví samého seba medzi obeť a cieľovú webovú aplikáciu (obr. 1.6). Podstatou útočníka je snaha dostať sa do pozície, ktorá mu umožňuje pozorovanie všetkej prevádzky tým, že sa stane aktívnym prostredníkom medzi nimi. Pozícia mu okrem odposluchu umožňuje prevádzku modifikovať. [22]



Obr. 1.6: Grafické zobrazenie MitM útoku [23]

Prevencia pred daným útokom je založená na všeobecnej ochrane, ktorou by sa mal riadiť každý užívateľ internetu [24]:

- Používať **silné prihlasovacie údaje** na svojom smerovači.
- Používať **komunikačný protokol HTTPS**.
- Používať najnovšie **WPA3**[3] (angl. *Wi-Fi Protected Access 3*) zabezpečenie svojej WiFi.

Sybil Attack

Útok Sybil je druh útoku, pri ktorom je systém reputácie[4] narušený vytváraním viacerých identít. Zraniteľnosť závisí od toho, ako jednoducho je možné generovanie identít a od stupňa, do akého systém reputácie prijíma vstupy od nedôveryhodných entít.

Entita v sieti *peer-to-peer*[5] je kus softvéru, ktorý má prístup k lokálnym zdrojom. Entita sa v sieti prezentuje pomocou identity a každá z nich môže mať identít

[3]V prípade, že ho smerovač nepodporuje, používať WPA2.

[4]Programy, ktoré umožňujú vzájomné hodnotenie používateľov s cieľom vybudovania si dôvery prostredníctvom reputácie.

[5]Počítačová sieť, v ktorej sú uzly navzájom prepojené.

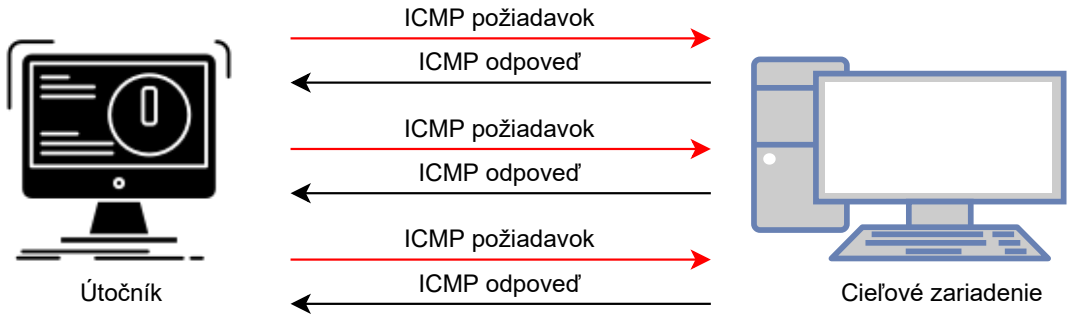
viac. Viacero identít sa používa na účely zdieľania zdrojov, integrity a spoľahlivosti. Útočník sa môže v sieti predstaviť viacerými identitami, aby sa mohol javiť ako viac samostatných uzlov. Môže tak získať väčšiu kontrolu nad sieťou. [25]

K základnej prevencii pred útokom Sybil patrí:

- **Overenie totožnosti**: Pomocou overenia telefónneho čísla, platobnej karty, IP adresy.
- **Potvrdenie osobnosti**: Overenie fyzickej prítomnosti osoby entity.
- **Obrana aplikácie**: Využitie algoritmov (`SumUp`, `DSybil`) odolných voči Sybil.

ICMP flooding

ICMP (angl. *Internet Control Message Protocol*) je protokol slúžiaci na odosielanie chybových správ. Pomocou nástroja *ping*[6] odosiela požiadavky a odpovede medzi odosielateľom a zariadením, ktorý zistí stav pripojenia medzi nimi. Princípom útoku je zasielanie ICMP požiadaviek na cieľové zariadenie, čo spôsobí jeho zahltenie (obr. 1.7). Sieť je nútená požiadavkám odpovedať, čo spôsobí neprístupnosť zariadenia v bežnej prevádzke. [26]



Obr. 1.7: Grafické zobrazenie ICMP útoku [26]

Prevenciou pred zahltením zariadenia ICMP požiadavkami je vypnutie ICMP funkcie na cieľovom smerovači, počítači a pod. Nastavením firewallu na blokovanie pingov je možné účinne zabrániť útokom spustených mimo lokálnej siete, ale taktiež nastavenie spôsobí, že zariadenie nebude schopné odpovedať na požiadavky *ping*, *traceroute*[7] a iných sieťových aktivít. [26]

Network Eavesdropping

Sieťový odposluch (angl. *Network Eavesdropping*) je útok, pri ktorom útočník získava informácie používateľa prostredníctvom sieťovej prevádzky pomocou dotazov

[6]Nástroj používaný na testovanie dosiahnuteľnosti zariadenia pomocou IP adresy.
[7]Príkaz slúžiaci na zobrazenie možných ciest a prípadných oneskorení odoslaných paketov.

na systém doménových mien DNS (angl. *Domain Name System*), požiadaviek a odpovedí na hypertextový prenosový protokol HTTP (angl. *Hypertext Transfer Protocol*) a pod. Zameriava sa na zachytávanie paketov zo siete prenášaných inými počítačmi a na čítanie dátového obsahu. [24]

Pasívny útok je druh sieťového odposluchu, pri ktorom útočník zozbiera údaje o svojom cieli, ale údaje nie sú zmenené. Pri **aktívnom útoku** sa útočník pripojí na sieť a vydáva sa ako legitímne spojenie, pričom môže meniť alebo blokovať pakety.

Detekcia pasívneho útoku je extrémne náročná, dokonca až nemožná, pretože nevykazuje žiadne zmeny v sieti. Aktívny útok je na detekciu ľahší, ale veľmi často sú už dáta zachytené útočníkom v čase, kedy je zmena v sieti viditeľná.

Prevencia proti sieťovému odposluchu pozostáva zo základných pravidiel zabezpečujúcich užívateľov internetu [27]:

- **Použitie bezpečných technológií** (Firewall, Antivírus).
- **Šifrovanie** emailov, sieťovej komunikácie, dát.
- **Overovanie** prichádzajúcich paketov.
- **Monitorovanie siete** kvôli výskytu abnormálnych aktivít (použitie IDS).

2 Charakteristika riešenia

Hlavnou úlohou práce je poskytnutie prostriedku na ochranu lokálnej počítačovej siete. Ochranný prostriedok pozostáva zo zapojeného detekčného systému Surikata na lokálnej sieti, ktorý pri detekcii útoku upozorní administrátora, že sa v sieti niečo deje. Administrátor siete si je schopný dané útoky zobraziť pomocou webovej aplikácie, aplikácia mu poskytuje najdôležitejšie informácie o daných incidentoch.

2.1 Štruktúra zapojenia pracoviska

Obsahom danej časti kapitoly je popis architektúry zapojenia výsledného pracoviska. Obsahuje technický popis komponentov vyskytujúcich sa v architektúre, ich fyzické rozdelenie a vzájomné prepojenie.

2.1.1 Technický popis komponentov

FileBeat

Filebeat[1] je program slúžiaci na presmerovanie a centralizovanie údajov. Monitoruje umiestnenie obsahujúce súbory, ktoré mu je vopred určené. Pre každý súbor spúšťa tzv. *harvester*, ktorý číta každý riadok všetkých súborov hľadajúci nové informácie. Po ich nájdení dané informácie odosiela ďalej do ElasticSearch (časť 2.1.1). [29]

Formát a štruktúra dát odosielaných z detekčného systému Surikata je nasledujúci:

- *Fast.log* – Dáta sú vo forme tzv. „logov," tzn. informácie sú typicky bez vnútornej štruktúry [30]. Logy sú zväčša ťažko spracovateľné, čo má za následok ich náročné čítanie. Na obrázku 2.1 je zobrazený príklad logov.

```
10/20/2020-16:36:30.264549  [**] [1:2:1]  Possible TCP SYN DoS  [**] [Classification: (null)]
[Priority: 3] {TCP} 77.231.169.152:2841 -> 192.168.0.117:0
10/21/2020-15:14:49.891661  [**] [1:100000011:0] Error Based SQL Injection Detected [**]
[Classification: (null)] [Priority: 3] {TCP} 192.168.40.11:6254 -> 192.168.40.183:80
```

Obr. 2.1: Obsah vytvoreného logu

- *Eve.json* – Dáta sú zapísané v štandardnom formáte JSON (angl. *JavaScript Object Notation*). Formát sa používa na reprezentáciu štruktúrovaných údajov na základe objektu (obr. 2.2), čo umožňuje jednoduché spracovanie a uloženie dát do databázy Elasticsearch.

[1]Viac informácií dostupných na: `https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html`

```
"timestamp": "2009-11-24T21:27:09.534255",
"event_type": "alert",
"src_ip": "192.168.2.7",
"src_port": 1041,
"dest_ip": "x.x.250.50",
"dest_port": 80,
"proto": "TCP",
"alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id" :2001999,
    "rev": 9,
    "signature": "Possible TCP SYN DoS",
    "category": "",
    "severity": 1
}
```


Obr. 2.2: Obsah logu v tvare JSON súboru

Elasticsearch

Elasticsearch[2] je vyhľadávací nástroj založený na knižnici Lucene[3]. Nástroj možno definovať ako NoSQL[4] databázu, ktorá ponúka rýchlosť a flexibilitu pri práci s údajmi v najrôznejších prípadoch. Elasticsearch je vyvinutý v jazyku Java, napriek tomu možno k databáze pristupovať pomocou oficiálnych klientov v rôznych programovacích jazykoch.

Elasticsearch ponúka nasledujúce funkcionality [5]:

- Distribuovaná architektúra umožňuje ukladanie na tisíce serveroch a ukladanie veľkého množstva dát.
- Využitie formátu JSON zabezpečuje serializovanie zložitých entít na indexované dokumenty.
- Extrémne rýchle vyhľadávanie (Dotaz, ktorým SQL databáza vyhľadá výsledok za 10 sekúnd, Elasticsearch vyhľadá za 10 milisekúnd).
- Správa životné cyklu indexu (angl. *Index Lifecycle Management*) – umožňuje automatické odstraňovanie zastaralých nepoužívaných indexov.
- Pokročilé vyhľadávanie – poskytuje funkcie ako napr. automatické dokončenie (angl. *Auto-completion*) a funkciu "mali ste na mysli"(angl. *“Did-you-mean”*).

[2]Viac informácií je dostupných na: `https://www.elastic.co/what-is/elasticsearch`

[3]Vyhľadávajúca knižnica bez funkcionalít vhodná ako základ iných nástrojov.

[4]Na ukladanie dát využíva dokumenty namiesto schém alebo tabuliek.

[5]Viac informácií o daných funkcionalitách je dostupných na: `https://qbox.io/blog/what-is-elasticsearch`.

Raspberry Pi

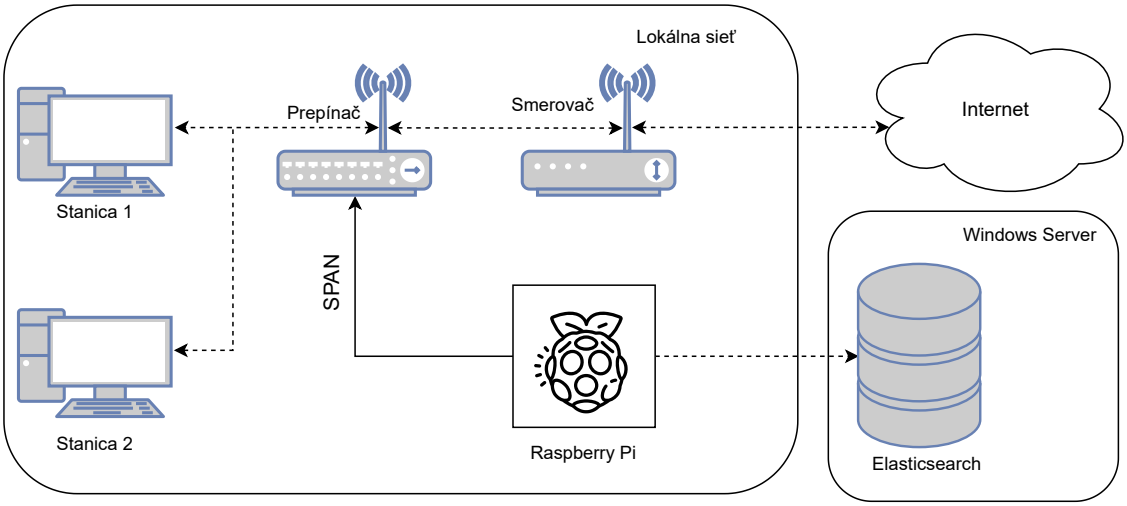
Raspberry Pi je jednodoskový počítač s veľkosťou platobnej karty pripájajúci sa k počítaču a používajúci štandardnú myš a klávesnicu. Je to malé zariadenie, ktoré vyvíja britská nadácia Raspberry Pi Foundation s cieľom podpory výučby informatiky. Zariadenie využíva bezplatný operačný systém Raspberry Pi OS založený na systéme Debian optimalizovanom pre hardvér Raspberry Pi. [28]

Zariadenie je v bakalárskej práci využité na inštaláciu detekčného systému Surikata (časť 1.2.4) a vyššie uvedeného komponentu Filebeat.

2.1.2 Architektúra zapojenia pracoviska

Na obrázku 2.3 je zobrazené zapojenie pracoviska výsledného riešenia práce. Lokálna sieť pozostáva z množiny koncových staníc, pričom každý prvok reprezentuje klienta napojeného na prepínač. Každý klient je pripojený na internet pomocou smerovača (angl. *Router*) slúžiaceho na sprostredkovanie prenosu dát medzi klientmi a inými počítačovými sieťami.

Na prepínač (angl. *Switch*) je ďalej pomocou SPAN (angl. *Switched Port Analyzer*) pripojené zariadenie Raspberry Pi (časť 2.1.1). *Port Mirroring* (SPAN) slúži na zasielanie kópie paketov prechádzajúcich prepínačom, poskytuje prístup k paketom na ich monitorovanie. Na zariadení Raspberry Pi je nainštalovaný softvér Surikata (časť 1.2.4), ktorý slúži na detekciu útokov na celej lokálnej sieti. Akonáhle Surikata deteguje útok na sieti, odošle na Raspberry Pi záznam so všetkými dostupnými informáciami (viď obr. 2.2). Na zariadení je ďalej nainštalovaný komponent Filebeat, ktorý dané logy zbiera a odosiela ich do centrálneho úložiska – databázy Elasticsearch.



Obr. 2.3: Architektúra zapojenia lokálneho pracoviska

2.2 Štruktúra zapojenia webovej aplikácie

V tejto časti kapitoly je popísaná architektúra zapojenia výsledného pracoviska. Obsahuje technický popis komponentov vyskytujúcich sa v architektúre, ich fyzické rozdelenie a vzájomné prepojenie. V nadchádzajúcich častiach sú popísané architektúry pracoviska a webovej aplikácie, ktoré poskytujú fyzické rozdelenie komponentov a ich vzájomné prepojenie. Jednotlivé komponenty sú podrobne vysvetlené.

2.2.1 Technický popis komponentov

Frontend

Termín *Frontend* sa používa v súvislosti s webovými aplikáciami. Označuje časť webovej stránky, ktorú je užívateľ schopný vidieť na svojej obrazovke počítača a dokáže s ňou komunikovať. [32]

Webový server

Webový server môže byť počítač alebo počítačový program, ktorý prijíma HTTPS požiadavky od užívateľa prostredníctvom prehliadača (napr. Google Chrome). Na základe tejto požiadavky zasiela odpoveď obsahujúcu dáta v závislosti na požiadavke. K najznámejším programom zabezpečujúcim webový server patrí *Apache HTTP Server*, *Internet Information Services*, *Sun Java System Web Server*.

Obsahuje všetku logiku aplikácie, tzn. na serveri je uložená tzv. *backendová* časť webovej aplikácie. Termín *backend* označuje časť aplikácie, ktorá slúži na spracovávanie dát a je bežnému užívateľovi skrytá [32].

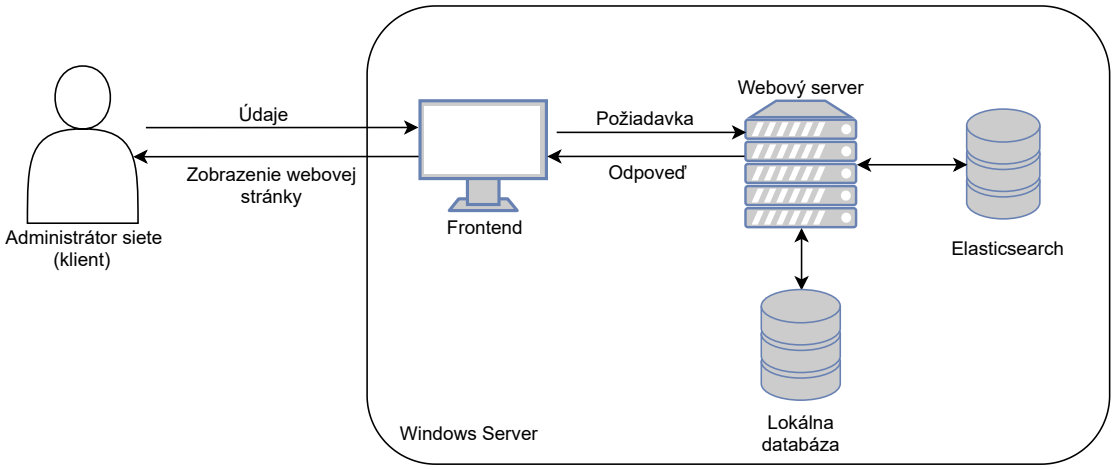
2.2.2 Architektúra zapojenia aplikácie

Na obrázku 2.4 je zobrazená štruktúra zapojenia webovej aplikácie. Akonáhle klient stlačí tlačidlo na odoslanie URL[6] (angl. *Uniform Resource Locator*) adresy webového prehliadača, vysiela HTTPS požiadavku na webový server s cieľom zobrazenia webovej stránky. Webová aplikácia odpovedá formou zobrazenia prihlasovacieho okna z dôvodu prenosu citlivých dát. Administrátor je po zadaní prihlasovacích údajov serverom overený a presmerovaný do primárnej sekcie stránky. Administrátor je overovaný každou odoslanou HTTPS požiadavkou a prípade, že overovanie zlyhá, nie je možné sa k daným dátam dostať, napr. po vypršaní doby platnosti overenia.

Aplikácia v prípade pokynu od užívateľa ďalej odosiela požiadavku na centrálne úložisko Elasticsearch, aby zistila prípadné nové počítačové útoky vyskytnuté na lokálnej sieti. Po prijatí nových dát ich aplikácia ukladá do svojej lokálnej databázy.

[6]Termín označujúci webovú adresu.

Webový server v danom momente komunikuje s prihláseným užívateľom prostredníctvom HTTPS požiadaviek už iba s dátami uloženými v lokálnej databáze.



Obr. 2.4: Architektúra zapojenia webovej aplikácie

3 Návrh webovej aplikácie

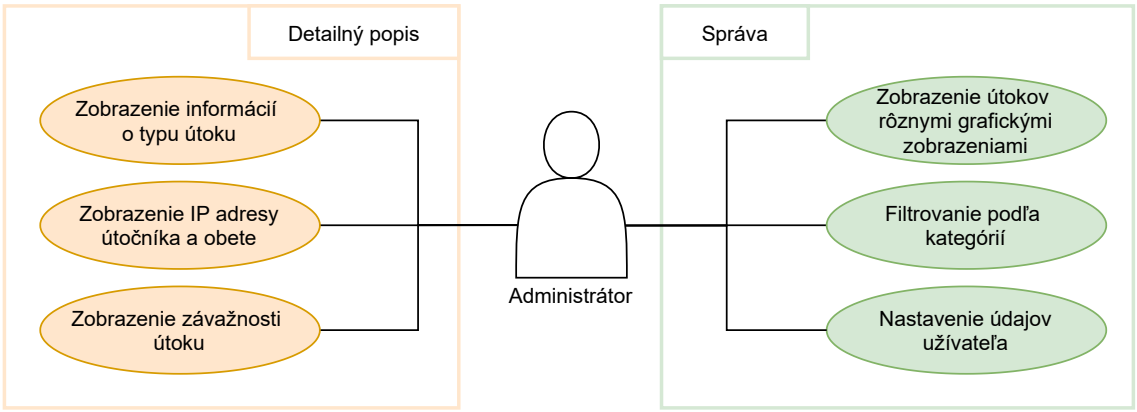
3.1 Analýza webovej aplikácie

Na začiatku kapitoly je potrebné vykonanie analýzy použitia webovej aplikácie. Daný proces tvorí základ úspešného návrhu riešenia, pretože umožňuje definovať činnosti nevyhnutné k jeho realizácii. [33]

V rámci sekcie je vytvorený model prípadu užitia. Na základe vytvoreného modelu sú špecifikované požiadavky webovej aplikácie.

3.1.1 Model prípadu užitia

Výhodou použitia tohto modelu je jednoduchá grafická reprezentácia funkcionality programu, ktorá zjednodušuje proces návrhu aplikácie [33]. Model obsahuje hlavné požiadavky, ktoré aplikácia musí splniť (obr. 3.1). Použitie modelu zaručí pochopenie toho, čo má výsledný systém robiť a prispeje k zjednodušeniu vytvorenia návrhu aplikácie.



Obr. 3.1: Model prípadu užitia

3.1.2 Špecifikácia požiadaviek

Ďalším krokom vytvorenia návrhu webovej aplikácie je definovanie požiadaviek na aplikáciu. Na ich definovanie je nevyhnuté vytvorenie modelu prípadu užitia, pretože je dôležité porozumieť potrebám administrátora siete a ponúknuť mu riešenie, ktoré splní jeho očakávanie. Požiadavky musia byť presne stanovené, aby nedochádzalo k prípadným nejasnostiam. [33]

Funkčné požiadavky

Funkčné požiadavky predstavujú základnú množinu úkonov, ktoré systém musí byť schopný vykonať. Zoznam funkčných požiadaviek na webovú aplikáciu je nasledovný:

- Webová aplikácia je rýchla z hľadiska odozvy vďaka použitiu architektúry single-page[1].
- Aplikácia prehľadne zobrazuje všetky útoky vzniknuté na lokálnej sieti pomocou niekoľkých grafických zobrazení.
- Poskytuje prehľadné zobrazenie útokov pomocou grafov.
- Obsahuje prihlasovací systém.
- V aplikácii je možné filtrovanie dát.
- Aplikácia obsahuje zabezpečenie.
- Prístup k údajom je možný iba autorizovaným užívateľom.
- Každý užívateľ je schopný zmeniť si osobné nastavenia.

Nefunkčné požiadavky

K druhej skupine požiadavkov patria úkony, ktoré daná aplikácia neobsahuje z dôvodu, že ich realizácia môže byť v rozpore s funkcionalitou aplikácie alebo jej obmedzovaním. Zoznam nefunkčných požiadaviek je nasledovný:

- Výsledná aplikácia využíva multi-page[2] architektúru.
- Aplikácia je funkčná na viacerých platformách.
- Výsledná aplikácia je spoľahlivá.
- Práca so serverovou časťou aplikácie je výkonná.

Kritické požiadavky

Pri tvorbe aplikácie je dôležité zameranie sa na nevyhnutné (kritické) funkcionality. V praxi bolo dokázané, že softvér zameraný len na kritické požiadavky je vo výsledku úspešnejší ako softvér realizujúci všetko [33]. Výhoda daného prístupu spočíva v tom, že jeho užívateľ sa dokáže rýchlejšie zorientovať a nájsť tú funkciu, ktorú potrebuje.

Kritická požiadavka v prípade aplikácie zobrazujúcej počítačové útoky je ich grafická reprezentácia. Na realizáciu tejto funkcie je kladený najvyšší dôraz, aplikácia prehľadne zobrazuje informácie o incidentoch a administrátor je schopný ich filtrovaniu podľa rôznych kategórií.

[1] Namiesto načítavania každej stránky dynamicky prepisuje jednu.

[2] Každá stránka sa odosielaním formuláru znovu načítava.

3.2 Porovnanie webových frameworkov

Webový framework je softvérový nástroj navrhnutý na tvorbu webových aplikácií, ako sú webové servery, portály, rozhrania atď. Najlepšie frameworky ponúkajú rôzne druhy šablónových knižníc, statické servery, prístupy do databázy, čo oproti minulosti zjednodušilo webovým vývojárom prácu a umožnilo im zmeniť štruktúry aplikácie aj po napísaní kódu. V procese tvorby webovej aplikácie je veľmi dôležité vykonanie analýzy dostupných webových frameworkov. Správna voľba frameworku je kľúčovým krokom k úspešnej tvorbe aplikácie.

V súvislosti s rôznymi požiadavkami na webovú aplikáciu sa webové frameworky všeobecne rozdeľujú na *Client-Side (Frontend)* frameworky a *Server-Side (Backend)* frameworky. [34]

3.2.1 Client-Side (Frontend) frameworky

Frontend vývojár je zodpovedný za tú časť webovej aplikácie, ktorú užívateľ vidí. Sústreďuje sa na vytváranie pozitívneho užívateľského zážitku a zabezpečuje dizajn a prehľadnosť stránky. K hlavným nástrojom frontendového vývojára patrí HTML, CSS a JavaScript. [32]

HTML (angl. *Hypertext Markup Language*) je značkovací jazyk určený na vytváranie webových stránok. Kladie dôraz na prezentáciu informácií (font písma, odseky, rozdelenie stránky...).

CSS (angl. *Crew Safety System*), kaskádové štýly, je všeobecné rozšírenie HTML. Umožňuje vizuálne formátovanie stránky, oddeľuje štruktúru HTML od vzhľadu.

JavaScript je skriptovací programovací jazyk, ktorý sa používa na tvorbu interakcie webovej stránky.

React

React je voľne dostupný webový framework považovaný za jeden z najjednoduchších na učenie. Používa sa pri vytváraní užívateľského rozhrania, najmä pri *single-page* aplikácii. Vyniká vďaka svojmu modelu DOM (angl. *Document Object Model*), ktorý ponúka výnimočnú funkčnosť. React je ideálny na zvládnutie veľkej prevádzky stabilným spôsobom. [35]

- **Výhody:**
  1. Znovupoužitie komponentov.
  2. Vytváranie komponentov bez použitia tried.
  3. Obsahuje pokročilé a užitočné nástroje.
- **Nevýhody:**
  1. Časté aktualizácie.

2. Komplexnosť frameworku.
3. Možnosť iba frontendového riešenia.

Webové stránky založené na frameworku React: Facebook, Instagram, Discord [36].

Angular

Angular je považovaný za jeden z najlepších frameworkov používajúci sa pri vývoji single-page aplikácií. Staršia verzia Angularu, AngularJS, využívala jazyk JavaScript. Novšia verzia, Angular, je založená na jazyku TypeScript. Rozdiel medzi nimi je taký, že TypeScript je považovaný za objektovo orientovaný jazyk a JavaScript za jazyk skriptovací. Pri tvorbe mobilných a webových aplikácií je Angular skvelá voľba.

Je jedinečný vďaka funkcii obojsmerného viazania. To znamená, že synchronizácia medzi modelom a pohľadom prebieha v reálnom čase, každá zmena modelu sa okamžite odrazí do pohľadu. Vďaka danej funkcii sa znižuje aj množstvo kódu, čo spôsobí jeho lepšiu čitateľnosť. [37]

- **Výhody:**
  1. Zmena modelu spôsobí okamžitú zmenu pohľadu.
  2. Znovupoužitie komponentov.
  3. Využitie novšieho jazyka TypeScript.
- **Nevýhody:**
  1. Možnosť vyskytnutia problémov načítavania pri veľkom množstve dát.
  2. Náročnosť učenia.

Webové stránky založené na frameworku Angular: Gmail, PayPal, Overleaf [38].

Vue.js

Vue.js je pokročilý framework, ktorý umožňuje prispôsobiť Vue jednu komplexnú časť už existujúceho projektu, a tak ju zjednodušiť. Je odporúčaný na tvorbu flexibilnej štruktúry dizajnu, umožňuje dizajnovanie všetkého od nuly. [35]

- **Výhody:**
  1. Dostupná podrobná dokumentácia.
  2. Jednoduchá syntax, základná znalosť JavaScriptu je pre začiatočníkov dostatočná.
  3. Flexibilita dizajnu.
  4. Podpora TypeScriptu.

- **Nevýhody:**
  1. Neobsahuje niektoré základné komponenty.
  2. Malá komunita podpory.

3. Väčšina prídavných modulov je napísaná v čínštine.

Webové stránky založené na frameworku Vue.js: Gitlab, Wizzair [39].

Bootstrap

Bootstrap je voľne dostupný webový framework, ktorý umožňuje tvorbu responzívneho[3] webu vďaka rozdeleniu obrazovky na 12 stĺpcov. Je založený na jazykoch CSS a JavaScript. [40]

- **Výhody:**
  1. Responzívny dizajn webovej stránky.
  2. Obsahuje mnoho komponentov.
  3. Komponenty dokáže jednoducho dizajnovať bez poznania CSS.
- **Nevýhody:**
  1. Nepoužíva sa často samostatne, ale spolu s iným frontendovým frameworkom.
  2. Pri používaní Bootstrapu vyzerajú všetky stránky veľmi podobne.

3.2.2 Server-Side (Backend) frameworky

Backend vývojár na druhej strane pracuje na tej časti aplikácie, ktorú užívateľ nevidí. Využíva široký okruh knižníc, rozhraní a pod. Je zodpovedný za implementáciu databázového systému, komunikáciu medzi webovými službami, vytváranie backendových funkcií a ďalšie. V tabuľke 3.1 sú vyznačené najznámejšie backendové frameworky, na akom programovacom jazyku sú založené a príklady užitia webových stránok vytvorených daným frameworkom. [41]

Tab. 3.1: Porovnanie webových frameworkov [41]

Framework	**Programovací jazyk**	**Prípady použitia**
Django	Python	Instagram
Spring Boot	Java	Trivago
ASP.NET	C#	Microsoft
Ruby on Rails	Ruby	GitHub

Django

Django je jeden z popredných, voľne dostupných frameworkov založený na programovacom jazyku Python. Riadi sa vzorom MVC[4] (angl. *Model-View-Controller*).

[3] Stránka sa zobrazuje v optimálnej veľkosti vo všetkých druhoch zariadení.

[4] Dizajnový vzor, model-pohľad-radič, ktorý rozdeľuje programovú logiku na tri navzájom prepojené elementy.

Je vhodný na vývoj funkcie bohatých databázových webových stránok. Poskytuje optimálne rozhranie pomáhajúce pri tvorbe operácií. [34]

- **Výhody:**
  1. Jednoduchý na používanie.
  2. Poskytuje ochranu pred niekoľkými bezpečnostnými problémami.
  3. Ponúka flexibilitu a škálovateľnosť.
  4. Poskytuje vhodnú dokumentáciu.
- **Nevýhody:**
  1. Pri práci je nutná znalosť celého systému.
  2. Nepatrí k najrýchlejším.

ASP.NET Core

ASP.NET Core je voľne dostupný framework vyvinutý spoločnosťou Microsoft. Je vhodný na tvorbu robustných webových aplikácií pre mobilné a desktopové zariadenia. Je založený na programovacom jazyku C#. Poskytuje vývojárom voľbu využitia asynchrónneho programovania. Vysoký výkon je jednou z kľúčových vlastností ASP.NET Core. [41]

- **Výhody:**
  1. Podpora viacerých platforiem (Windows, Mac, Linux).
  2. Obsahuje technológie, vďaka ktorým je potrebné menšie množstvo kódu.
  3. Menšie množstvo kódu znamená jeho ľahšiu údržbu.
- **Nevýhody:**
  1. Niektoré rozhrania, napr. rozhranie priraďujúce užívateľom rolu, sú zbytočne komplikované.
  2. Slabšie bezpečnostné prvky.

Spring

Spring je najobľúbenejší aplikačný, voľne dostupný framework programovacieho jazyku Java. Najdôležitejším účelom Springu je možnosť tvorby J2EE[5] (angl. *Java 2 Enterprise Edition*) aplikácií. Systémy a aplikácie založené na jazyku Java sa definujú ako rýchle, jednoduché a flexibilné. [42]

- **Výhody:**
  1. Dobrá voľba pre Java aplikácie.
  2. Flexibilný framework.
  3. Vlastnosti aplikácie sú ľahko prispôsobiteľné.
- **Nevýhody:**

[5] Aplikácia vytvorená z komponentov ako JavaServer, Java servlet...

1. Príliš komplexný.
2. Dokáže byť nevybalansovaný.

Ruby on Rails

Ruby on Rails, známy ako Rails, je framework založený na programovacom jazyku Ruby s licenciou MIT[6]. Je známy ako framework pre začiatočníkov pomáhajúci programátorom k rýchlemu programátorskému rastu. Rails je MVC framework poskytujúci predvolené databázové štruktúry, webové stránky a služby. [41]

Rails povoľuje vývojárom vykonávať interakciu s databázou bez námahy. *Query*[7] sú písané v programovacom jazyku Ruby a tie sú konvertované do jazyka SQL, databáza vracia objekty.

- **Výhody:**
  1. Obsahuje rôzne druhy knižníc a nástrojov.
  2. Jednoduchý proces automatických testov.
  3. Rýchly vývoj aplikácií.
- **Nevýhody:**
  1. Nie je preferovaný pri veľkých aplikáciach.
  2. Pomalé spôsoby spustenia.

3.3 Spôsoby zobrazenia získaných dát

Kľúčový bezpečnostný prístup na predchádzanie útokom je identifikácia a reakcia na bezpečnostné udalosti v reálnom čase tak, aby minimalizovali dopad na infraštruktúru a vzniknuté škody. Softvér na zabezpečenie informácií a správu udalostí **SIEM** (angl. *Security Information and Event Management Software*) umožňuje udržiavať prehľad o bezpečnostných varovaniach v reálnom čase vďaka analýze aktivity v celej IT infraštruktúre. [44]

SIEM poskytuje nasledujúce funkcionality [44]:

- Bezpečnostné monitorovanie.
- Detekcia a skúmanie možných hrozieb.
- Zasielanie oznámení a výstrah.
- Reakcia na hrozby.

Spoločnosti využívajú SIEM na ochranu svojich najcitlivejších údajov, každý server prijíma údaje z rôznych zdrojov a generuje správu so všetkými zaznamenanými udalosťami. Na obrázku 3.2 je zobrazený celý proces riešenia SIEM.

[6]Voľne dostupná softvérová licencia s veľmi malými obmedzeniami. Kód s MIT licenciou sa dá využiť komerčne.

[7]Požiadavka na informácie z databázy.

Obr. 3.2: Proces riešenia SIEM [44]

3.3.1 SIEM nástroje

K najznámejším SIEM nástrojom patria nástroje Splunk, IBM QRadar, OSSEC, RSA NetWitness. Nasledujúce časti kapitoly sa zaoberajú popisom uvedených SIEM nástrojov.

Splunk

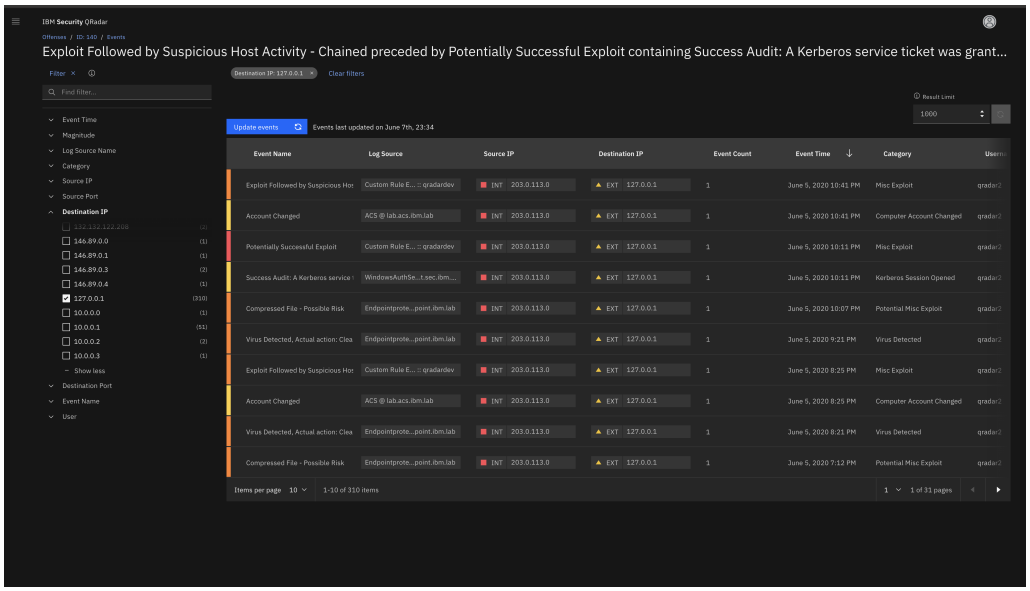
Splunk je považovaný za jeden z najpopulárnejších SIEM nástrojov, navrhnutý pre operačné systémy Windows a Linux. Dáta sú monitorované v reálnom čase, pretože systém vyhľadáva potenciálne chyby zabezpečenia a dokonca dokáže poukazovať na abnormálne správanie. Obsahuje jednoduché užívateľské rozhranie, užívateľ pri vykonávaní kontroly môže začať so základným prehľadom a dopracovať sa k podrobnému prehľadu danej udalosti. [45]

Splunk poskytuje vylepšené bezpečnostné operácie, ako je štatistická analýza a kontrola, klasifikácia nehôd a prispôsobiteľný dashboard[8]. Poskytuje bezpečnostné služby pre verejný sektor, finančné služby a zdravotnú starostlivosť. [46]

IBM QRadar

IBM QRadar je ďalší populárny SIEM nástroj umožňujúci nasadenie ako hardvérového, tak aj virtuálneho alebo softvérového zariadenia v závislosti od potrieb organizácie [44]. Je využiteľný na operačnom systéme Red Hat Enterprise Linux. Systém disponuje analýzou modelovania rizík, ktorý dokáže simulovať potenciálne útoky. IBM QRadar je jeden z najkompletnejších riešení SIEM, platforma ponúka sadu funkcií, analytiky, zhromažďovania údajov a detekcie prienikov. [45]

[8]Produkt, ktorý má za cieľ integráciu ifnromácií z viacerých zložiek do jednotného zobrazenia.



Obr. 3.3: Prehľad sady analytických funkcií[9]

OSSEC

OSSEC sa zaraďuje k najlepším uzlovo orientovaným systémom detekcie prienikov HIDS. Je voľne dostupný pre operačné systémy Windows, Linux, Unix a Mac. Softvér sa zameriava na informácie dostupné v log súboroch, v ktorých hľadá dôkazy o vniknutiach. Sleduje aj kontrolné súčty súborov kvôli neoprávnenej manipulácii. Frontendová časť riešenia nie je z dôvodu jeho nekomerčnosti najlepšia, a preto jeho užívatelia zväčša využívajú iné prostriedky na analýzu správy protokolov (viď obrázok 3.3). [45]

RSA NetWitness

Platforma využíva rôzne zdroje údajov ako napr. RSA NetWitness Network, RSA NetWitness logs a RSA NetWitness Endpoint. Analytikom poskytuje spojenie s incidentmi v reálnom čase, čím pomáha eliminovať hrozby pred ich dopadom na infraštruktúru (obr. 3.4). Pomocou špecializovaných algoritmov automaticky extrahuje metadáta relevantné k hrozbe a poskytuje kompletnú správu vykonaných incidentov (obr. 3.4). RSA NetWitness je možné nasadiť ako samostatné či viacnásobné zariadenie, čiastočne alebo úplne virtualizované zariadenie uložené na cloude[10]. [46]

[9]Prehľad sady funkcií a webová aplikácia sú dostupné z: `https://www.ibm.com/products/qradar-siem`.

[10]Vzdialené zariadenie poskytujúce služby.

Obr. 3.4: Prehľad správy vykonaných incidentov[11]

3.4 Návrh riešenia

Voľba správneho frameworku závisí od viacerých faktorov, ktoré musia byť pri výbere zohľadnené. K hlavným kritériám výberu patria odpovede k nasledujúcim otázkam:

1. Je voľne dostupný?
2. Vyvíja sa v súčasnosti?
3. Obsahuje všetky knižnice a nástroje potrebné pre zadanie?
4. Je možné dostatočne zaistiť bezpečnost aplikácie?
5. Je dostupná dostatočná dokumentácia?
6. Aká je časová náročnosť porozumenia frameworku?

Po vykonaní hĺbkovej analýzy všetkých možností a zodpovedaní si daných otázok je vyvodený záver, že na tvorbu webovej aplikácie, ktorá zobrazuje počítačové útoky, je najvhodnejšia nasledujúca voľba:

- **ASP.NET** na tvorbu backendovej časti aplikácie.
- **Angular** na tvorbu frontendovej časti aplikácie.

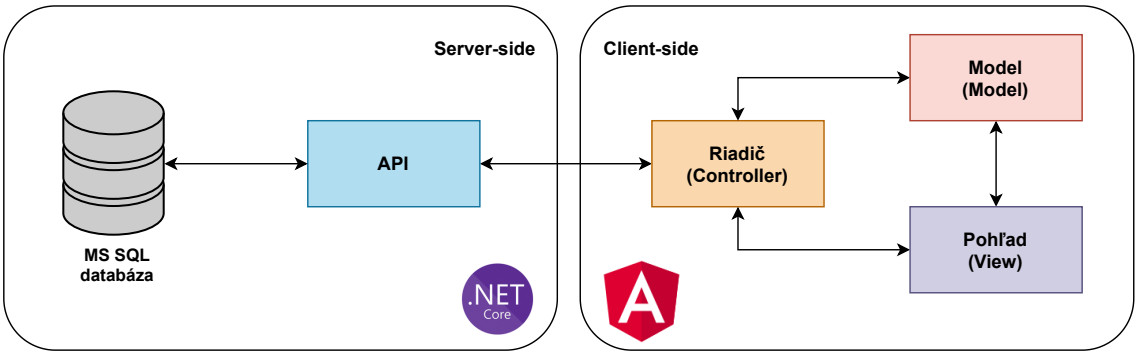
ASP.NET je zvolený z dôvodu využitia programovacieho jazyka C# a obsahovania veľkého množstva knižníc a funkcií dôležitých pre webovú aplikáciu. Je založený na obľúbenom vzore MVC, ktorého účelom je oddelenie logiky od výstupu. Veľkou výhodou ASP.NET je *Entity framework*, ktorý slúži na zjednodušenie práce s databázou, využíva princípy objektovo orientovaného programovania, pretože k databáze pristupuje pomocou objektov. ASP.NET sa zväčša využíva iba na tvorbu tzv. API. (angl. *Application Programming Interface*), pretože programovanie fron-

[11] Prehľad správy incidentov a webová aplikácia sú dostupné z: `https://www.rsa.com/`.

tendovej časti aplikácie v ňom nie je ideálne. Z daného dôvodu je nutné pre frontendovú časť aplikácie zvoliť populárnejší *Client-Side* framework.

Angular je považovaný za vhodnú voľbu ku frameworku ASP.NET z dôvodu jeho zámeru na tvorbu *single-page* aplikácií. Obsahuje techniky ako napr. *Dependency Injection*[12], vďaka ktorým je načítavanie stránky rýchlejšie. Angular je síce náročnejší na pochopenie kvôli jeho rozsiahlosti, čo môže odradiť nových vývojárov, avšak investícia času sa pre podrobné pochopenie klientského rozhrania vyplatí. Angular taktiež obsahuje veľké množstvo knižníc a grafických šablón, ktoré vývojárovi zjednodušujú vývoj klientskej časti aplikácie.

Na obrázku 3.5 je znázornené prepojenie webových frameworkov ASP.NET a Angularu a ich vzájomná komunikácia pomocou HTTPS requestov.



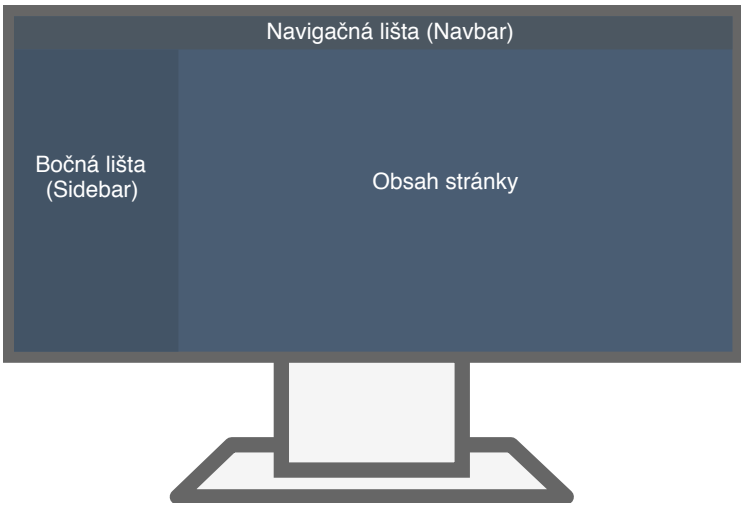
Obr. 3.5: Architektúra ASP.NET s využitím frameworku Angular [43]

3.4.1 Návrh webovej stránky

Pri tvorbe návrhu webovej stránky je vhodné premyslieť jej základnú štruktúru. Pri zvažovaní typu rozloženia je potrebné vziať do úvahy účel webovej stránky, podľa účelu by mala obsahovať špecifické vlastnosti a prvky. Stránka by mala byť vždy vhodne rozdelená do niekoľkých častí, aby sa v nej užívateľ vedel rýchlo zorientovať. Základné rozloženie webovej stránky by malo zahŕňať nasledovné časti (obr. 3.6):

- Bočnú časť, ktorá umožňuje presmerovanie na jednotlivé sekcie webovej stránky.
- Navigačnú časť, ktorej obsahom sú údaje o prihlásenom užívateľovi, možnosť odhlásenia sa a pod.
- Obsahovú časť, ktorá zobrazuje náplň jednotlivých sekcií dynamicky sa meniacich po presmerovaní užívateľa na inú sekciu.

[12]Technika, pri ktorej objekt prijíma iný objekt, na ktorom je závislý.



Obr. 3.6: Štrukturálny návrh webovej stránky

Okrem tvorby základnej štruktúry stránky je dôležité premyslieť, akým spôsobom je najlepšie zozbierané dáta zobraziť tak, aby boli pre administrátora siete prehľadné a najmä užitočné. Predmetom vykonanej štúdie dostupných najznámejších SIEM riešení (kapitola 3.3.1) a ich následného porovnania je vytvorenie si predstavy, akým grafickým spôsobom je možné daný cieľ splniť. Výsledkom štúdie je vytvorenie 2 sekcií stránky umiestnených v bočnej časti stránky. Obsah prvej sekcie, založenej na správe incidentov podľa SIEM riešenia RSA NetWitness (obr. 3.4), je možné zaradiť pod pojmom analýza z dôvodu podrobného rozboru vyskytnutých dát. Úlohou druhej časti vychádzajúcej zo sady analytických funkcií SIEM riešenia IBM QRadar (obr. 3.3) je umožnenie administrátorovi vykonať investigáciu všetkých incidentov.

4 Realizácia tvorby riešenia práce

4.1 Postup implementácie riešenia

Postup implementácie je rozdelený do dvoch fáz. V rámci semestrálnej práce, vo fáze prvej, je dôležité:

1. Zostrojenie pracoviska na lokálnom zariadení – inštalácia Filebeat a Elasticsearch.
2. Vytvorenie backendu aplikácie.
3. Prepojenie lokálnej Elasticsearch databázy k backendu aplikácie.
4. Vytvorenie jednoduchého grafického zobrazenia počítačových útokov.

V rámci bakalárskej práce, vo fáze druhej, je dôležité:

1. Vytvorenie *dashboardu*[1] – na základe dostupných riešení zobrazenie prijatých dát.
2. Zhotovenie sekcie *Investigate* – filtrovanie dát.
3. Pridanie prihlasovacieho systému.
4. Implementovanie zabezpečenia API najnovšími bezpečnostnými technikami.
5. Pridanie sekcie na zmenu nastavení a tvorbu nového užívateľa.
6. Konfigurácia Raspberry Pi (povolenie vzdialeného pripojenia, *iptables*).
7. Inštalácia a nastavenie Surikaty na zariadení Raspberry Pi – zmena výstupného formátu na JSON.
8. Vytvorenie triedy na deserializáciu dát JSON.
9. Konfigurácia Windows Serveru a inštalácia databázy Elasticsearch.
10. Inštalácia a nastavenie Filebeat na zariadení Raspberry Pi – nastavenie monitorovaného súboru a odosielanie dát do databázy Elasticsearch na Serveri Windows.
11. Prepojenie zariadenia Raspberry Pi a Windows Serveru otvorením portu.
12. Testovanie konektivity pracoviska.
13. Vykonanie testovania experimentálneho pracoviska – realizácia troch počítačových útokov.

4.2 Prvá fáza

V prvej fáze realizácie je potrebné lokálne nainštalovať komponenty Filebeat a Elasticsearch, a nakonfigurovať ich na vzájomnú spoluprácu. Po overení funkčnosti konektivity odoslaním súboru obsahujúceho príklad útoku je možné venovať sa tvorbe základu backendovej časti aplikácie.

[1] Časť aplikácie, ktorá zjednocuje informácie z viacerých zložiek do jednotného zobrazenia.

4.2.1 Implementácia webovej aplikácie

Prvým krokom implementácie je tvorba lokálnej databázy MS SQL založenej na triede **Attack** obsahujúcej vlastnosti nevyhnutné na popis jednotlivých útokov. Nasleduje pridanie klienta **NEST**, pomocou ktorého sa aplikácia dokáže pripojiť na databázu Elasticsearch. Na prácu s HTTP požiadavkami je dôležité vytvoriť riadiacu jednotku (angl. *Controller*), ktorá slúži na centrálne riadenie všetkých požiadaviek. Rozhoduje, aký model a aké údaje sa vyberú a odošlú späť.

Základnou *get* metódou *Controlleru* je funkcia na pripojenie sa na databázu Elasticsearch, vďaka čomu je aplikácia schopná vytiahnuť všetky dáta, ktoré databáza obsahuje:

```
var scanResults = SearchAPI.Client.Search<Attack>(s => s
                            .From(0)
                            .Size(2000)
                            .Index("attacks")
                            .Query(q => q.MatchAll()));
```

Po vytiahnutí všetkých dát je potrebné uložiť ich do lokálnej databázy MS SQL. Súčasťou riadiacich jednotiek sú tzv. *Services*, ktoré slúžia na ukladanie logiky všetkých funkcií a oddelenie ich od riadiacej jednotky. Najdôležitejšia funkcia tejto *Service* pripojenej k hlavnej riadiacej jednotke je uloženie dát získaných pomocou vyššie uvedeného kódu do lokálnej databázy MS SQL.

Okrem vytvorenia serverovej časti je potrebné dokázať funkčnosť konektivity lokálneho pracoviska zobrazením dát pomocou grafického rozhrania. Na klientskej časti aplikácie je z daného dôvodu vytvorené jednoduché grafické zobrazenie (tabuľka).

Posledným krokom je potrebné prepojiť klientskú a serverovú časť. Na dané prepojenie slúži vo frameworku Angular modul *HttpClient* obsahujúci metódy na odosielanie HTTPS požiadaviek. Funkcia na pripojenie sa na serverovú časť a vytiahnutie dát z databázy je nasledovná:

```
  getAttacks(): void {
  this.tableData.getAttacks().subscribe(attacks => {
    this.attacks = attacks;
  })
};
```

Daná funkcia pomocou *subscribe()* metódy mapuje dáta zo servera do objektu rozhrania vytvoreného na klientskej časti. Týmto spôsobom sú úspešne prepojené obe časti aplikácie.

4.2.2 Zhrnutie dosiahnutých výsledkov

Praktická časť prvej fázy je vo väčšom množstve zameraná na serverovú časť, čiže na prácu s dátami. Grafické rozhranie v tomto momente obsahuje iba jednoduchú tabuľku, ktorá má za úlohu ukázať funkčnosť zapojeného pracoviska, čo znamená nasledovné:

1. Po vložení dát na vopred definované úložisko zariadenia je komponent Filebeat schopný dané informácie zozbierať a odoslať do lokálnej databázy Elasticsearch.
2. Po spustení webovej stránky je aplikácia schopná odoslať požiadavku na databázu Elasticsearch s cieľom získania nových dát.
3. Ak sa v databáze nové dáta nachádzajú, aplikácia ich uloží do svojej lokálnej databázy MS SQL a ďalej s nimi pracuje.
4. Webová stránka je schopná všetky informácie uložené v lokálnej databáze zobraziť.

4.3 Druhá fáza

Druhá fáza realizácie projektu, bakalárska práca, sa ďalej venovala praktickej časti práce. Postup implementácie bol rozdelený do nasledujúcich etáp:

1. Implementácia grafického zobrazenia aplikácie.
2. Zaistenie dostatočného zabezpečenia aplikácie.
3. Vytvorenie experimentálneho pracoviska.

V nasledujúcich častiach kapitoly sú jednotlivé etapy podrobne vysvetlené.

4.3.1 Grafické zobrazenie aplikácie

Dôležitou súčasťou webovej aplikácie je vhodné zobrazenie dát. Ako bolo spomenuté v podkapitole 3.4.1, webová stránka je rozdelená na dve časti zobrazujúce dáta (*Dashboard* a *Investigate*) a časť určenú na nastavenia aplikácie (*Settings*).

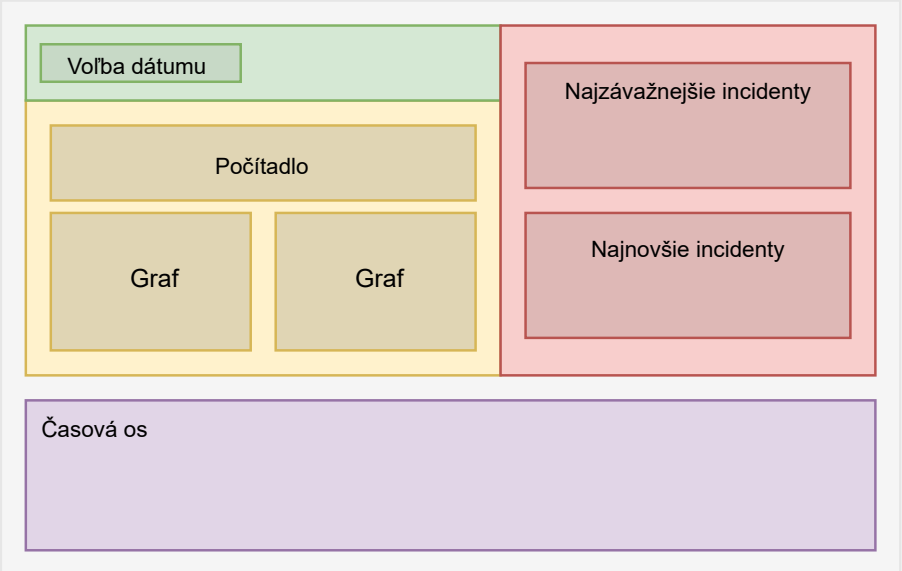
Dashboard

Po vstupe administrátora do aplikácie sa automaticky zobrazí prvá sekcia (analytická) s najnovšími dátami za aktuálny deň. Daný dátum je možné zmeniť prostredníctvom komponentu na voľbu rozsahu dátumu znázornených dát. Pri výbere je možné zvoliť jednu z rýchlych volieb – zobraziť dáta za dnešný deň, včerajší deň, za posledný týždeň, mesiac a rok – alebo podľa vlastného rozsahu. Po zvolení dátumu sa v danej sekcii vo zvolenom časovom rozmedzí aktualizujú všetky komponenty:

- **Tabuľky** – zachytávajú najnovšie a najzávažnejšie incidenty obsahujúce základné informácie, ako presný čas vzniknutia, názov útoku a jeho závažnosť, farebne rozlíšenú podľa skupiny, do ktorej patrí. Po kliknutí na riadok tabuľky sa riadok rozšíri, a tým je možné zistiť ďalšie informácie, ako je IP adresa útočníka a obete.
- **Počítadlo incidentov** – znázorňuje počet incidentov rozdelených podľa kategórie závažnosti. Incidenty sú do jednotlivých kategórií zaradené podľa oficiálneho *CVSS Ratingu*[2].
- **Grafické komponenty** – zobrazujú dve najdôležitejšie vlastnosti, ktoré je pri incidentoch nutné sledovať - najčastejšie vyskytujúce sa IP adresy útočníkov a najčastejšie sa vyskytujúce kategórie útokov.
- **Časová os** – znázorňuje výskyt incidentov v reálnom čase. Ak je rozmedzie zvoleného dátumu maximálne 24 hodín, aplikácia útoky spočíta pre každú hodinu zvlášť. V prípade, ak je zvolené rozmedzie väčšie, aplikácia počíta incidenty za každý deň zvlášť.

V prípade, že sa vo zvolenom časovom rozmedzí nenachádzajú žiadne incidenty, užívateľ obdrží upozornenie.

Na obrázku 4.1 je znázornené grafické rozmiestnenie vyššie uvedených komponentov analytickej časti webovej stránky.



Obr. 4.1: Návrh grafického zobrazenia analytickej časti webovej stránky

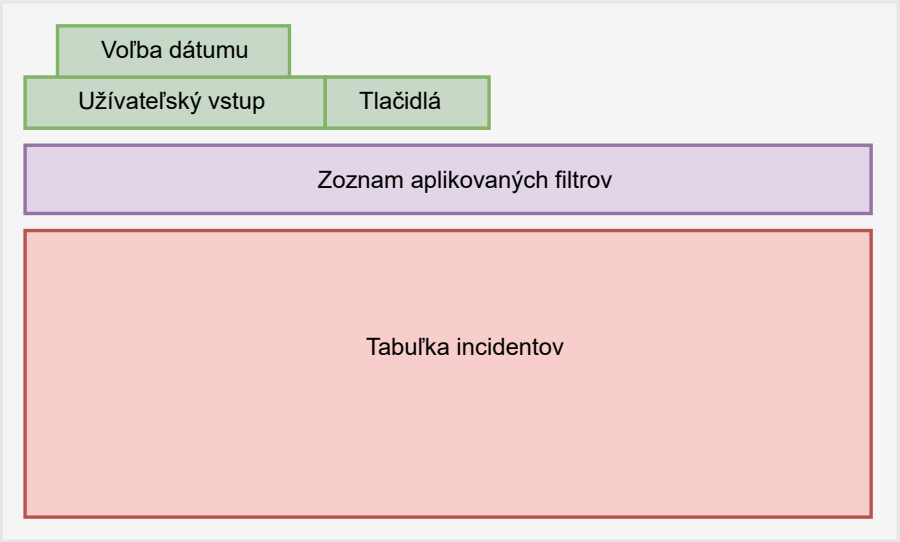
[2]Hodnotenie je dostupné na: `https://nvd.nist.gov/vuln-metrics/cvss`.

Investigate

Druhá časť aplikácie je zameraná na investigáciu incidentov (obr. 4.2). Administrátor siete si je schopný zobraziť dáta podľa zvoleného filtrovania. Filtrovanie je zaobstarané prostredníctvom užívateľského vstupu (angl. *User Input*), do ktorého je schopný vkladať dotazy na tabuľkové zobrazenie dát.

Užívateľský vstup zobrazuje administrátorovi ponúkané možnosti filtrovania (filtrovanie je možné podľa druhu útoku, závažnosti, IP adresy útočníka a obete). Užívateľ dokáže vložiť niekoľko filtrov súčasne. Každý vložený filter je užívateľovi následne zobrazený a je možné ho odstrániť samostatne, alebo pomocou tlačidla na odstránenie všetkých filtrov. Okrem dotazov je možné dáta filtrovať aj podľa dátumu pomocou komponentu vyskytujúceho sa aj v prvej časti aplikácie. Ak užívateľ zadávanie filtrov dokončil, kliknutím tlačidla na filtrovanie odosiela danú požiadavku na server.

V prípade, že sa vo zvolenom časovom období a ostatnými aplikovanými filtrami nenachádzajú žiadne incidenty, užívateľ obdrží informáciu, že žiadne dáta neboli nájdené.

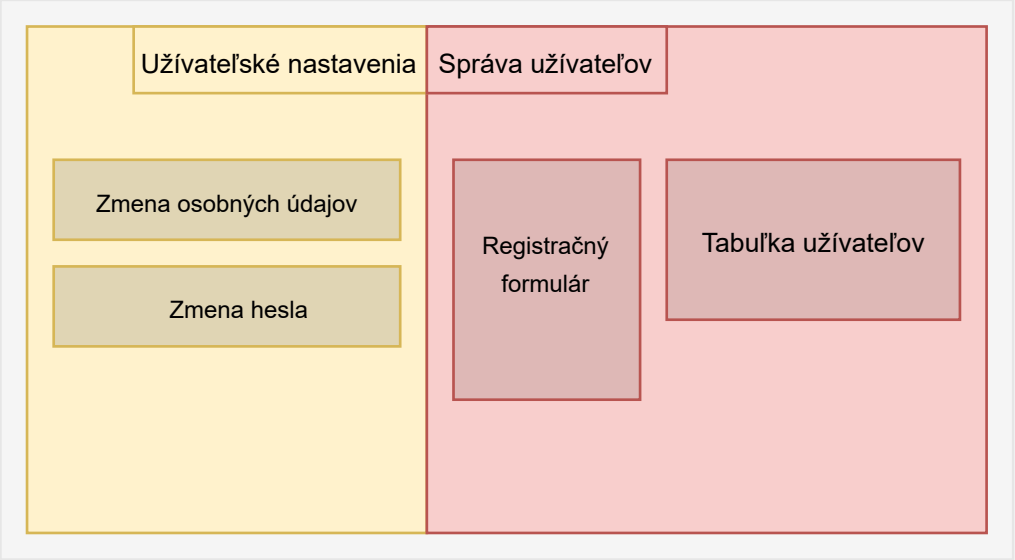


Obr. 4.2: Návrh grafického zobrazenia investigačnej časti webovej stránky

Settings

Posledná časť aplikácie je rozdelená na dve záložky (obr. 4.3). Prvá záložka je určená pre užívateľské nastavenia účtu. Daná sekcia je určené pre každého užívateľa aplikácie. V záložke je možné vykonať zmenu osobných údajov (meno a priezvisko užívateľa) a zmenu hesla využitého pri prihlasovaní sa do aplikácie. Druhá záložka nastavení je dostupná iba pre užívateľov s pridelenou rolou *Admin*. Sekcia obsahuje

formulár na registráciu nových užívateľov a tabuľku všetkých existujúcich užívateľov aplikácie a možnosťou ich vymazania v prípade, ak sa jedná o bežného užívateľa, tzn. jeho pridelená rola je *User*.



Obr. 4.3: Návrh grafického zobrazenia nastavovacej časti webovej stránky

Kompozícia webovej stránky je založená na návrhu voľne dostupnej šablóny[3] Angularu, vďaka ktorej je vytvorená správna štruktúra frontendovej časti aplikácie. Každá z vyššie uvedených sekcií (analytická, investigačná a sekcia nastavení) stránky je označovaná za rodičovský kontajner obsahujúci niekoľko odvodených komponentov. Daná kompozícia umožňuje jednotlivé komponenty vhodne prepojiť a komunikovať medzi sebou, čo znamená, že zmena v jednom diele vyvolá zmenu v ďalších dieloch, napr. zmena časového rozmedzia vyvolá zmenu zobrazených dát.

4.3.2 Zabezpečenie aplikácie

Neoddeliteľnou súčasťou webovej aplikácie je tvorba jej primeraného zabezpečenia. Pre adekvátne zabezpečenie je nutné implementovať hĺbkovú ochranu na serverovej strane aplikácie, pretože opatrenia na klientskej strane je oveľa ľahšie dostať pod kontrolu útočníka. Účelom hĺbkovej ochrany je v prípade zlyhania jednej línie zabezpečenia zastavenie útočníka v nadväzujúcej. V nasledujúcich častiach sekcie sú podrobne vysvetlené zavedené bezpečnostné opatrenia webovej aplikácie.

[3]Šablóna je dostupná na: `https://flatlogic.com/templates`.

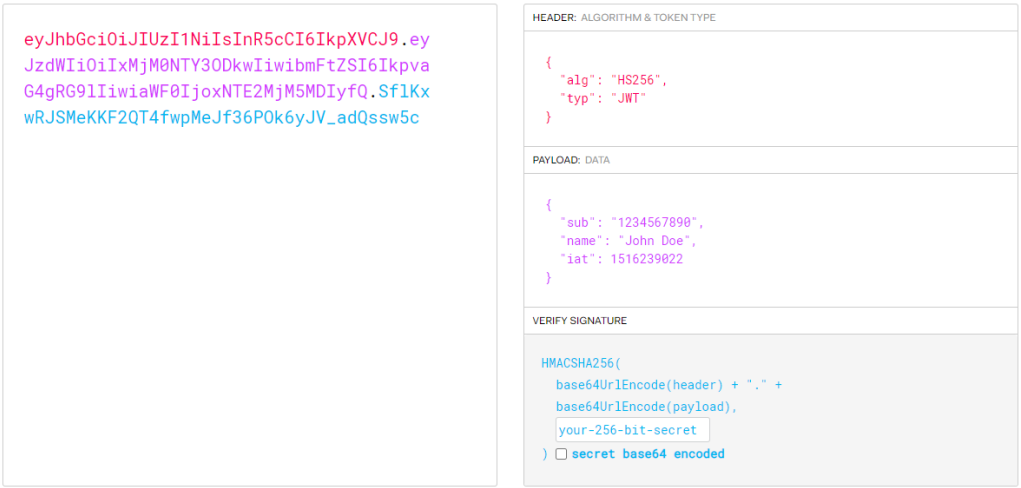
Autentizácia a autorizácia užívateľa

Z dôvodu prenosu citlivých dát je zrejmé, že by mali byť zobrazené iba oprávneným osobám.

Prvým dôležitým krokom je preto **autentizácia a autorizácia** užívateľa, čo znamená, že aplikácia musí overenie užívateľa vyžadovať pri každej citlivej operácii a musí byť implementovaná na strane servera. Opatrenie je možné dosiahnuť pomocou JWT (angl. *JSON Web Token*). JWT je definovaný ako objekt JavaScriptu. Skladá sa z hlavičky obsahujúcej informácie o danom tokene (napr. typ tokenu, druh šifrovacieho algoritmu), z *payloadu*, ktorý obsahuje necitlivé údaje o užívateľovi a podpisu, ktorý sa používa na overenie, či token obsahuje platné informácie. Jedná sa o digitálny podpis generovaný kombináciou hlavičky, payloadu a tajného kľúča uloženého na serveri.

V prípade, že sa užívateľ so zlým úmyslom pokúsi pozmeniť údaje v *payloade* na ceste od užívateľa k serveru, musí vygenerovať úplne nový podpis, na ktorý potrebuje kľúč uložený na serveri. Integritu údajov je teda možné ľahko overiť porovnaním prijatého digitálneho podpisu od klienta s novým, vypočítaným z hodnôt pochádzajúcich od klienta.

Na obrázku 4.4 je znázornená ukážka zašifrovaného JWT (vľavo) a jeho následná dešifrovaná verzia prostredníctvom šifrovacieho kľúča (vpravo).



Obr. 4.4: Vzorový náhľad *JSON Web Tokenu*[4]

Princíp využitia JWT tokenov je nasledovný [47]:

1. Klient zašle svoje údaje na server pomocou prihlasovacieho formulára.
2. Server zaslané údaje overí a v prípade, že potvrdí jeho poverenie (autentizácia), odošle mu vygenerovaný JWT token.

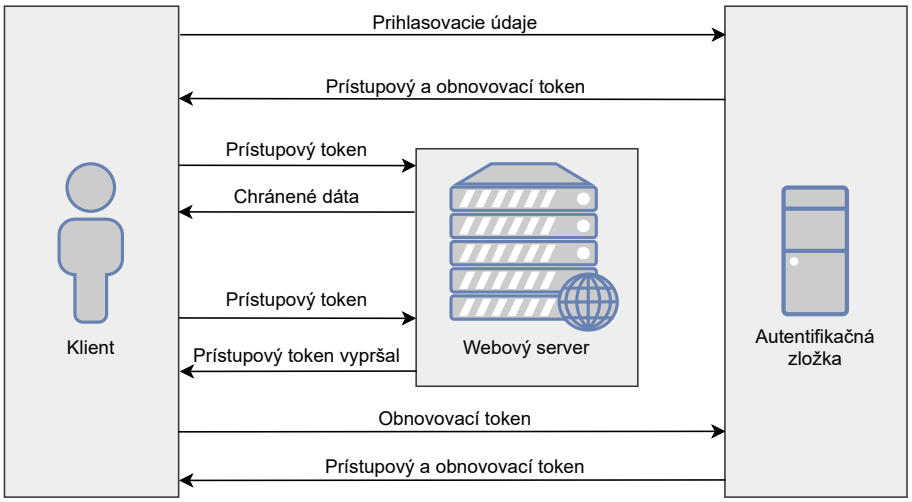
[4]Dešifrovanie JWT je dostupné na: `https://jwt.io/`.

3. Klient si JWT uloží do svojho lokálneho úložiska a je odosielaný spolu s každou požiadavkou na server, ktorý užívateľa následne jednoducho overí.

JWT má spravidla krátku dobu životnosti (napr. päť minút), tzn. že po vypršaní doby platnosti sa užívateľ musí znovu prihlásiť. Dlhá doba expirácie tokenu by spôsobila problém v tom, že ak by sa útočníkovi podarilo prístupový token získať, mohol by sa do aplikácie dostať na dlhú dobu, dokonca aj po zmene hesla. Pomocou obnovovacieho tokenu je možné, aby po vypršaní doby platnosti prístupového tokenu užívateľ nebol nútený znovu sa prihlásiť. [47]

Postup obnovy je nasledovný (obr. 4.5):

1. Užívateľ overí svoju identitu pomocou prihlasovacích údajov.
2. Server vygeneruje prístupový a obnovovací JWT.
3. Po vypršaní doby platnosti prístupového tokenu klient odošle požiadavku na obdržanie nového odoslaním obnovovacieho tokenu.
4. Predošlý bod sa opakuje, kým nevyprší doba platnosti obnovovacieho tokenu (doba platnosti je spravidla niekoľko dní).
5. Po jeho vypršaní sa klient musí znovu prihlásiť.



Obr. 4.5: Postup obnovy prístupového tokenu [47]

Ako bolo spomenuté, súčasťou JSON Web Tokenu je *payload* obsahujúci informácie o autentizovanom užívateľovi. Dané informácie (necitlivé údaje) sú označované za tzv. *claims*. Ich súčasťou je zväčša meno, priezvisko a ID užívateľa, email a jeho prislúchajúca rola (napr. administrátor, bežný užívateľ). Rola užívateľa sa využíva pri autorizácii založenej na jeho roli (angl. *role-based authorization*), pomocou ktorej sa overuje oprávnenie užívateľa na vykonanie činnosti. Keďže sa vo webovej aplikácii nachádzajú citlivé informácie, dané overovanie sa využíva pri tvorbe užívateľa (registrácia užívateľa nie je verejne dostupná, je prístupná iba administrátorovi aplikácie).

Šifrovanie a konfigurácia servera

Ďalšie dôležité opatrenie zabezpečenia aplikácie je využitie **šifrovania**. Na webový portál sa pristupuje prostredníctvom bezpečného protokolu HTTPS. Identita portálu by mala byť zabezpečená dôveryhodným certifikátom vydaným certifikačnou autoritou. V prípade bakalárskej práce nie je dôležité, aby webová aplikácia bola zabezpečená certifikátom vydaným autoritou, pretože webová aplikácia beží na lokálnom hostiteľovi (angl. *Localhost*). Z daného dôvodu sa pre šifrovanú komunikáciu využíva tzv. *self-signed Certificate*, obsahujúci vlastný privátny šifrovací kľúč. Údaje citlivé z hľadiska integrity alebo dôvernosti (napr. prihlasovacie údaje užívateľa) sa na server prenášajú pomocou zašifrovaného spojenia TLS.

Na serverovej strane sa nachádza databáza užívateľa obsahujúca prihlasovacie meno a zašifrované heslo. Databáza aplikácie nesmie uschovávať citlivé informácie, ako napríklad heslo v nezašifrovanej podobe, a preto je ukladané v podobe tzv. hashu a soli. Pri overovaní identity sa prijaté heslo od klienta hashuje využitím symetrickej šifry **SHA512** (šifra využíva šifrovacie kľúče o veľkosti 512 bitov), a tým sa porovnáva s hashom uloženým v databáze vzniknutým pri vytváraní užívateľského účtu. Pri registrácii užívateľa sa pri tvorbe hashu a soli využívajú šifrovacie kľúče generované pomocou HMAC (angl *Hash-based Message Authentication*). HMAC je mechanizmus, ktorý sa používa na výpočet náhodne generovaného kľúča.

K bezpečnostným opatreniam na serveri ďalej patrí:

- Využitie mechanizmu HSTS (angl. *HTTP Strict Transport Security*), vďaka čomu sa v hlavičke odpovede webového servera nenachádzajú informácie o použitých technológiách, serveri, internej infraštruktúre, bezpečnostných prvkoch a pod., ktoré by mohli byť útočníkom zneužité.
- Server pri pokuse o vstup pomocou HTTP protokolu nevyužíva presmerovanie na HTTPS z dôvodu nevedomého zadania citlivých údajov, ktoré by mohli byť útočníkom pri HTTP protokole zneužité (server na HTTP požiadavky nereaguje).
- Naslúchanie aplikácie na viacerých portoch predstavuje vysoké riziko vzniknutia hrozieb, ako napr. anonymné pokusy o prihlásenie. Z daného dôvodu server naslúcha na jednom porte zabezpečenom SSL certifikátom.

Pri zabezpečení webových aplikácií je dôležité poznať najčastejšie riziká a zraniteľnosti využívajúce útočníkom. OWASP (angl. *Open Web Application Security Project*) je organizácia zaoberajúca sa propagovaním bezpečnosti webových aplikácií, publikuje zoznamy najvýznamnejších bezpečnostných rizík za posledné obdobie. K najčastejším rizikám patrí [48]:

- *SQL Injection* – Upravenie vstupného parametra o ktorom je známe, že sa odosiela do nespracovaného SQL príkazu.

- *Cross-Site Scripting* – Webová stránka umožňuje útočníkovi vkladanie vlastných skriptov.
- *Cross-Site Request Forgery* – Užívateľ navštívi škodlivú stránku, ktorá vykoná nežiadúce akcie vo webovej aplikácii, v ktorej je práve autentizovaný.

Po zistení najčastejších zraniteľností webových aplikácií boli na serveri zavedené vhodné protiopatrenia. Keďže sa v sekcii na skúmanie dát nachádza vstupný parameter, ktorý odosiela požiadavky užívateľa na SQL databázu, je dôležité, aby ho aplikácia nezasielala do databázy priamo. Pri neošetrení užívateľského vstupu je možné spôsobiť vymazanie celej databázovej tabuľky, odmietnutie služby, alebo je možné inak zmeniť charakter vykonávanej operácie. Pomocou *Entity frameworku* je možné tomuto útoku zabrániť použitím metódy *Where*:

```
var data = _appDbContext.Attacks
        .Where(o => o.Timestamp == givenDate)
        .ToList();
```

Vyššie uvedený príklad využitia metódy kontroluje riadky tabuľky porovnávaním zhody vstupu užívateľa a hodnoty uloženej vlastnosti.

Cross-Site Request Forgery nastáva v prípade, že aplikácia využíva *Session-based* autentizáciu. Problém spočíva v automatickom odosielaní *Session ID* pri každej odoslanej požiadavke na aplikáciu. Keďže aplikácia využíva *Token-based* autentizáciu – pomocou JWT tokenov, ktoré sa automaticky neodosielajú – daný typ útoku nemôže nastať.

Najlepšou prevenciou pred útokom *Cross-Site Scripting* je nevkladanie akýchkoľvek nedôveryhodných dát do HTML vstupov bez ich overenia.

Bezpečnostné opatrenia sú zavedené aj na klientskej strane aplikácie. K nevyhnutným krokom patrí zobrazenie chybových hlášok v prípade nesprávneho nasmerovania URL adresy. Chybová hláška sa spravidla prepisuje na všeobecnú z toho dôvodu, že predvolená hláška obsahuje informácie o webovom serveri, ktoré by mohli byť útočníkom zneužité. Aplikácia taktiež ošetruje užívateľské chyby a výnimky, a pri registrácii ďalej vyžaduje komplexné heslo z dôvodu prevencie proti slovníkovým útokom a útokom hrubou silou.

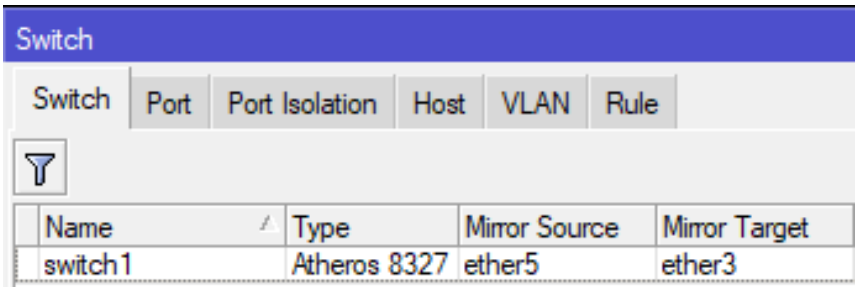
4.3.3 Experimentálne pracovisko

Po vytvorení webovej aplikácie nasleduje prenesenie experimentálneho pracoviska z lokálnej podoby vytvorenej v rámci prvej fázy do podoby reálnej. Prenesenie pozostáva z konfigurácie dvoch hlavných komponentov – Raspberry Pi a Windows Serveru.

Raspberry Pi

Konfigurácia zariadenia Raspberry Pi pozostáva z nasledujúcich krokov:

1. Konfigurácia *Port mirroringu* na routeri Mikrotik. SPAN je možné nastaviť pomocou grafického užívateľského rozhrania `Winbox`[5] nastavením portu, z ktorého sa má prevádzka zrkadliť (angl. *Mirror Source*) a portu, do ktorého sa má prevádzka zrkadliť (angl. *Mirror Target*). Na obrázku 4.6 je zobrazené nastavenie *Port mirroringu*, kde v *ether5* je zapojené monitorované zariadenie a v *ether3* zariadenie Raspberry Pi.



Obr. 4.6: Nastavenie *Port mirroringu* na routeri `Mikrotik`.

2. Konfigurácia detekčného systému Surikata. Po vykonaní úplnej inštalácie, v rámci ktorej sa nainštalujú základné pravidlá na detekciu, je potrebné pridať vlastné pravidlá detekcie, napr. na detekciu ICMP útoku (viď 5.2.1). Na Surikate je ďalej možné nastaviť, aké typy detekcií bude do logu zapisovať a ako k nim bude pristupovať – buď sa výstrahy do súboru iba zapíšu alebo sa zapíšu a škodlivé pakety sú zahodené.
3. Konfigurácia systému Filebeat. Pri systéme Filebeat sú dôležité 2 nastavenia:
   - Nastavenie vstupu definovaním cesty k súboru, ktorý má monitorovať a definovanie jeho typu.
   - Nastavenie výstupu stanovením hostiteľa databázy Elasticsearch:

     ```
     hosts: ["93.185.100.77:8172"],
     ```

     kde 93.185.100.77 je IP adresa servera, na ktorom databáza beží a : 8172 je port, na ktorom naslúcha.

Microsoft Windows Server

Microsoft Windows Server je serverový operačný systém s veľkým množstvom funkcií, ako napríklad *webhosting* a pracovanie na diaľku. Obsahuje konzolu *Server Manager* na správu všetkých serverov.

[5]Softvér na prácu s routerom `Mikrotik` je dostupný na: `https://mikrotik.com/download`.

Konzola ponúka na chod webovej aplikácie webový server IIS (angl. *Internet Information Services*). IIS patrí k populárnym webovým serverom, najmä pre ASP.NET aplikácie. Obsahuje mnoho funkcionalít, ako napríklad zabudovaná autentifikácia alebo správa TLS certifikátov na povolenie HTTPS protokolu.

Postup nasadenia webovej aplikácie na webový server je nasledovný:

1. Publikovanie aplikácie – pri nasadzovaní sa nepoužíva súbor, ktorý sa používa pri vývoji, ale súbor, ktorý vznikne publikovaním aplikácie.
2. Vytvorenie novej webovej stránky na webovom serveri IIS.
3. Presun publikovanej aplikácie do koreňovej zložky vytvorenej webovej stránky.

Vzhľadom na to, že aplikácia pracuje s lokálnou SQL databázou, je potrebné ju nainštalovať aj na Windows Server. Pomocou *ConnectionString*[6] je možné lokalizovať umiestnenie databázy absolútnou cestou. Databázu je potrebné vložiť do koreňovej zložky aplikácie ručne z dôvodu, že daná zložka vznikne až po publikovaní aplikácie.

Poslednou časťou konfigurácie serveru je inštalácia databázy Elasticsearch. Ako bolo spomenuté v časti 4.3.3, Filebeat sa napája na databázu Elasticsearch pomocou portu : 8172. Z daného dôvodu je databázu nutné na tento port nakonfigurovať.

Webový server pracuje s lokálnou SQL databázou z nasledujúcich dôvodov:

- Databáza Elasticsearch obsahuje veľké množstvo dát nevyužitých webovou aplikáciou.
- V prípade, že v databáze Elasticsearch nastane výpadok, aplikácia stále dokáže pracovať s uloženými dátami.
- Neobsahuje všetky kategórie a formát dát nevyhnutných pre webovú aplikáciu.

[6]Reťazec, ktorý špecifikuje informácie o databáze.

5 Testovanie experimentálneho pracoviska

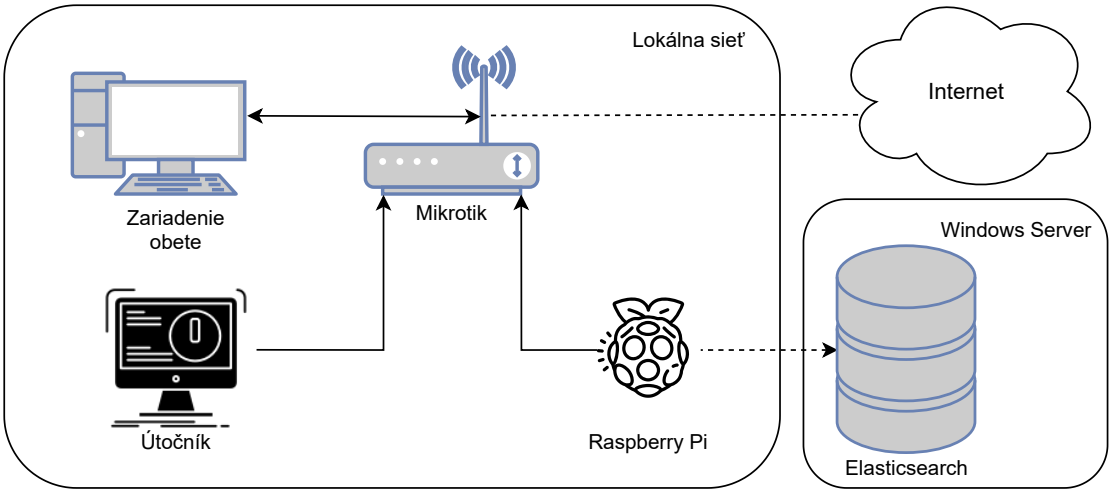
5.1 Postup testovania

Po zapojení experimentálneho pracoviska je potrebné overiť jeho funkčnosť. Na vykonanie lokálneho testovania je nutné do pracoviska zapojiť zariadenie, na ktorom sa dané útoky budú vykonávať.

Postup testovania je nasledovný:

1. Nainštalovať *Kali Linux* na zariadenie útočníka.
2. Nainštalovať potrebné nástroje na vykonanie útokov.
3. Pripojiť útočníka do siete, v ktorej sa nachádza router `Mikrotik` s pripojeným zariadením Raspberry Pi.
4. Realizovať 3 typy počítačových útokov na zariadenie obete pripojenej na routeri v lokálnej sieti.
5. Zobraziť dosiahnuté výsledky vo webovej aplikácii.

Na obrázku 5.1 je zobrazená architektúra zapojenia všetkých komponentov (vrátane útočníka) potrebných na vykonanie lokálneho testovania.



Obr. 5.1: Architektúra realizácie testovania útokov

5.2 Realizácia počítačových útokov

Na vykonanie testovania boli vybrané útoky – *ICMP Flood*, *SYN Flood*, *UDP Flood*. Všetky vyššie uvedené útoky sú realizované pomocou nástroja `Hping3`. Nástroj je možné inštalovať na zariadenie útočníka (Kali Linux) príkazom:

```
sudo apt install hping3
```

Vyhľadanie útokov vo webovej aplikácii je možné pomocou nasledujúcich informácií o ich realizácii (tab. 5.1):

Tab. 5.1: Porovnanie webových frameworkov [41]

Útok	Čas vykonania	IP adresa útočníka	IP adresa obete
ICMP Flood	29.5.2021	192.168.88.248	192.168.88.246
SYN Flood	29.5.2021	192.168.88.248	192.168.88.246
UDP Flood	29.5.2021	192.168.88.248	192.168.88.246

V nadchádzajúcich častiach sú jednotlivé útoky podrobne popísané.

5.2.1 ICMP Flood Attack

Ako bolo vysvetlené v kapitole 1.3.3, podstatou záplavového ICMP útoku je odosielanie veľkého množstva ICMP požiadaviek na adresu obete, čo spôsobí zahltenie jeho zariadenia.

Detekčný systém Surikata je nevyhnutné nakonfigurovať na zachytávanie ICMP požiadaviek. Je potrebné pridať nasledujúce pravidlo:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg :"Possible ICMP
    flood PING DoS "; flow:stateless; threshold : type both,
        track by_dst, count 100, seconds 5; icode :0; itype :8;
            sid :1; rev :6;)
```

Spustenie útoku je možné uskutočniť nasledovným príkazom:

```
sudo hping3 <cieľová_IP_adresa> --icmp --flood
```

Na zariadení obete je útok možné zachytiť pomocou softvéru `Wireshark`[1], ktorý slúži na analýzu prevádzky v počítačovej sieti.

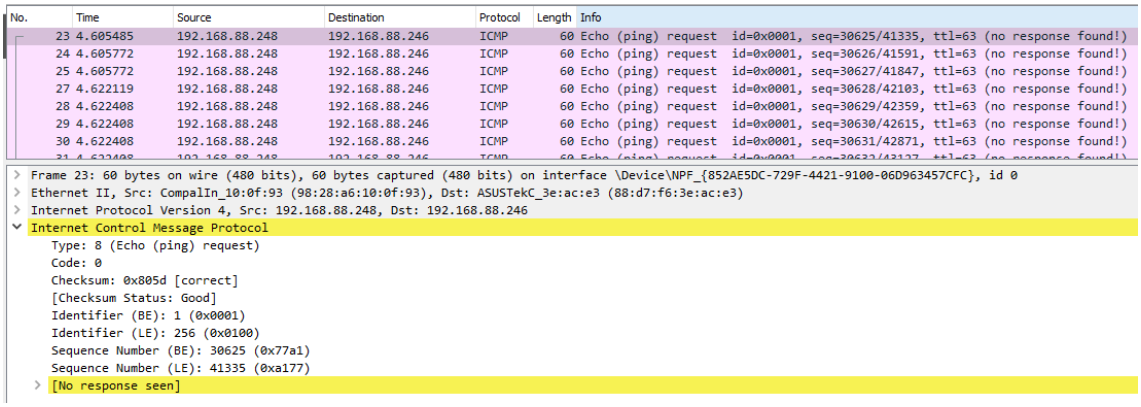
Na obrázku 5.2 je možné vidieť odosielanie veľkého množstva ICMP paketov na zariadenie obete.

5.2.2 SYN Flood Attack

Nadväzovanie spojenia medzi užívateľom a serverom pozostáva v odoslaní troch TCP paketov [49]:

1. Klient odosiela na server paket s príznakom SYN (synchronizovať).
2. Server uzná žiadosť o synchronizáciu a posiela paket s príznakom SYN a ACK (potvrdenie).
3. Klient odosiela paket s príznakom ACK.

[1]Viac informácií o softvéri je dostupných na: `https://www.wireshark.org/`.



Obr. 5.2: Zachytenie útoku *ICMP Flood* pomocou softvéru `Wireshark`

Záplavový útok SYN je druh útoku, pri ktorom útočník (klient) odosiela veľké množstvo paketov na určitý port s príznakom SYN a serveru ďalej neodpovedá [49].

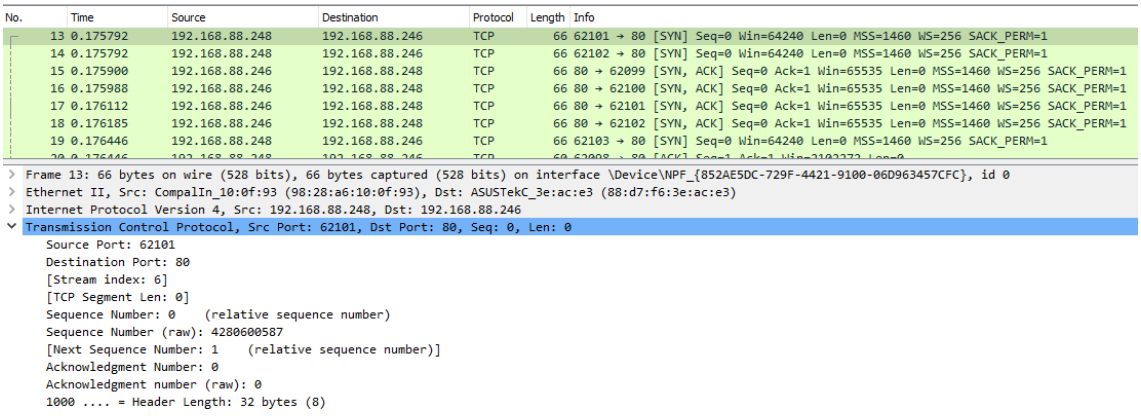
Detekciu *SYN Flood* útoku je možné zaistiť nasledovným pravidlom:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg :" Possible TCP
    SYN DoS "; flow:stateless; threshold : type both , track by_dst ,
        count 100 , seconds 5; flags :S ; sid :2; rev :1;)
```

Útok je, podobne ako vyššie uvedený útok, realizovaný pomocou nástroja `Hping3` nasledovným príkazom:

```
sudo hping3 <cieľová_IP_adresa> --syn -p 80 --flood
```

Útok je taktiež možné zachytiť pomocou softvéru `Wireshark`. Na obrázku 5.3 je možné vidieť odosielanie ICMP paketov s príznakom SYN a odpovede s príznakom SYN, ACK a žiadny paket od útočníka s príznakom ACK.



Obr. 5.3: Zachytenie útoku *SYN Flood* pomocou softvéru `Wireshark`

5.2.3 UDP Flood Attack

UDP Flood útok je ďalší druh záplavového útoku, ktorého podstatou je odosielanie UDP (angl. *User Datagram Protocol*) paketov útočníkom s cieľom zahltiť náhodné porty na hostiteľovi. [50]

Pri tomto type útoku systém vyhľadáva aplikácie naslúchajúce na daných portoch. V prípade, že žiadnu aplikáciu nenájde, vydá paket *Destination Unreachable* späť útočníkovi. Systém je tak zaneprázdnený vyhľadávaním aplikácií, že nereaguje na oprávnenú premávku. [50]

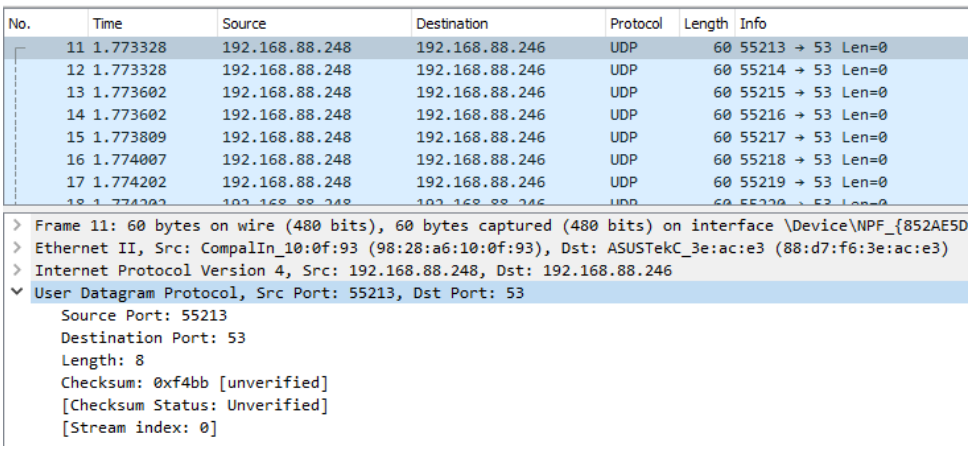
Detekciu *UDP Flood* útoku je možné zaistiť nasledovným pravidlom:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"Possible UDP DoS";
    flow:stateless; threshold: type both, track by_dst, count 100,
        seconds 10;  sid:3;)
```

Útok je taktiež realizovaný pomocou nástroja `Hping3` nasledovným príkazom:

```
sudo hping3 <cieľová_IP_adresa> --udp --flood
```

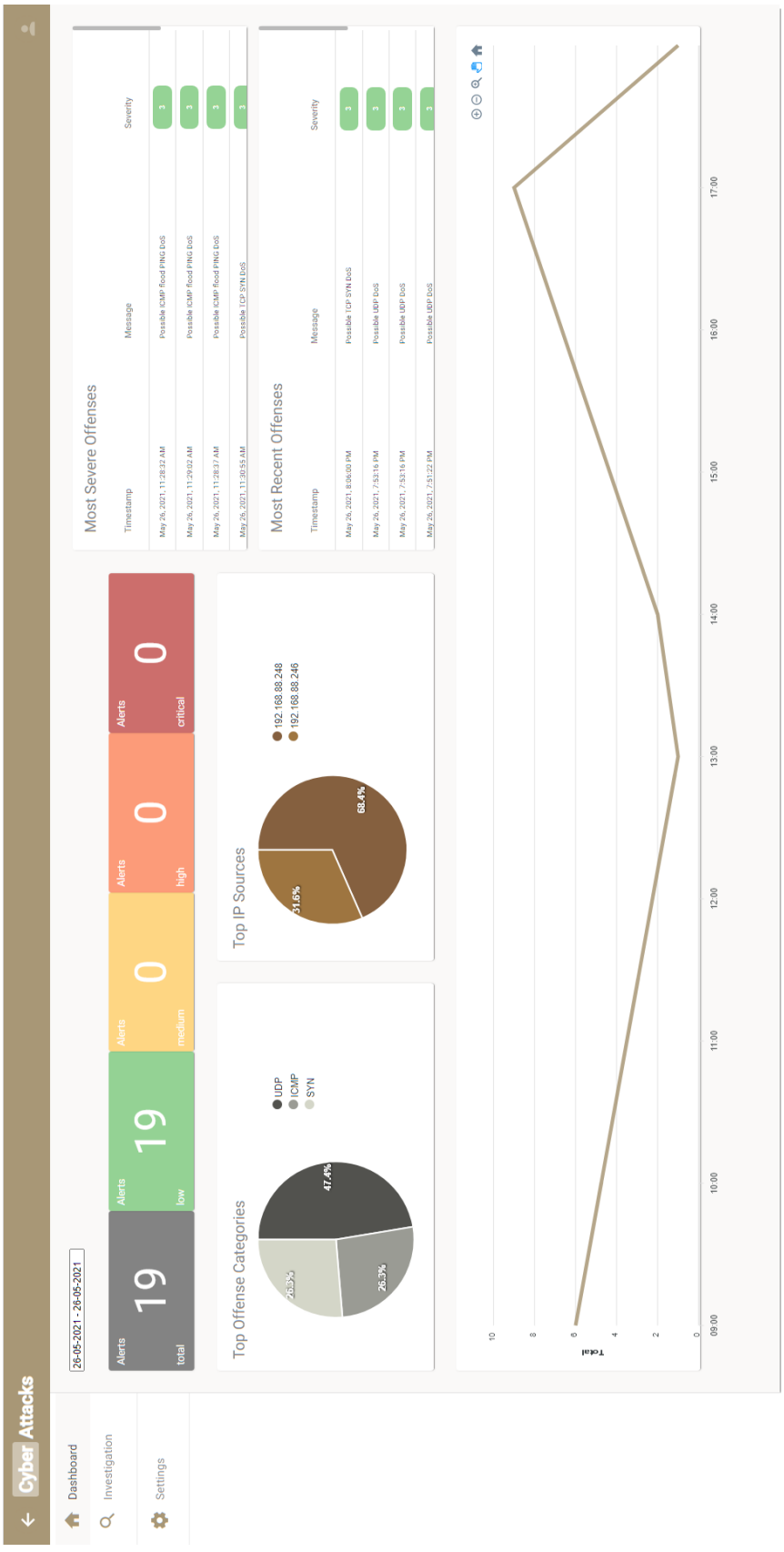
UDP Flood je na zariadení obete možné zachytiť monitorovaním UDP paketov (obr. 5.4).



Obr. 5.4: Zachytenie útoku *UDP Flood* pomocou softvéru `Wireshark`

5.3 Dosiahnuté výsledky testovania

Po úspešnej realizácii počítačových útokov je možné zobraziť si dosiahnuté výsledky. Po funkčnom zapojení pracoviska, experimentálne pracovisko podľa obrázku 2.3 a aplikácie podľa obrázku 2.4, systém Filebeat odosiela zachytené výstrahy útokov do databázy Elasticsearch umiestnenej na webovom serveri, kde ich ďalej spracováva webová aplikácia. Funkčnosť pracoviska je overená vo webovej aplikácii nasledujúcimi obrázkami 5.5 a 5.6.



Obr. 5.5: Výsledok testovania v sekcii *Dashboard*

Obr. 5.6: Výsledok testovania v sekcii *Investigate*

Záver

Na záver bakalárskej práce je možné povedať, že všetky kroky praktickej implementácie stanovené v časti 4.1 boli úspešne splnené. Cieľom práce bolo vytvorenie webovej aplikácie v experimentálnom pracovisku umožňujúca zobraziť počítačové incidenty a previesť ich investigáciu.

V rámci teoretickej časti bakalárskej práce bola vykonaná štúdia problematiky počítačových útokov. Na začiatku prvej kapitoly boli popísané základné definície pojmov týkajúcich sa danej tematiky (viď 1.1). Nasledovalo zoznámenie sa so spôsobmi detekcie a prevencie prienikov a s ich systémami (viď 1.2). Posledná časť prvej kapitoly sa zaoberala vybranými druhmi počítačových útokov vyskytujúcich sa na dátovej a sieťovej vrstve (viď 1.3).

Obsahom druhej kapitoly bola charakteristika výsledného riešenia celej práce. Kapitola bola rozdelená na dve časti. Prvá časť kapitoly sa venovala popisu zapojenia celého pracoviska. Na úspešnú detekciu útokov bolo potrebné nadobudnutie znalostí o jednotlivých komponentoch nevyhnutných pre funkčnosť pracoviska (viď 2.1). Sekcia taktiež obsahovala definíciu formátu a štruktúry logov (viď 2.1.1). Druhá časť kapitoly sa venovala zapojeniu webovej aplikácie a popisu komponentov využitých pri jej vývoji (viď 2.2).

Tretia kapitola sa zaoberala tvorbou návrhu webovej aplikácie. Na začiatku bolo nevyhnuté vykonať analýzu aplikácie, ktorá sa skladala z tvorby modelu prípadu užitia (viď 3.1.1) a špecifikácie požiadaviek (viď 3.1.2) z dôvodu tvorby úspešného základu návrhu danej webovej aplikácie. Ďalším krokom bolo potrebné vykonať porovnanie dostupných webových frameworkov (viď 3.2). Súčasťou kapitoly bol aj prieskum dostupných SIEM nástrojov z dôvodu získania prehľadu, akými typmi grafických zobrazení je možné počítačové incidenty reprezentovať (viď 3.3). Na konci kapitoly bol vytvorený návrh riešenia práce tvorený výberom vhodných frameworkov na tvorbu webovej aplikácie a prvotný návrh implementácie aplikácie (viď 3.4).

Obsahom štvrtej kapitoly bola charakteristika praktického výstupu práce. Kapitola sa venovala vopred definovaným jednotlivým krokom, ktoré rozložili prácu na dve fázy (viď 4.1). V prvej fáze bolo zostrojené experimentálne pracovisko na lokálnej sieti z dôvodu porozumenia, ako jednotlivé komponenty spolu súvisia, a akým spôsobom sú prepojené. V prvej fáze bola taktiež implementovaná základná časť backendovej aplikácie potrebná na správnu konektivitu celého pracoviska (viď 4.2). V druhej etape bola webová aplikácia rozšírená o GUI (viď 4.3.1) – sekcia *Dashboard*, *Investigation* a *Settings* – a o zabezpečenie aplikácie najmodernejšími technikami (viď 4.3.2). V závere kapitoly je pracovisko prenesené do reálnej podoby (viď 2.1).

Práca bola zakončená výslednou kapitolou o testovaní zostaveného pracoviska. Na začiatku bol vysvetlený postup overenia funkčnosti pracoviska (viď 4.1). Kapi-

tola ďalej obsahuje stručnú charakteristiku realizovaných útokov (*SYN Flood*, *ICMP Flood* a *UDP Flood*), pravidlá na ich detekciu, príkazy spustenia a výstup v softvéri `Wireshark` (viď. 5.2). V závere kapitoly je zosumarizovaný výsledok celého testovania (viď 5.3), jeho súčasťou sú aj snímky obrazovky, ktoré dokazujú funkčnosť experimentálneho pracoviska a webovej aplikácie.

Literatúra

[1] MANSHIP, Ryan. *The Information Security Industry: Understanding and Evaluating Service Providers* [online]. [cit. 2020-11-21]. Dostupné z URL: https://www.redteamsecure.com/blog/why-is-information-security-important/.

[2] ČANDÍK, Marek. *Informační bezpečnost* [online]. [cit. 2020-11-21]. Dostupné z URL: https://moodle.unob.cz/pluginfile.php/15173/mod_resource/content/2/Informa%C4%8Dn%C3%AD%20bezpe%C4%8Dnost.pdf.

[3] *Slovník kybernetickej bezpečnosti* [online]. [cit. 2020-11-21]. Dostupné z URL: https://cybersec.sk/slovnik-kybernetickej-bezpecnosti/.

[4] JIRÁSEK, Petr, Luděk NOVÁK a Josef POŽÁR. *Výkladový slovník kybernetické bezpečnosti* [online]. Praha, 2012 [cit. 2020-11-18]. Dostupné z URL: https://afcea.cz/wp-content/uploads/2015/03/Slovnik_Final_screen_v2_0.pdf.

[5] Barracuda. *What is a Intrusion Detection System?* [online]. [cit. 2020-12-01]. Dostupné z URL: https://www.barracuda.com/glossary/intrusion-detection-system.

[6] ENDORF, Carl F., Eugene SCHULTZ a Jim MELLANDER. *Detekce a prevence počítačového útoku*. Praha: Grada, 2005. ISBN 80-247-1035-8.

[7] PETTERS, Jeff. *How Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) Work* [online]. [cit. 2020-11-21]. Dostupné z URL: https://www.varonis.com/blog/ids-vs-ips/.

[8] WU, Handong; SCHWAB, Stephen; PECKHAM, Robert Lom. *Signature based network intrusion detection system and method* [online]. [cit. 2020-11-21]. Dostupné z URL https://patentimages.storage.googleapis.com/4e/4b/37/e1c55f53b4409c/US7424744.pdf.

[9] JYOTHSNA, Veeramreddy; PRASAD, Rama; PRASAD, Munivara. *A review of anomaly based intrusion detection systems*. International Journal of Computer Applications, 2011, 28.7: 26-35. 10.5120/3399-4730.

[10] FEKOLKIN, Roman. *Intrusion Detection and Prevention Systems: Overview of Snort and Suricata* [online]. [cit. 2020-11-09]. Dostupné z URL: https://www.researchgate.net/publication/297171228_Intrusion_Detection_and_Prevention_Systems_Overview_of_Snort_and_Suricata.

[11] *Top 10 BEST Intrusion Detection Systems (IDS) [2021 Rankings]* [online]. [cit. 2021-02-08]. Dostupné z URL: https://www.dnsstuff.com/network-intrusion-detection-software#solarwinds-security-event-manager.

[12] MARTINÁSEK, Zdeněk. *Problematika logování, systémy IDS a IPS*. Brno: Vysoké učení technické v Brně, 2020. Dostupné z kurzu: https://www.vutbr.cz/studenti/predmety/detail/224181.

[13] SINGH, P.K., A.K. KAR, Y. SINGH, M.H. KOLEKAR a S. TANWAR. *Proceedings of ICRIC 2019: Recent Innovations in Computing*. Switzerland: Springer Nature, 2019. ISBN 9783030294076.

[14] Quadrant. *The Sagan Log Analysis Engine* [online]. [cit. 2020-11-21]. Dostupné z URL: https://quadrantsec.com/sagan_log_analysis_engine/.

[15] ČÍKA, Petr. *Multimediální služby*. Brno: Vysoké učení technické v Brně, 2012. ISBN 978-80-214-4443-0.

[16] LOCKHART, Andrew. *Network Security Hacks: Tips & Tools for Protecting Your Privacy*. California: O'Reilly Media. ISBN 978-0596527631.

[17] Veracode. *ARP Spoofing* [online]. [cit. 2020-10-31]. Dostupné z URL: https://www.veracode.com/security/arp-spoofing.

[18] JAJODIA, Sushil a Chandan MAZUMDAR. *Information Systems Security: First International conference, ICISS 2005, Kolkata, India, December 19-21, 2005, Proceedings*. Switzerland: Springer. ISBN 9783540324225.

[19] PROWSE, David. L. *CompTIA Security+ SYO-301 Cert Guide, Deluxe Edition: CompT Secur SY030 Cert Gui_2*. Londýn: Pearson Education, 2011. ISBN 9780132801294.

[20] BOYLES, Tim. *CCNA Security Study Guide*. New Jersey: John Wiley, 2010. ISBN 9780470636336.

[21] DigitalFortressLK. *Common Attack Types on Switches* [online]. [cit. 2020-11-10]. Dostupné z URL: https://digitalfortresslk.wordpress.com/2018/03/22/common-attack-types-on-switches/.

[22] PETTERS, Jeff. *What is a Man-in-the-Middle Attack: Detection and Prevention Tips* [online]. [cit. 2020-11-21]. Dostupné z URL: https://www.varonis.com/blog/man-in-the-middle-attack/.

[23] NOHE, Patrick. *Executing a Man-in-the-Middle Attack in just 15 Minutes* [online]. [cit. 2020-12-06]. Dostupné z URL: https://www.thesslstore.com/blog/man-in-the-middle-attack-2/.

[24] DE RYCK, Philippe, Lieven DESMET, Frank PIESSENS a Martin JOHNS. *Primer on Client-Side Web Security*. Cham: Springer, 2014. ISBN 978-331-9122-267.

[25] Trifa, Zied, Maher, Khemakhem. *Sybil Nodes as a Mitigation Strategy Against Sybil Attack*, [online]. [cit. 2020-11-12]. Dostupné z URL: https://www.sciencedirect.com/science/article/pii/S1877050914007443.

[26] Netscout. *ICMP Flood Attacks* [online]. [cit. 2020-11-11]. Dostupné z URL: https://www.netscout.com/what-is-ddos/icmp-flood.

[27] SHEA, Sharon. *How to prevent network eavesdropping attacks* [online]. [cit. 2020-11-09]. Dostupné z URL: https://searchsecurity.techtarget.com/answer/How-to-prevent-network-sniffing-and-eavesdropping.

[28] Raspberry Pi Foundation. *Raspberry Pi Foundation* [online]. [cit. 2020-11-21]. Dostupné z URL: https://www.raspberrypi.org/.

[29] Elastic. *Filebeat overview* [online]. [cit. 2020-11-13]. Dostupné z URL: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html.

[30] SIX, Brad. *Log Formats – A (mostly) complete Guide* [online]. In: . 15.1.2020 [cit. 2020-11-21]. Dostupné z URL: https://www.graylog.org/post/log-formats-a-complete-guide.

[31] Elastic. *What is Elasticsearch?* [online]. [cit. 2020-11-13]. Dostupné z URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html.

[32] PASTORINO, Marcelo. *Frontend vs Backend: What's the Difference?* [online]. [cit. 2020-11-21]. Dostupné z URL: https://www.pluralsight.com/blog/software-development/front-end-vs-back-end.

[33] SAFONOV, Yehor. *Aplikace pro grafickou reprezentaci průběhu útoku*. Brno, 2018, 95 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

[34] MISTRY, Jigar. *A Detailed Comparison of 10 Popular Web Development Framework* [online]. [cit. 2020-11-19]. Dostupné z URL: https://www.monocubed.com/web-development-framework-comparison/.

[35] DHADUK, Hiren. *Best Frontend Frameworks of 2020 for Web Development* [online]. [cit. 2020-11-17]. Dostupné z URL: https://www.simform.com/best-frontend-frameworks/.

[36] PATEL, Manish. *10 Famous Apps Using React.js* [online]. [cit. 2020-12-03]. Dostupné z URL: https://dzone.com/articles/10-famous-apps-using-reactjs-nowadays.

[37] CHIARETTA, Simone. *Front-end Development with ASP.NET Core, Angular, and Bootstrap*. New Jersey: John Wiley, 2018. ISBN 9781119181408.

[38] FLEURY, Daniel. *12 Companies That Use Angular in 2020 Successfully* [online]. [cit. 2020-12-03]. Dostupné z URL: https://trio.dev/blog/companies-use-angular.

[39] MIŚTA, Wojciech. *12 Companies That Have Utilized Vue.js in Their Applications* [online]. [cit. 2020-12-03]. Dostupné z URL: https://naturaily.com/blog/companies-vue-js-applications.

[40] SPURLOCK, Jake. *Bootstrap: Responsive Web Development*. Newton: O'Reilly Media, 2013. ISBN 9781449344597.

[41] CLARK, Jessica. *Top 10 backend frameworks* [online]. [cit. 2020-11-19]. Dostupné z URL: https://blog.back4app.com/backend-frameworks/.

[42] WILLIAMS, Martin. *Top 8 Best Backend Frameworks* [online]. [cit. 2020-11-22]. Dostupné z URL: https://www.keycdn.com/blog/best-backend-frameworks.

[43] Credo Systemz. *ASP.NET with Angular Training* [online]. [cit. 2020-11-22]. Dostupné z URL: https://www.credosystemz.com/training-in-chennai/asp-dot-net-mvc-with-angularjs-training/.

[44] PETTERS, Jeff. *What is SIEM? A Beginner's Guide* [online]. [cit. 2021-02-10]. Dostupné z URL: https://www.varonis.com/blog/what-is-siem/.

[45] KEARY, Tim. *10 Best SIEM Tools of 2021: Vendors & Solutions Ranked* [online]. [cit. 2021-03-02]. Dostupné z URL: https://www.comparitech.com/net-admin/siem-tools/.

[46] SOFTWARE TESTING HELP. *Top 11 Best SIEM Tools In 2021 For Real-Time Incident Response And Security* [online]. [cit. 2021-03-03]. Dostupné z URL: `https://www.softwaretestinghelp.com/siem-tools/`.

[47] WATMORE, Jason. *Jason Watmore's Blog* [online]. [cit. 2021-5-18]. Dostupné z URL: `https://jasonwatmore.com/`.

[48] SYNOPSYS. *OWASP Top 10* [online]. In: . [cit. 2021-5-18]. Dostupné z URL: `https://www.synopsys.com/glossary/what-is-owasp-top-10.html`.

[49] CLOUDFLARE. *SYN flood attack* [online]. [cit. 2021-5-18]. Dostupné z URL: `https://www.cloudflare.com/en-gb/learning/ddos/syn-flood-ddos-attack/`.

[50] CLOUDFLARE. *UDP flood attack* [online]. [cit. 2021-5-18]. Dostupné z URL: `https://www.cloudflare.com/en-gb/learning/ddos/udp-flood-ddos-attack/`.

Zoznam symbolov a skratiek

ARP	Address Resolution Protocol – Protokol rozlišovania adries
CAM	Context-Addressable Memory – Asociatívna pamäť
CSS	Crew Safety System – Kaskádové štýly
DNS	Domain Name System – Systém doménových mien
DOM	Document Object Model – Objektový model dokumentu
GUI	Graphical User Interface – Grafické užívateľské rozhranie
HIDS	Host-Based Intrusion Detection System – Systém detekcie prieniku na hostiteľskom počítači
HIPS	Host-Based Intrusion Detection System – Systém prevencie prieniku na hostiteľskom počítači
HMAC	Hash-based Message Authentication – Autentifikácia správ založená na hašovaní
HTML	Hypertext Markup Language – Hypertextový značkový jazyk
HTTP	Hypertext Transfer Protocol – Hypertextový prenosový protokol
HTTPS	Hypertext Transfer Protocol Secure – Zabezpečený hypertextový prenosový protokol
ICMP	Internet Control Message Protocol – Internetový kontrolný protokol
IDS	Intrusion Detection System – Systém na odhalenie prieniku
IIS	Internet Information Services – Internetové informačné služby
IP	Internet Protocol – Internetový protokol
IPS	Intrusion Prevention System – Systém na prevenciu prieniku
ISO/OSI	Open Systems Interconnection Reference Model – Referenčný model prepojenia otvorených systémov
JSON	JavaScript Object Notation – Označenie JavaScript objektu
JWT	JSON Web Token – JSON webový token
MAC	Media Access Control – Riadenie prístupu k médiu

MitM	Man-in-the-Middle – Človek uprostred
MVC	Model-View-Controller – Model architektúry aplikácie
NIDS	Network-Based Intrusion Detection System – Systém detekcie prieniku siete
NIPS	Network-Based Intrusion Detection System – Systém prevencie prieniku siete
NoSQL	Non-Structured Query Language – Neštruktúrovaný dotazovací jazyk
SIEM	Security Incident and Event Management – Správa bezpečnostných incidentov a udalostí
SPAN	Switched Port Analyzer – Zrkadlenie portov
SQL	Structured Query Language – Štruktúrovaný dotazovací jazyk
SSH	Secure Shell – Zabezpečený prístup k príkazovému riadku
STP	Spanning Tree Protocol – Protokol odstraňujúci slučky siete
TLS	Transport Layer Security – Šifrovací protokol
UDP	User Datagram Protocol – Používateľský datagramový protokol
UML	Unified Modeling Language – grafický jazyk na vizualizáciu programových systémov
URL	Uniform Resource Locator – Webová adresa
VLAN	Virtual Local Area Network – Virtuálna lokálna sieť
WPA3	Wi-Fi Protected Access 3 – Bezdrôtová šifrovacia metóda

Zoznam príloh

A Návod na spustenie aplikácie

Webovú aplikáciu je možné spustiť nasledujúcimi spôsobmi:

- Kompiláciou zdrojového kódu.
- Stiahnutím a spustením virtuálnej stanice.
- Prístupom na webový server.

A.1 Kompilácia zdrojového kódu

Prvým spôsobom overenia funkčnosti webovej aplikácie je prevedenie kompilácie zdrojového kódu na priloženom médiu alebo vykonanie príkazu na klonovanie repozitára[1]:

```
git clone https://github.com/xmatus32/BP.git
```

Na úspešnú kompiláciu zdrojového kódu sú potrebné nasledujúce nástroje:

- `npm package manager` – správca balíkov je nainštalovaný pomocou *Node.js*[2].
- `Microsoft .Net SDK` (verzia `.NET Core 3.1 Runtime`)[3] – na spustenie `.NET Core` aplikácií pomocou konzoly.
- `VC_redist` – distribučný balíček[4] potrebný na inštaláciu SQL databázy.
- `LocalDB` – lokálna databáza je inštalovaná pomocou `SQL Server Express 2019`[5], stiahnutím média (angl. *Download Media*) `LocalDB`.

Po nainštalovaní všetkých vyššie uvedených balíčkov a nástrojov je potrebné vytvoriť `build` serverovej časti aplikácie. Po navigovaní do zložky `/backend/bakalarska_praca` pomocou konzoly je potrebné spustiť príkaz:

```
dotnet build
```

Príkazom sa nainštalujú všetky závislosti aplikácie a zostaví sa celý projekt. Pred spustením je potrebné prekopírovať súbor databázy `bakalarska_praca.mdf` do zložky projektu `/bin/Debug/netcoreapp3.1`.

Spustenie backendovej časti aplikácie je možné príkazom:

```
dotnet run --urls=https://localhost:44386/
```

[1] Repozitár je dostupný na: `https://github.com/xmatus32/BP`.
[2] Dostupný na `https://nodejs.org/en/`.
[3] Dostupná na: `https://dotnet.microsoft.com/download/visual-studio-sdks`.
[4] Dostupný na: `https://www.microsoft.com/en-us/download/details.aspx?id=48145`.
[5] Dostupný na: `https://www.microsoft.com/en-us/sql-server/sql-server-downloads`.

Na spustenie klientskej časti je potrebné nainštalovať `Angular CLI` a vykonať `build` aplikácie pomocou príkazov:

```
npm install -g @angular/cli
ng build
```

V prípade, že tvorba projektu zlyhá, je potrebné aktualizovať balíčky príkazom `npm update`.

Príkazom `ng serve` sa spustí klientská časť aplikácie. Aplikácia je prístupná na URL adrese `https://localhost:4200/`. Ak sa vyskytne problém s certifikátom, príkazom `ng serve -ssl false` sa spustí nezabezpečená verzia aplikácie.

A.2 Spustenie virtuálnej stanice

Spustenie aplikácie vo virtuálnej stanici je oproti kompilácii zdrojového kódu jednoduchšie vďaka tomu, že stanica obsahuje nainštalované vyššie uvedené kroky a nie je nutná žiadna ďalšia konfigurácia.

Virtuálna stanica je dostupná na úložisku[6] *Google Drive* pre členov organizácie VUT.

Postup zostavenia je nasledovný:

1. Nainštalovať aplikáciu na prácu s virtualizovaným prostredím, napr. `VirtualBox`[7].
2. Importovať virtuálnu stanicu do nainštalovanej aplikácie.
3. Spustiť virtuálnu stanicu.

Spustenie backendovej časti aplikácie je možné príkazom v konzole *Command Prompt*:

```
dotnet run --urls=https://localhost:44386/
```

Príkazom `ng serve` v ďalšej konzole *Command Prompt* sa spustí klientská časť aplikácie. Aplikácia je prístupná na URL adrese `https://localhost:4200/`.

V súbore *README.txt* umiestnenom na pracovnej ploche virtuálnej stanice sú dostupné ďalšie informácie ohľadom webovej aplikácie (napr. prihlasovacie údaje).

A.3 Prístup na Windows Server

Overenie funkčnosti webovej aplikácie je po dohode možné aj prostredníctvom prístupu na Windows Server.

[6] Dostupná na: `https://drive.google.com/drive/folders/1KVkOSS4HEcpgYRN-5qQGOuz7_Xhd18Fu?usp=sharing`.

[7] Aplikácia je dostupná na: `https://www.virtualbox.org/`.

B Obsah priloženého média

K textu bakalárskej práce sa prikladá optické médium obsahujúce zdrojový kód webovej aplikácie dôležitý na overenie jej funkčnosti, databázový súbor obsahujúci incidenty vygenerované v dobe testovania aplikácie. Zdrojový kód je možné spustiť podľa návodu uvedeného v prílohe A.1.

Adresárová štruktúra optického média je nasledovná:

- `/` ..... koreňový adresár média
  - `backend` ..... adresár serverovej časti aplikácie
    - `bakalarskaPraca.mdf` ..... databázový súbor obsahujúci generované dáta
    - `bakalarska_praca` ..... adresár na kompiláciu backendu
      - `Controllers` ..... triedy na prácu s požiadavkami užívateľov
      - `Extensions` ..... rozšírenie servisných funkcií
      - `Migrations` ..... databázové migrácie
      - `Models` ..... objektoré triedy aplikácie
      - `Services` ..... pomocné funkcie *Controllerov*
      - `Program.cs` ..... predvolený súbor ASP.NET aplikácií
      - `Startup.cs` ..... vstupná konfigurácia aplikácie pri jej spustení
      - `appsettings.json` ..... konfiguračný užívateľský súbor
  - `frontend` ..... adresár klientskej časti aplikácie
    - `src`
      - `app` ..... koreňový súbor komponentov
        - `consts` ..... konštanty farieb a trás
        - `pages`
          - `auth` ..... sekcia prihlasovania sa
          - `dashboard` ..... sekcia *Dashboard*
          - `not-found` ..... sekcia v prípade zlého smerovania
          - `settings` ..... sekcia nastavení užívateľa
          - `tables` ..... sekcia na filtrovanie dát
        - `shared` ..... komponenty zdieľané v každej sekcii
        - `styles` ..... definované farby, premenné, písmo
        - `app-routing.module.ts` ..... súbor na smerovanie
        - `app-module.ts` ..... globálne moduly aplikácie
      - `assets` ..... použité obrázky
      - `index.html` ..... predvolená stránka aplikácie
      - `main.ts` ..... predvolený konfiguračný súbor