Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze

Fakulta informačních technologií

Department of Katedra teoretické informatiky

Diplomová práce

Analýza bezpečnostních rizik aplikací z logů v reálném čase

Bc. Vojtěch Krákora

Vedoucí práce: Pavel Pivoňka, GWCPM

30. dubna 2017

Poděkování

THANKS (remove entirely in case you do not with to thank anyone)

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Prague dne 30. dubna 2017 .....

Odkaz na tuto práci

Krákora, Vojtěch. *Analýza bezpečnostních rizik aplikací z logů v reálném čase.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Obsah

Seznam obrázků

Seznam tabulek

Introduction

[[Napsat max jednu stranku]] S roustoucím významem informačních systémů a informací jenž jsou zpracovány počítaočovými systémy je kladen důraz na zabezpečení informací. Bezpečnost informačních systémů je zajišťována zákonem o kybernetické bezpečnosti 181/2014 Sb [2]. Mnoho systémů vyžaduje i vyšší zabezpečení například pomocí ISO normy 27001 [3].

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Kapitola 1

Úvod do problematiky

1.1 Kybernetická bezpečnost

Pro dnešní dobu je běžné využívání informačních technologíích prakticky kdekoliv. Jde o stavební kámen mnoha podniků. Informačním systémem rozumíme kombinaci softwaru, hardwaru, infrastruktury a trénovaného personálu [4] . Tam, kde se informační systémy vyskytují pomáhájí práci zjednodušovat, tvořit či se jinak podílet. Systémy je třeba udržovat v provozu a co nejvíce se vyhnout všem možným rizikům, které mohou businessový proces narušit.

Obecně lze na zabezpečení informačního systému nahlížet z mnoha různých úhlů. Zabezpečit lze síť pomocí certifikovaného přístupu, správně zabezpečené bezdrátové sítě a podobně. Do operačních systémů se nainstalují antivirové programy. Samotná zařízení se fyzicky uzamknou a znemožní se přístup nepovolaným osobám.

Povinnost zabezpečovat informační systémy přikládá zákon o kybernetické bezpečnosti 181/2014 Sb. Tento zákon stanovuje povinnosti a práva orgánů veřejné moci v oblasti kybernetické bezpečnosti [2]. Tento zákon mimo jiné nařizuje orgánům veřejné moci **[[Definovat orgán veřejné moci]]** povinnost zajišťovat kybernetickou bezpečnost.

Kromě zákonu jsou k dispozici normy. Jednou z takových norem je norma ISO 27001. Tato norma stanovuje požadavky, které je nutné dodržovat pro kybernetickou bezpečnost. Tyto požadavky jsou stanoveny jak na samotné informační systémy, tak i na zaměstnance, informační procesy, či strategie firmy. [3].

Lze říci, že zajištění kybernetické bezpečnosti a následné detekci jednotlivých bezpečnostních rizik je velmi specifické. Každá danná doména má své rizika a jiné druhy způsobů jejich detekce.

V práci řeším kybernetickou bezpečnost pro konkrétní část informačního systémy. Podstatná tedy jsou bezpečností rizika na integrační platformě Unify. Definice integrační platformy a její popis je v sekci 1.2.

1.1.1 Bezpečnostní rizika

[[Nadefinovat bezpečnostní rizika]] Jak již bylo uvedeno pro tuto práci jsou podstatná bezpečností rizika na konkrétní integrační platformě. Cílem je hledat pouze ta, která vznikají na straně programu. Zabezpečení přístupů fyzicky k serverům Unify a podobně nebudou probírána.

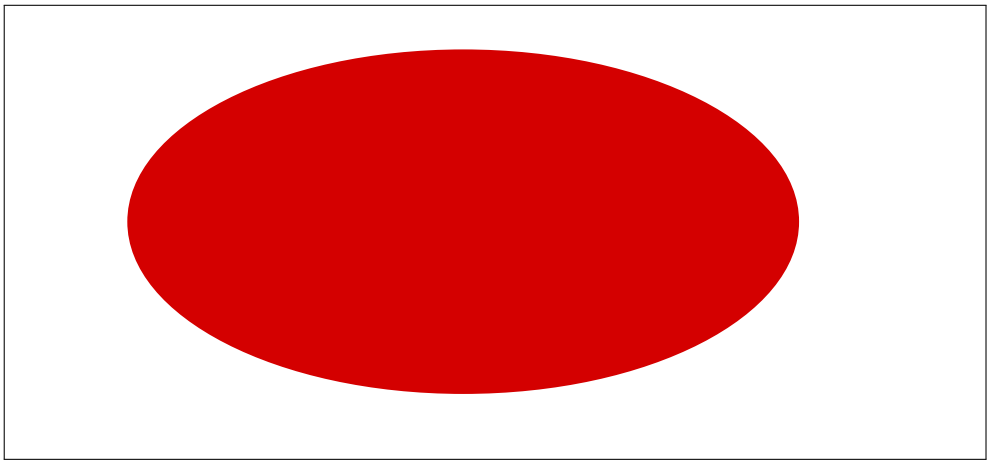
1.1.1.1 Zero day útoky

Zero day útoky lze volně přeložit jako útoky nultého dne. Jde o taková bezpečnostní rizika, která využívají zranitelností nějakého systému, nebo jeho části, která nebyla zveřejněna [5].

Integrační platforma je složena z mnoha modulů, který tvoří její celek. S rostoucím množstvím softwaru a použitých knihoven jsou šance na skrytou chybu velmi vysoké.

1.1.1.2 Odepření služby

Útoky, jejichž cílem je znepřístupnit konkretní jednu či sadu služeb rozhodně patří mezi narušení kybernetické bezpečnosti. Útok spadající do kategorie Dos **[[Definovat Dos]]** je DDos, jde o útok, kde hromadně více zařízení zahltí požadavky konkrétní služby a vyřadí je tím z provozu [6].



Obrázek 1.1: Grafické znázornění DDos útoku.

Problémem nemusí být když se takové riziko projevuje přímo. Pokud například důležitý partner je pod takovým útokem, ohrožuje to business procesy i na naší straně. Je třeba zavčas podobný incident detekovat a na základě dalších informací se rozhodnout jak vzniklou situaci řešit.

[[Chtěl bych zde konkrétně popsat DDos na O2 z lonskeho roku]]

1.1.1.3 Nevalidní dotazy

Riziko, které je třeba hlídat, ale nutně způsobené za účelem poškodit někoho nebo něco. Integrační platforma musí zpracovávat různé požadavky od konzumentů a ty zpravidla přeposílat poskytovatelům služeb. Mohou nastat situace, kdy konzument posílá nevalidní požadavek.

V případě, že požadavky jsou v xml formě, lze definovat přesné znění zprávy. V případě, že je její validita narušena dochází k nekompatibilnímu dotazu na stranu poskytovatele. To může vyústit v nevyřízené a odmítnuté požadavky. Z této situace můhou vzniknout nechtěné útoky *odepření služeb*. V případě, že požadavek nebude zamítnut, možným důsledkem může být zisk nevalidních dat nebo dokonce i dat, která se ke konzumentovi neměla dostat.

Tyto popřípadě i jiné neočekávané situace je třeba včas odhalit a řešit.

1.1.2 Ostatní

Mnoho bezpečnostních rizik může být neznámých. Nelze vše zaškatulkovat do obecných kategorií. Rizikem mohou být i služby, které po autorizaci nabízí různé možnosti získání dat, například pomocí selectů z databáze.

1.1.3 Zjišťování bezpečnostních rizik

Rozpoznávat bezpečnostní rizika a je při jejich různorodosti složitý úkol.

1.1.3.1 Posouzení kódu

Posouzení kódu (anglicky code review) je metoda kontroly kvality kódu. Programátor, který napsal nějaká kus komponenty dá svou část programu na kontrolu druhé osobě. Není přímo nutné, aby kontrolující osoba byla nadřízený. Kontrolující programátor hlídá kvalitu kódu a zároveň prověřuje jeho funkčnost. Tato metoda vede ke zkvalitnění dodávaného softwaru, ale je finančně i časově náročná. [[**cite**]]

Zvýšením kontroly se i zvyšuje šance na detekci bezpečnostního rizika. Jako velmi efektivní metodu ji uvádí zdroj [7].

1.1.3.2 Analýza z pohledu uživatele

Ve snaze celý proces detekce bezpečnostních rizik zautomatizovat se používá i metoda založená na pohledu uživatele aplikace [7]. Aplikace má své uživatele, proto je možné vytvářet scénáře, kdy se uživatel pokouší najít v aplikaci nějaký nedostatek. Takový nedostatek může následně produkovat bezpečnostní riziko. Automatizace těchto scénářů je zpravidla snadno proveditelná a často patří do klasických testovacích scénářů.

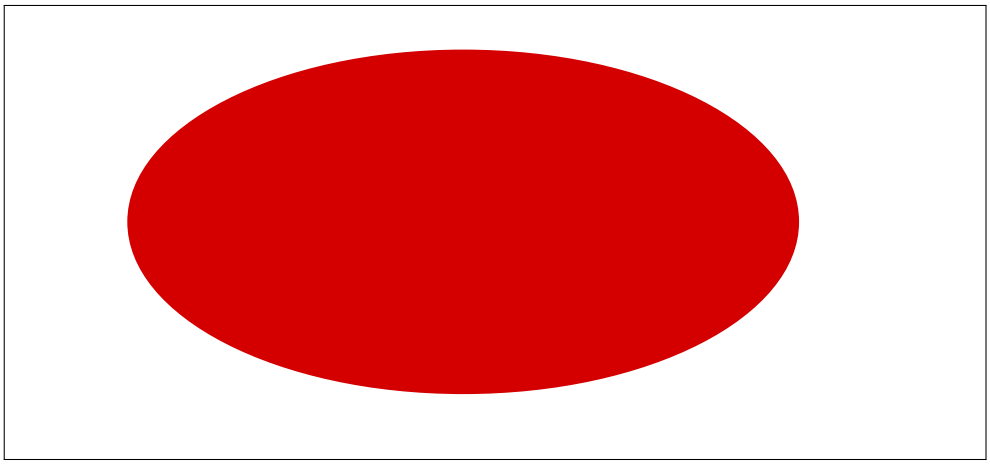
Testují se například SQL injection, přístup do neopravněných míst a další.

1.1.4 Analýza na straně serveru

Předchozí způsoby se snažili zabránit výskytu rizik. Pro zachování kybernetické bezpečnosti je vhodné sledovat i aktuální situaci v aplikaci. Vzhledem k rozshahu moderních aplikací je takřka vyloučené, že bychom dokázali rizikům předejít. Sledování aktuálního stavu a provozu můžeme být schopni detekovat podezřelé činnosti. Na základě toho pak dokážeme zjistit bezpečnostní riziko.

I zde lze využit automatizaci. Například ve zdroji [8] je uveden systém na rozpoznání vzorů zpráv. Koncept založený na znalostním inženýrství může pomoci detekovat i využití zranitelnosti Zero-day**[[Odkaz na kapitolu]]** [9].

Podobný princip využívají i systémy SIEM (security information and event management). SIEM se zabývá bezpečnostním management. Jeho myšlenka je taková, že rozsáhlé aplikace mají své zdroje informací (logy) na různých místem, ale je zapotřebí k nim přistupovat z jednoho místa. Z tohoto místa se provádí následná analýza kompletně všech zdrojů a vyhodnocují se bezpečnostní rizika [10]. Pomocí tohoto vyhodnocení pak může specializovaný personál reagovat na vzniklou situaci.



Obrázek 1.2: Grafické znázornění SIEM.

1.1.5 Zkušenosti z Unify

[[Popsat jaké zkušenosti mám z unify]] [[Chyba může být nevalidní request, Snaha o umělý request ...]]

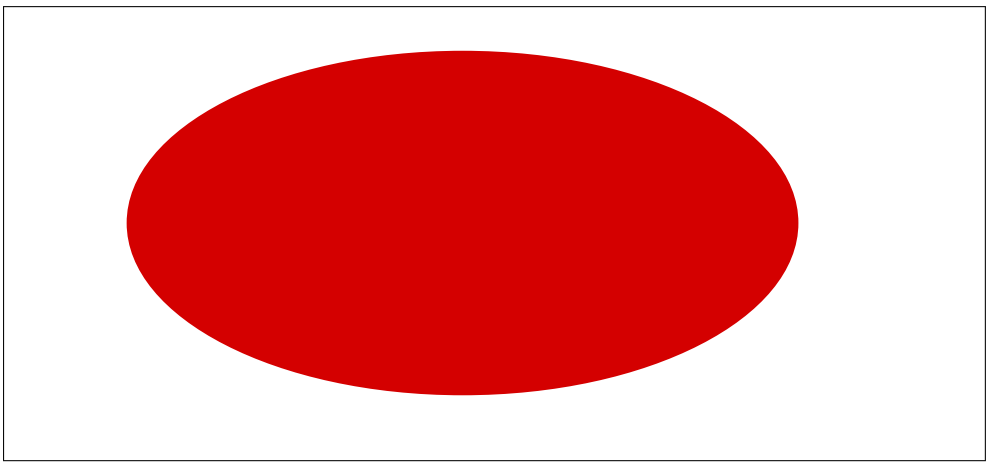
1.2 Platforma Unify

[[Celá tato kapitola se mi nepovedla, je to spíše náčrt.]] Cílem práce je prověřit jeji funkčnost na platformě Unify [11].

1.2.1 Integrační platforma

Unify je produkt, který slouží jako integrační platforma. Integrace slouží k propojení různorodých systémů a služeb.

Integrace se zpravidla zkládají ze sběrnice [12]. Sběrnice, která propojuje konzumující aplikaci a poskytující aplikace se nazývá *ESB* (enterprise service bus). ESB je stavěno na architektuře orientované na služby [13].



Obrázek 1.3: Enterprise Service Bus.

Konzumující aplikací rozumíme takovou aplikaci, která posílá dotaz na konkrétní službu. Poskytující aplikace je taková aplikace, která poskytuje rozhraní, jehož výstupem jsou data pro různé konzumující aplikace. V reálné aplikaci je možné poskytovat interface, který například spustí nějaký proces.

ESB je využíváno interně v rámci jedné firmy nebo logické struktury. Pokud je žádané, aby některé služby byly konzumovány aplikací takzvaně odjinud používá se sběrnice B2B (business to business) [14].

B2B poskytuje rozhraní a zpravidla následně samo je konzumentem ESB.

1.2.2 O Unify

Unify je integrační platforma, která je orientovaná na služby [11]. **[[Sklada se z B2B, ESB, ETL, SFE + jednoduché popisy]]**

Unify je postavena na open source technologiích [11]. Základním stavebním kamenem je ESB, které umožňuje propojit různorodé systémy, služby.

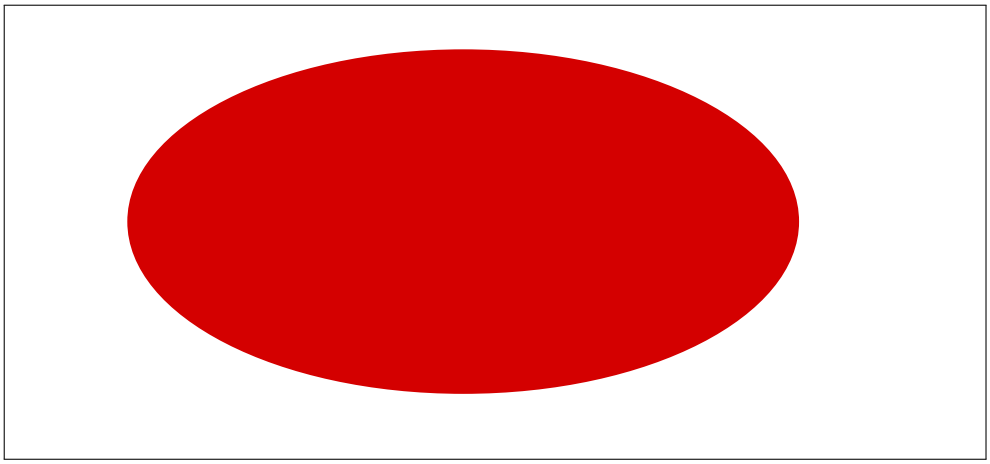
Pro pro připojení externích služeb a aplikací využívá B2B. Technologicky je B2B velmi podobné ESB, ale zpravidla je vhodné ho více zabezpečit, nebo používat validace na požadavky a vyhnout se tím takovým zprávám, jenž neodpovídají ani předem danému předpisu.

Další z mnoha komponent jsou ETL (extract transform load), pomocí kterého dochází k zisku nějakých dat jejich transformaci a nahraní na jíné místo.

Nebo SFE (secure file exchange) jehož význam je přenos souborů, hlavně mezi zónami Z2 a Z4. **[[nemohu si ted vybavit jak se odborně nazývají]]**.

Stavební kámen integrační platformy Unify je Jboss AS 7, který zcela podporuje standard Javy EE 6 [15].

[[Používá se hlavně Java]] [[Podporuje SOAP, REST, HTTP, DB procedury]] [[Rychlé popsání logování]] [[Stávající řešení ochrany - info v GUI, journaling, logy a SL2]]



Obrázek 1.4: High Level Desing architektura Unify.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

[[Zkratky B2B,ESB]]

1.3 Text mining

V logách integrační platformy je veškerá komunikace v textovém formátu. Kromě zpráv ve formátu xml jsou zde i zprávy ve formátu json, popřípadě výpis chyby, pokud nějaká nastala. Protože v prací řeším analýzu takových logů, tak následující kapitola přiblíží těžení znalostí z textu.

Text mining má velký význam, neboť 80% informací uložených v počítačích jsou textové formy [16].

Text mining je metoda, která těží informace z textu [17]. Přesto, že text mining spadá do kategorie data miningových metod, rozdíl je v tom, že nepracuje s číselnými a většinou ani nominálními hodnotami. Datamining dokáže detekovat skryté informace ze vstupních dat. Text mining informace nemá většinou ve svých datech nijak skryté. Při zpracování textu jde o automatizaci procesu, tak jak by ho zvládl člověk, počítačem. [17].

Dle zdroje [18] lze probletiku text miningu rozdělit následovně:

1.3.1 Získání informací

Systémy získání informací identifikují v kolekci souborů takové, jaké jsou vhodné pro vstupní požadavek. Je například o problematiku vyhledávácích nástrojů. Princip je takový, že hledáme-li konkrétní dotaz, jsou vybrány z kolekce všech souborů jen ty, která se dotazu týkají. To se rozhoduje například slovy použitými v dotazu a jejich synonimy.

1.3.2 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka je jeden z nejtěžších problémů text miningu. V této disciplíně se řeší převod textu do mluveného slova, rozpoznání řeči a podoně. Princip je naučit stroje rozumět přirozenému jazyku. To pomahá například při anotaci souborů.

1.3.3 Data mining

Data mining je proces, který hledá skryté vzory v datech. V použití s text miningem je využití pro prezentaci různých výsledků koncovému uživateli.

1.3.4 Extrakce informací

Extrakce informací je proces, kde jsou vytažena data ze vstupu a vložena do logických struktur. **[[víCe]]**[19]

[[Možná to nad bude lepší v odrážkách než podkapitolách.]]

1.3.5 Transformace textových dokumentů

Aby bylo možné jednotlivé textové dokumenty klasifikovat nebo shlukovat je třeba je převést na číslný vektor. V této části kapitoly se budu zabývat možnostmi takového převodu.

1.3.5.1 TF-IDF

TF-IDF je zkratka aglického názvu *Term Frequency Inverse Document Frequency*. Snažší překlad bude, pokud název rozdělíme na Term Frequency, což je v pře-

kladu četnost slova a Inverse document frequency, jenž znamená převrácena četnost dokumentu [20].

Četnost slova vyjadříme následovně: máme slovo w a množinu dokumentů D, skládající se z dokumentů $d_1, d_2 \dots d_3 \in D$. Potom četnost slova $TF(w, d_x)$ vyjadřuje kolikrát se slovo w vyskytlo v dokumentu d_x.

Převrácená četnost dokumentu vystihuje jak podstatné slovo w je. Značíme ji jako

$$IDF(w, D) = \log \frac{|D|}{|D_w \subseteq D|}$$

kde $|D|$ vyjadřuje velikost množiny všech dokumentů a D_w množina všech dokumentů, ve kterých se slovo w. $|D_w|$ pak značí velikost takové množiny [20].

Pro slovo w v dokumentu d vypočteme TF-IDF následovně:

$$TF - IDF(w, d, D) = TF(w, d) * IDF(w, D)$$

Na základě uvedeného vzorce jsme schopni reprezentovat textový dokument pomocí vektoru. Vektor takové dokumentu bude vždy nezáporný a bez úprav bude mít tolik dimenzí, kolik má jednoznačných slov v sobě.

1.3.5.2 Snížení dimenzionality

Při práci s textovými dokumenty je třeba snížit jejich dimenzionalitu. Jak je uvedeno v sekci 1.3.5.1 bez úprav dimenzioality TF-IDF pomůže vytvořit vektor o velikosti rovné počtu unikátních slov. Do takové seznamu slov by ovšem v ten moment mohly zapadat slova stejného významu, jen například jinak skloňována. Dalé bude-li pro jednotlivá slova v dokumentu oddělovač mezera, mohou vzniknout jako dvě unikátní slova například „slovo“ a „slovo,“.

Před snížením dimenzionality je vhodné zbavit se veškeré interpunkce. Kromě interpunkce je vhodné i text převést kompletně na velká nebo malá písmena.

Pro snížení dimenzionality se používá odstranění takzvaných stop-slov. Jde o slova taková, která nemají velký význam. Přiklad pro to mohou být předložky a spojky. Pro mnoho jazyků jsou k dispozici slovníky s takovými slovy. Dle konkrétní charakteristiky textového dokumentu může být vhodné nadefinovat si svá stop-slova.

Dalším nástrojem pro redukci počtu dimenzí je stemitizace. Cílem stemitizace je redukce slov tím, že jsou převedena na kořen slova [21]. Je třeba si uvědomit, že tímto krokem je možné ztratit původní význam slova a je tedy třeba promyslet, jestli na konkrétních datech stemitizaci využít.

1.4 Těžení Asociačních pravidel

Jednou z prvních myšlenek jak dosáhnout cíle bylo využití metody těžení asociačních pravidel.

1.4.1 Definice

Cílem těžení asociačních pravidel je najít zajímavé korelace či časté vzorce v množině dat uložených v relační databázi nebo jiných uložiští [22]. Jako ukázka praktického využití lze představit využití v obchodních řetězcích. Obsah nákupu, který si zákazník zakoupil, je uložen v rámci jedné transakce. Algoritmus vyhledá zajímavé vzorce a pravidla mezi položkami v transakcích. Výsledkem je možnost predikovat co si zákazník zakoupí. Jako příklad uvedeme zákazníka, který si koupil housku a salát. Jako predikce další položky lze očekávat hamburgerové maso. Tato znalost pak lze využít k lepšímu přemístění zboží v regálech či k osobnějším nabídkám zboží v reklamních tiskovinách.

Těžení asociačních pravidel lze dle zdroje [23] popsat následovně. Nechť $I = I_1, I_2, \ldots, I_m$ je množina binárních atributů, kterou budeme nazývat položka. Nechť T je databáze transakcí. Každá transakce t je reprezentována binárním vektorem, kde platí, že $t[k] = 1$ pokud t zakoupila položku I_k. V opačném případě $t[k] = 0$. Nechť X je množina některých položek z I. Říkáme, že transakce t splňuje X pokud platí pro všechny položky I_k v X pravidlo $t[k] = 1$.

Asociačním pravidlem rozumíme implikaci $X \implies I_j$, kde X je množina některých položek v množině I a $I_j \in I$ ale zároveň $I_j \notin X$.

Pravidlo $A \implies B$ platí s podporou S v transakci T, kde $S(A \implies B) = P(A \cup B)$. Pravidlo $A \implies B$ má v transakci T spolehlivost C, kde $C(A \implies B) = P(B|A)$.

Cílem těžení je zjistit vztah mezi různými položkami tak, že přítomnost některých položek v transakci implikuje přítomnost jiné položky.

1.4.2 Myšlenka pro využití

Myšlenka využití při analýze logu spočívala v tom, že by každá jednotlivá zpráva byla počítána jako transakce. Jednotlivá slova (po předzpracování) by tvořila položky. Pak by bylo možné predikovat, že výskyt některých termů bude znamenat například to, že dojde k odmítnutí nevalidního požadavku. Tyto získané informace by ale neměli nikterak velkou hodnotu. O odmítnutí požadavku se v platformě dozvíme zpravidla v rámci milisekund v synchronní odpovědi.

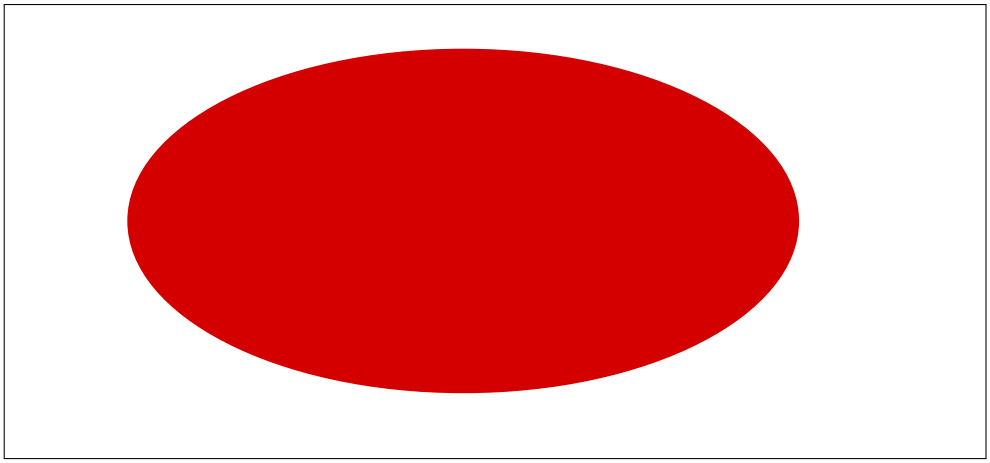
Velmi pravděpodobně bezpečnostní rizika budou přibývat. Tato možnost by šla využít pro již existující nebo alespoň několikrát se vyskytující případy.

Z těchto důvodů jsem tuto myšlenky zavrhl, stím, že bude lepší vyzkoušet takové metody, které budou mít šanci rozpoznat špatný požadavek, i když ho uvidí poprvé. Tyto metody jsou rozepsány v následujících sekcích.

1.5 Shluková analýza

Jako jednu z možností vyřešení detekce bezpečnostních rizik na základě dat z logů jsem si vybral shlukovou analýzu (dle anglického názvu také nazýván clustering). Clustering využívá znalosti ze vstupních dat k tomu, aby dokázal vstupní požadavky rozdělit do shluků. Všechny informace jsou do těchto shluků přiřazovány na základě podobnosti. Podobnost lze definovat podle charakteristiky dat a požadovém účelu shlukování. Jednotlivé shluky pak tvoří užitečné a logické skupinky .

Všechny objekty ve shluku si jsou navzájem podobné a zároveň se nepodobají objektům v jiném shluku [24]. Na obrázku 1.5 je ukázka několika příkladů, kdy je na stejná data použita shluková analýza s jiným počtem shluků.



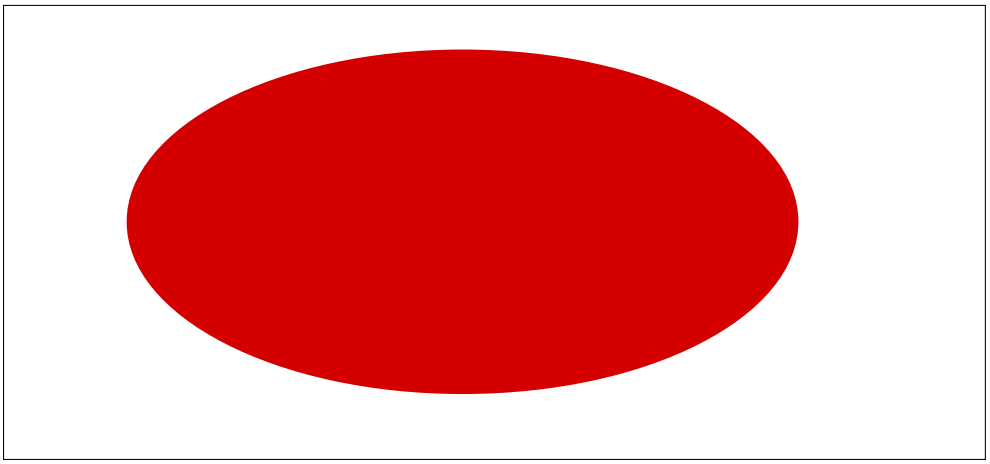
Obrázek 1.5: Ukázka shlukové analýzy s různym počtem shluků.

[[Napsat nějaký rozcestník o tom, že dále uvedu tu a tu metodu clusteringu]]

1.5.1 K-means

K-means a K-medoids jsou metody shlukové analýzy. Tyto metody jsou založené na principu centroidů. Centroidem je vyjádřen pomyslný střed každého shluku. V případě K-means jde o střed shluku nezávisle na tom, je-li tento bod i objektem vstupních dat, nebo ne. K-medoids jako střed shluku určí vždy nejvhodnějšího zástupce ze vstupních dat. Graficky je tento rozdíl zobrazen na obrázku. Z obrázku vyplýv, že shluk v K-means může mít střed mimo data, kdežto K-medoids má možnost středem shluku určit pouze nejvhodnější objekt ze vstupních dat 1.6.

Postup algoritmu pro K-means je nejdříve vybrat K bodů jako centroidy. Tyto centroidy lze vybrat například náhodně, nebo prvních N. Volbou prvních centroidů se například zabývá zdroj [25]. Po vybrání středů se všechny



Obrázek 1.6: Ukázka shluku a jeho centroidu pomocí metody K-means a K-medoids.

data přiřadí k nejvhodnějšímu shluku. Výběr takové shluku často ovlivňuje například vzdálenost bodu k centroidu. Následuje přepočet nových středů. Přiřazení do shluků a výpočet nových centroidů se opakuje, dokud shluky mění. Po té považujeme algoritmus za hotový a data rozdělená do skupin.

[[Přidat algoritmus.]]

Jedním z kroků algoritmu je přiřadit vstupním objektům správný shluk. K tomuto důvodu je nutné definovat funkci pro měření vzdáleností mezi jednotlivými objekty [24]. Požadavky na takovou funkcni jsou, aby byla jednoduchá, protože je velmi často volaná. Pokud data jsou v eukleidovském prostoru používají se například následující funkce na měření vzdálenosti:

1.5.1.1 Eukleidova vzdálenost

Eukleidova vzdálenost určuje vzdálenost mezi dvěma body v eukleidově prostoru. Výpočet euiklediovy vzdálenosti mezi body $p = (p_1, p_2, \ldots, p_n)$ a $q = (q_1, q_2, \ldots, q_n)$ je

$$d_e(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

1.5.2 Cosinova vzdálenost

[[Dodělat, je dobrá na různě dlouhé vektory, prostě jen vypočte úhel]]

1.5.2.1 Manhattanská vzdálenost

Dalším příkladem měření vzdáleností bodů $p = (p_1, p_2, \ldots, p_n)$ a $q = (q_1, q_2, \ldots, q_n)$ v eukleidově prostoru je manhattanská vzdálenost. Její výpočet je dle vzorce:

$$d_m(p, q) = \sum_{i=1}^{n} |p_i - q_i|$$

[[Ostatní vzdálenosti]] [[Zmínit, že to je bez učení]] [[Závislé na trénovacích datech]]

1.5.3 Důvod využití

Informace, které integrační platformou prochází jsou velmi různorodé. Požadavky je možné rozdělovat do různých skupin jako například:

- skupiny dle druhu služby (SOAP, REST, databázová ...)
- skupiny dle služeb
- skupiny dle toho, zdali je zpráva požadavek nebo odpověď

Jako jedno z možných rozdělení je možné na požadavky v pořádku, podezřelé požadavky a chybné požadavky. Cílem je najít takovou konfiguraci, která by podobné rozdělení dokázala najít. Tedy rozeznat požadavky, které jsou v pořádku (běžná komunikace na integrační platformě.) od těch, jenž jsou podezřelé (ne příliš častý vzor požadavku, ...), či zaručeně chybné (chyba protokolu, validace, ...).

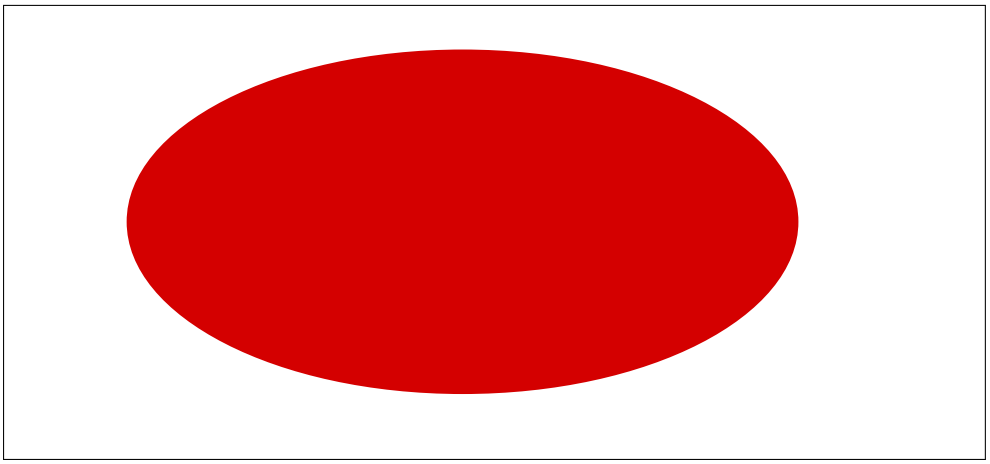
[[Vysvětlit někde princip komunikace request -> response, popřípadě asynchronní, jms atd.]]

[[Proč použít clustering]] [[Druhy clusteringu]]

1.6 Detekce anomálie

Většina požadavků, která projde skrz integrační platformu jsou v pořádku. Chyba či podezřelá zpráva se vyskytují zpravidla minimálně. Proto jsem jako další metodu zvolil detekci anomálie. Detekce anomálie je proces, při kterém se vyhledávají taková data, která se od ostatních výrazněji liší. Detekování odlišností se hojně využívá právě zajišťování bezpečnosti [26]. Příklad anomálie je na obrázku 1.7.

Princip detekce anomálií je definovat běžná data / chování. Následně veškerá data / chování, která těmto požadavků neodpovídají označit za anomálie 1.7.



Obrázek 1.7: Ukázka anomálie v datech.

1.6.1 Metody detekce anomálie

Detekce anomálií lze rozdělit na tři základní druhy [27]:

- statistické rozdělení
- metody založené na vzdálenosti
- metody založené na hustotě

Statitistické metody očekávají, že data mají průběh nějakého rozdělení. Tomu ja následně přizpůsoben přístup zjišťování anomálií. Zpravidla proto tyto metody nejsou v praxi využívané [27].

U metod založených na vzdálenosti se výpočíta vzdálenost konkrétního data a jeho sousedů. Je-li nalezena vzdálenost větší než předem daný práh, je cílový prvek chápán jako anomálie.

Metody založené na hustotě vypočítávají takzvaný LOF (Local Outlier Factor). LOF je hodnota, která u každého prvku určuje míru, jako moc je anomálií [28].

Základ výpočtu LOF je, výpočet vzdálenosti ke k nejbližším sousedům. Tato vzdálenost je použita pro odhad hustoty. Následné porovnání hustoty elementu s hustotami svých k sousedů rozhoduje o tom, bude-li označen za anomálii.

Definice LOF dle [28] je popsána pomocí několika dílčích definicích:

1.6.1.1 K-vzdálenost

Nechť pro libovolné celé číslo k, kde $k > 0$ je určena *k-vzdálenost* objektu p ($k - dist(p)$) jako vzdálenost $d(p, o)$, mezi objektem p a objektem $o \in D$, kde platí:

- pro nejméně k objektů $o' \in D \backslash p$ platí, že $d(p, o') \leq d(p, o)$ a zároveň
- pro alespoň $k - 1$ objektů $o' \in D \backslash p$ platí, že $d(p, o') < d(p, o)$

Množinu k nejbližších sousedů definujeme jako $N_k(p)$

1.6.1.2 Dosažitelná vzdálenost

Nechť k je přirozené číslo. Dosažitelná vzdálenost objektu p s ohledem na objekt o je definována následovně:

$$reach - dist_k(p, o) = max(k - distance(o), d(p, o))$$

1.6.1.3 Hustota dosažitelnosti

Hustota dosažitelnosti objektu p je definována jako

$$lrd_{N_k}(p) = \frac{1}{\frac{\sum_{o \in N_k(p)} reach - dist_k(p,o}{|N_k(p)|}}$$

Hustota dosažitelnosti lze popsat jako inverze průměru dosažitelné vzdálenosti k nejbližších sousedů p.

1.6.1.4 Činitel anomálie objektu p (LOF

LOF objektu p je definován jako

$$LOF_{N_k}(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_{N_k}(o)}{lrd_{N_k}(p)}}{|N_k(p)|}$$

1.6.2 Analýza hlavních komponent

Analýza hlavních komponent, častěji označována jako PCA z anglického překladu *Principal Component Analysis*. **[[Zkratka PCA]]** PCA provede analýzu vstupních dat. Tyto data se skládají z mnoha proměnných, které na sobě mohou být závislé. Účelem je vybrat z těchto proměnných důležitou informaci tu reprezentovat jako nové proměnné zvané hlavní komponenty [29]. Tyto hlavní komponenty jsou lineární kombinací originálních hodnot.

[[Být alespoň na straně 20]]

Kapitola 2

Návrh řešení

[[Nic moc tento odstavec]] V této kapitole se zabývám principy, technologiemi algoritmy, které jsem se rozhodl použít, k tomu abych splnil cíle této práce. Tedy vytvoření aplikace, jenž umožní sledovat bezpečností rizika v reálném čase.

2.1 Architektura aplikace

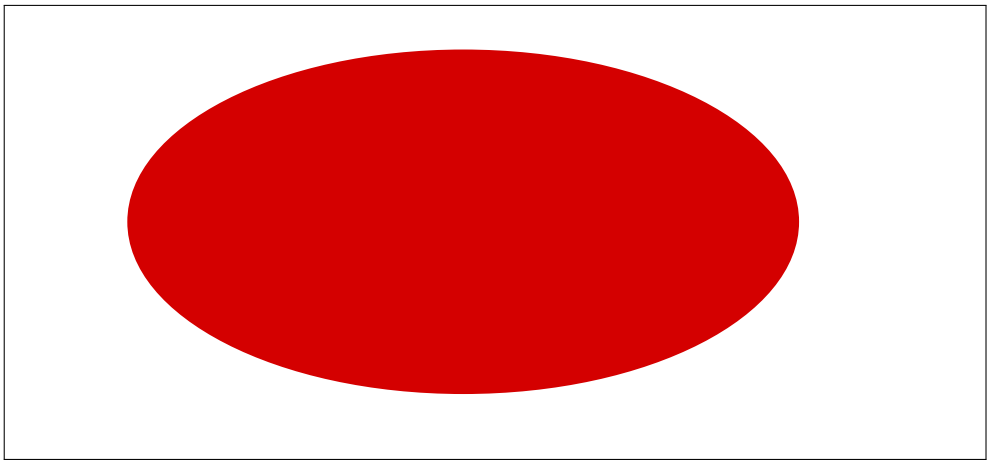
[[Pozor na duplici s odstavcem v úrovni nad]] [[Lze rozdělit na podsekce]] Požadavkem na aplikaci je, aby požadavky zpracovávala a predikovala v reálném časem. Proto je princip položen na čtení požadavků proudících přes platformu. Jejich předzpracování a odeslání do cloudového řešení Microsoft Azure (Více v sekci 2.2). Po přijetí odpovědi je výsledek uložen do databáze. Pro vizualizaci dat slouží REST API [30], které vypisuje předem definované informace ve formátu pro ideální zobrazení v Google Charts [31].

Pro snažší představu o architektuře aplikace poslouží obrázek 2.1, na kterém je vidět High Level Desing [32].

Základem celé aplikace je neohrozit stávající integrační platformu. Na základě toho jsem se rozhodl, že informace o proběhlé komunikaci získám pomocí čtení logovacích souborů. Pomocí čtení přírůstků k jednotlivým auditovým logům získám jednotlivé požadavky a pro Unify to nepředstavuje žádnou zátěž.

Dalším stavebním kamenem je použitý aplikační server Jboss (více v kapitole 2.3). Na serveru je celá aplikace. Dochází zde k předzpracování zpráv, jejich odslání do Microsoft Azure (2.2) a také k ukládání do DB.

Jak jsem již zmiňoval provoz z platformy je po zpracování odesílán do MS Azure, zde jsou definovány jednotlivé algoritmy, jejichž výsledky jsou vracený zpět do aplikačního serveru. Azure jsem se rozhodl používát, protože umožňuje rozložení výkonu na server Microsoftu a protože využití služeb v cloudu se stává stále oblíbenějším. Díky spolupráci s Microsoftem je možné i získat, popřípadě zakoupit, instanci Azure do vlastní sítě.



Obrázek 2.1: High Level Desing architektura aplikace.

Získané výsledky jsou zpracovány a uloženy do NoSQL databáze MongoDB (více v kapitole 2.5).

Aby výsledky nebyly jen hodnoty uložené v databází, je použité REST API, přes které lze výsledky sdílet. API je navržené tak, aby v případě použití Google Charts nevznikly žádné potíže. Google Charts se u cílového zákazníka pužívají již nyní například na zobrazení stavu objednávek. Proto jejich se jejich použití jeví jako další logický krok. Nicméně, není problém stejné api použít pro svoji libovolnou aplikaci, která hodnoty použije buď pro zobrazování přehledů, nebo jako jeden z dalších vstupů například do různých systémů SIEM.

2.2 Microsoft Azure

Na integrační platformě Unify [11] je předpokládaný provoz 20 požadavků za vteřinu. Vzhledem k takto silnému provozu bude potřeba i přiměřeně velký výpočetní výkon.

Spolupráce se společností Miscrosof [33] mi umožnola jako řešení vyzkoušet její cloudové služby Microsfot Azure [34].

Microsoft Azure je sada integrovaných cloudových služeb. Azure nabízí cloudová řešení pro mnoho činností. Motivací k použití této služby k detekci bezpečnostích rizik je nástroj Microsoft Azure Machine Learning Studio [35].

Microsoft Azure Machine Learning Studio je plně cloudová služba, která umožňuje vytváření prediktivních modelů pro strojové učení [35]. Výhodou studia je to, že veškerý výkon je rozprostřen vevnitř cloudu. Díky grafickému rozhraní je snadné vytvořit učící model, který je následně převeden do modelu prediktivního.

Aby měl prediktivní model smysl, je třeba mu poskytovat nějaká data, u kterých je predikce využita. K tomu se využívají webové služby. Prediktivní model se vystaví na specifické URL adrese. Zde je pak očekáván na vstupu konkrétní formát JSONu a služba vrací předem definovanou odpoveď se správnými parametry.

Výhodou využití cloudu je přenesení výpočetní zatěže mimo společnost. Naopak rizikem je problém s konektivitou, který může vytvořit výpadek služby a nebude tedy možné po tuto dobu predikovat rizika. Jednou z možností, jak řešit takové riziko je nechat přenést instanci MS Azure do své sítě.

[[Více rozepsat rozdíl mezi učícím a prediktivním modelem]]

2.3 JBoss

Platforma Unify je postavená na aplikačním server JBoss AS 7 [11]. Z toho důvodu je třeba aby aplikace byla zcela kompatibilní.

[[Nají nějaké zdroje kde je popsáno o co vlastně jde]] Jboss AS je aplikační server pro Javu EE[36].

[[Popsat proč jboss]] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.4 Logování Unify

Platforma Unify loguje veškerý průběh do souborových audit logů. Protože celá integrace je založena na Javě, je využit logovací framework Log4j [37]. Knihovna Log4j umožňuje nastavit si pattern logování [38]. Na Unify je použit pattern **[[Přidat pattern Log4j]]**.

Každý požadavek, který na platformu příde je zalogován do audit.logu. Zpráva je vždy na jedné řádce a zároveň na jedné řádce je povolena pouze jedna.

[[Ukázka logu]]

[[Popsat ukázku logu]]

Vzhledem k tomuto principu jsem se rozhodl přistoupit k tomu, že se vybrané logovací soubory budou kontinuálně číst, kde se bude ke každému

řádku přistupovat jako k samostatné zprávě, která bude následně zpracována dál.

Díky této volbě nebude nutné nijak zasahovat do integrační platformy Unify a minimalizuje se tím riziko, jakéhokoliv nebezpečí ze strany mojí aplikace.

Unify využívá pro některé služby logování do Oracle databáze. Ale vzhledem k tomu, že jde pouze o několik málo určených služeb, připojení na databázy by tak kromě případných komplikací ani nepřineslo žádaný účinek.

2.5 Ukládání dat

[[Porovnání SQL/NoSQL]] Rozhodl jsem se pro ukládání dat využít NoSQL databázi. Protože aplikace Unify momentálně ukládá veškerý svůj provoz do souborů (vyjímečně jsou některé konkrétní služby journalovány do DB), bude databáze využita i pro ukládání veškeré komunikace. To zpřístupní do budoucna snažší operování s jednotlivými zprávami. Popřípadě snažší zpětnou analýzu. **[[Schéma DB]]**

Protože se bude ukládat veškerá komunikace, rozhodl jsem se v databázi mít uvedené následující informace:

[[Byt je to na pohled jasne, tak popsat co ktery param. je]]

- ObjectId - automaticky generováno z MongoDB
- timestamp- časové razítko uložení záznamu do DB
- original-message - nezměněná zpráva
- normalized-message - zpráva po normalizaci
- platformId - jedinečný identifikátor platformy
- assignment - skupina, kterou azure vyhodnotil pro zprávu jako správnou

2.6 Předzpracování dat

[[Dopsat čištění dat]]

Pro snažší prácí s informacemi z logů jsem se rozhodl pro normalizaci jednotlivých požadavků. Normalizace je jeden z požadavků při zpracovávání textu [39].

V kapitole 2.4 jsou vidět informace, které se kromě samotné zprávy logují. V každé zprávě se objevuje takzvaná integrační hlavička. V té jsou základní údaje, jako čas odeslání, jednoznačné identifikátory, zdrojové a cilové systémy. Položka jako je timestamp bude zpravidla pro každý požadavek jiná, stejně na tom budou jednoznačné identifikátory. Z tohoto důvodu jsem se rozhodl zvolit jejich nahrazení.

Při normalizaci dat jsem se podobně jako ve zdroji [40] rozhodl použít následovně:

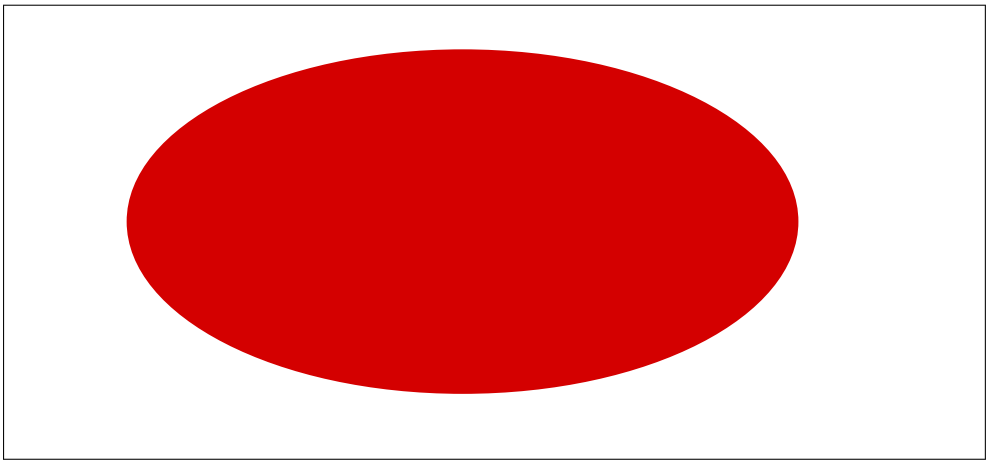
- Nahrazení všech čísel - Pomocí speciálního symbolu nahradím všechny výskyty čísel.
- Velikost písmem - Všechna písmena jsou z velkých znaků převedena na znaky malé.
- Odstranění speciálních znaků - Veškeré znaky jako jsou čárka, tečka ...jsou odstraněny
- Odstranění xml tagů - Rozhodl jsem pro zpracovávat jen obsah zpráv bez xml tagů.

Nahrazení čísel se mi jeví jako logický krok. V jednotlivejch požadavcích jsou čísla například výsledky různých měření na sítí nebo právě čas v časovém razítku. Pro další využití považuji za podstatné vědět, že se v daném místě vystkytovalo číslo, než že to bylo nějaké konkrétní číslo.

Převod písmen na malá zajistí, aby slova, lišící se právě jen ve velikosti nějakých písmen byla vyhodnocena jako stejná.

Všechna komunikace na platformě je převedena do xml (není-li již od počátku vedena v xml). Protože téměř u všech zpráv stejného druhu se používají ty samé xml tagy, nebudou pro další zpracování podstatné a budou zcela odstraněny. Algoritmus bude dále pracovat jen s reálným obsahem zprávy.

Na obrázku 2.2 je ukázka originálu zprávy a její normalizované alternativy.



Obrázek 2.2: Original and normalized message

2.7 Vytvoření vektoru

V okamžik, kdy máme předzpracovaná, znormalizovaná textová data je nutné najít vhodný způsob pro jejich převod do numerické podoby. To umožní snažší zpracování jak v případě clusteringu, tak i v případě detekce outlinerů.

Cílem tedy je vytvořit vekotr, který bude dostatečně jednotlivé zprávy reprezentovat.

2.7.1 TF-IDF algoritmus

Při clusteringu dokumentů lze využívat algoritmus TF-IDF (term frequency - inverse document frequency) [41].

[[Trošku lépe pořešit]]

2.7.1.1 Frekvence slova

Ten funguje na principu, že se spočíta frekvence daného slova w v dokumentu d, označujeme $TF(w, d)$. Vypočteme se tak, že se spočítá suma výskytů slova w v dokumentu d. Výšší číslo znamená častější výskyt a tedy o to více w charakterizuje d.

2.7.1.2 Frekvence dokumentu

Frekvence dokumentu pro slovo w DF(w) je počet dokumentů, ve kterých se slovo w nachází.

2.7.1.3 Inverzní frekvence dokumentu

IDF neboli inverzní frekvence dokumentu je daná následující formulí [42]:

$$IDF(w) = \log \frac{|D|}{DF(w)}$$

Kde $|D|$ je počet souborů.

2.7.1.4 TF-IDF

Samotný vzorec na výpočet TF-IDF je [41]:

$$TFIDF(w, d) = TF(w, d) * IDF(w)$$

2.7.1.5 Použití

Pro své účeli budu předpokládat, že jednotlivé zprávy jsou soubory a slova budou mezerou oddělený obsah zprávy.

Protože slov může být velké množství, rozhodl jsem se najít nějakou hranici, například takovou, že do výsledného vektoru zanesu TF-IDF pro taková slova, která se vyskytují nejvíce v 95% zpráv, ale minimálně v 10%.

2.7.2 Forma vektoru

V sekci 2.7.1 jsem navrhl, jak textová data převést do vektoru. Tím je zaručené, že bodou-li se data přenášet přes internet do Microsoft Azure, budou anonymizována. Z vektoru nedokážeme zpětně zprávu vyčíst.

I když z Azure dostáváme synchronně odpověď zpět, a je tedy jasné, ke které zpráve dostávám výsledek, rozhodl jsem se odesílat i jednoznačný identifikátor platformy. To vede k tomu, že pro znalého člověka lze jednotlivé požadavky sledovat i uvnitř MS Azure. Identifikátor sám o sobě vypovídající hodnotu žádnou nemá, ale máme-li k dispozici původní zprávu, jsem ji schopni dohledat.

[[Ukázka vektoru]]

2.8 Konstrukce clusteringu

Microsoft Azure nabízí k přípravě experimentů svoje studio dostupné na adrese `https://studio.azureml.net`.

Ve studiu Azureml je možné vytvářet své projekty, do projektů umístit své experimenty a ty následně vystavit jako webovou službu.

Základem úspěšného experimentu je vytvořit učící model. To je takový model, pro který máme zvolený cílový algoritmus a na předpřipravených datech ho naučíme aby dokázal v našem případě co nejlépe rozdělovat zprávy do clusterů.

2.8.0.1 Předzpracování

Veškeré předzpracování a čištění dat probíhá v mojí aplikaci i přesto jsem základní předzpracování zvolil i do experimentu samotného.

Po načtení vstupních dat dochází odstranění duplicitních řádků. Jako další metoda je využití modulu, který smaže řádky, jimž chybí nějaká data.

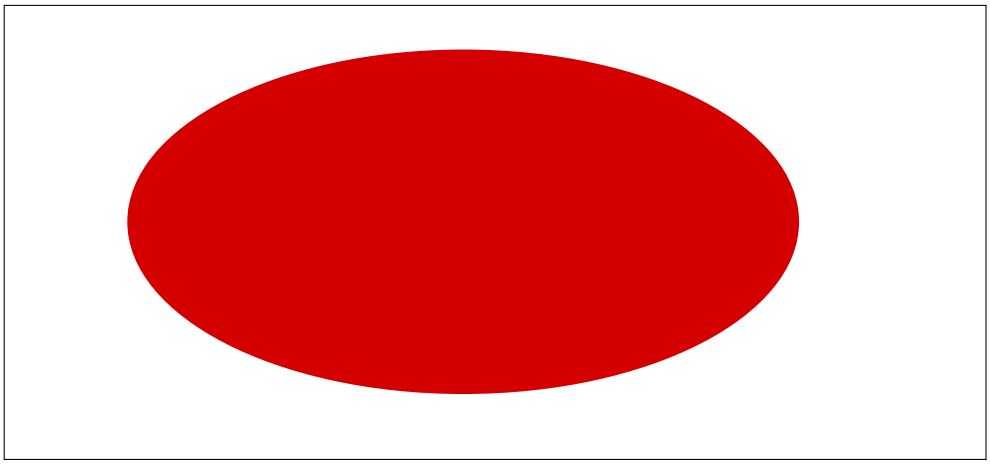
[[Mohl bych použít zároven s klasifikacnim modelem, ale nevim.]]

2.8.0.2 Zpracování

Pro zpracování jsem zvolil K-Means modul, který je připojený na modul pro trénování clusterovacích modulů.

Po natrénování přiřadíme zbytku testovacích dat clustery a může zhlédnout výsledek.

Na obrázku 2.3 je vidět celý vytvořený trénovací experiment.



Obrázek 2.3: Clustering k-means v prostředí MS AZURE ML Studio.

[[Doplnit sem ukázku přiřazení dat a grafy z azure]] [[jak jsem zjistil nejlepší vhodné nastavení]]

2.9 Konstrukce detekce anomálie

Druhou možností, kterou bych rád vyzkoušel je detekce anomálie. To, že chybné požadavky nebo bezpečnostní požadavky se budou výrazněji lišit od běžných zpráv se dá předpokládat.

Princip předzpracování dat v Azureml studiu je stejný jako v při konstrukci modelu pro clustering. Řádky s chybějícími hodnotami a duplikované pro trénování nebudeme používat.

Kromě výše uvedené předzpracující části i zde je část učící a část vyhodnovací.

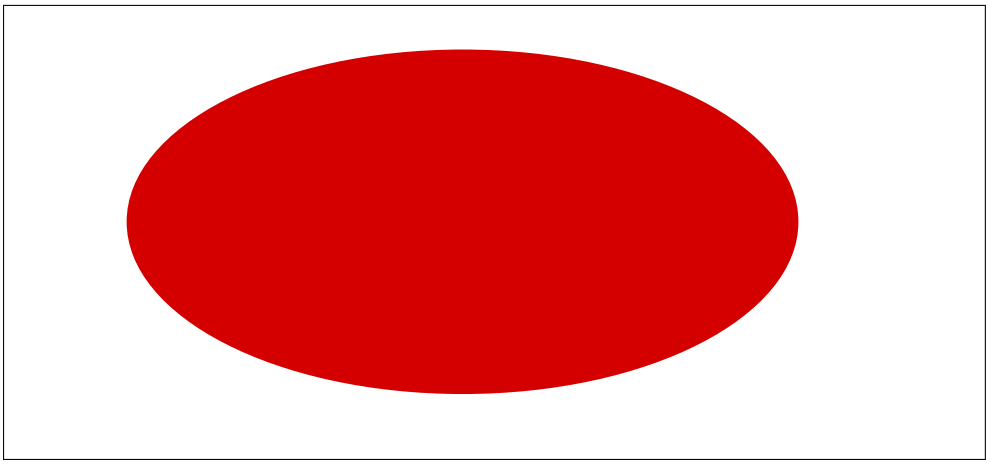
Trénovací model je vidět na obrázku 2.4.

[[jak jsem zjistil nejlepší vhodné nastavení]]

2.10 Prezentace dat

Vzhledem k tomu, že by aplikace měla být schopna určovat bezpečnostní rizika, je třeba nějakým způsobem prezentovat její výstupy. Moninitoring na aplikaci Unify je momentálně postaven na tom, že konkrétní lidé hlídají logy a v případě vyskytu chyb, varování nebo jiné netypické události zjišťují co bylo příčinou.

Rozhodl jsem se tedy, že nejlepší bude grafické znázornění. Kromě údajů o tom, že byl zaznamenán požadavek, který je podezřelý budu grafy využívat i k prezentaci základního monitoringu.



Obrázek 2.4: Detekce anomálií v prostředí MS AZURE ML Studio.

Vzhledem k tomu, že se bude veškerá komunikace ukladát bude vhodné prezentovat například i kolik požadavků na jednotlivé komponentě proběhlo za poslední hodinu a podobně.

Společnost Cetin a.s. [43] ve které bude aplikace testována a jenž je uživatelem integrační platformy používá pro různá grafická znázornění grafy od Google Charts[31].

Tyto grafy jsou napsané v jazyce Javascript. Je tedy možné jejich umístění například na intranetové stránky, kde se vysoce postavení lidé společnosti vyznají lépe než v jednotlivých monitorovacích aplikacích.

Na tomto základě jsem se rozhodl vytvořit REST API [30], jenž budou Google Charts schopny snadno konzumovat a v případné jiné aplikace, které by stály o podobná data budou schopny se jim přizpůsobit.

2.11 Využití dát systémy 3. stran

Do budoucna je potřeba počítat s rozšířením monitoringu a je proto vhodné aplikaci připravit tak, aby její výsledky mohly být využity v aplikacích 3. stran.

Lze předpokládat, že k monirování bezpečnosti provozu budou použity systémy SIEM (Security Information and Event Management) [44]. SIEM funguje na principu, kdy zpracovává co nejvíce údajú, na jejichž základě pak rozeznává neočekávané situace a rizika [45].

Tím, že jsem se rozhodl data ukládat tak, jak uvádím v kapitole 2.5 bude libovolný SIEM po připojení do DB schopen získat jak originální zprávu, tak její normalizovanou verzi popřípadě i výsledek vyhodnocení mé aplikace.

Dále je možnost napojit SIEM i na REST API obdobně jako Google Charts v kapitole 2.10.

Kapitola 3

Realizace

[[Sem vepsat nějaký úvod k této kapitole]]

3.1 Nutné přípravy pro jboss

3.1.1 Připravení modulů

V aplikaci využívám různé java knihovny, abych k nim měl přístup i v aplikačním serveru, je nutné do něj přidat speciální modul.

Jboss umožňuje snadné přidání modulů. Veškeré moduly jsou umístěny v *wildfly/jboss-eap-7.0/modules/system/layers*. Zde jsem vytvořil svůj modul s konkrétními java knihovnami:

- commons-codec-1.10.jar
- json-simple-1.1.1.jar
- mongo-java-driver-3.4.2.jar

3.1.2 Port offset

[[Je možné, že offset ve finále ještě změním.]] Dále bylo nutné pro jboss nastavit portový offset. Protože na serveru není jedinou aplikací, je běžný problém v kolizi portů. Z tohoto důvodu jsem zvolil offset 10000. Webové služby tedy místo portu 8080 běží na portu 18080.

3.1.3 Zapnutí CORS

CORS (Cross-origin resource sharing) neboli *sdílené zdroje odjinud* umožňuje odesílání odpovědí na požadavky z jiné domény [46]. V aplikaci je to potřebné pro rest api, kterého se následně dotazuje Google charts.

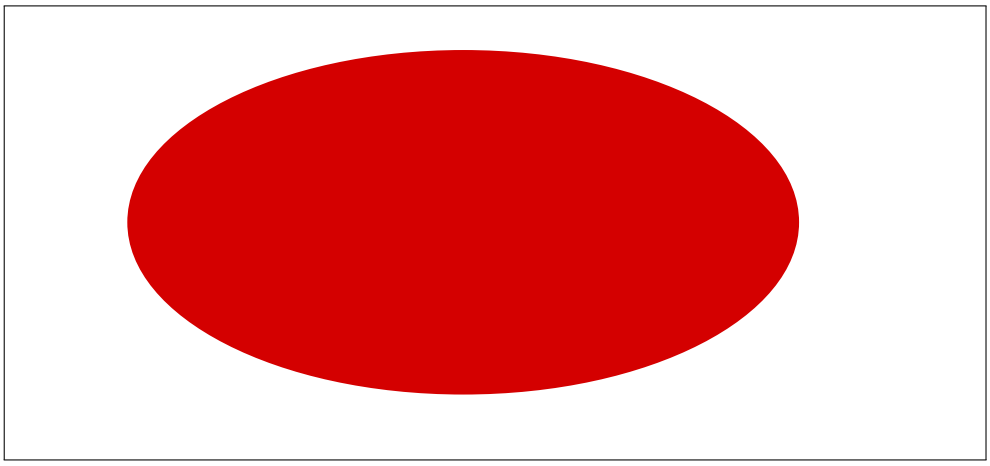
Corse se v Jboss povoluje v konfiguračním souboru pro standalone aplikaci *standalone.xml* pro doménovou *domain.xml*.

3.2 Vytvoření modelu na Azure

[[Ukázka URL]] [[Ukázka Vstupního JSONu a výstupního]] [[V této části bude Prediktivní algoritmus]]

3.2.1 Clustering v Azure

[[Popsat prediktivní experiment]] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

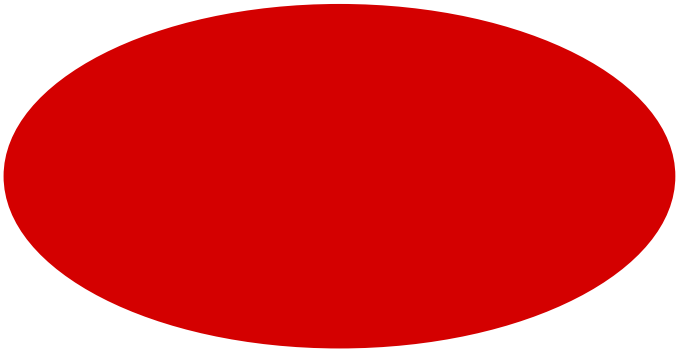


Obrázek 3.1: Prediktivní model clusteringu v Azure.

3.2.2 Detekce anomálií v Azure

[[Popsat prediktivní experiment]] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus.

Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

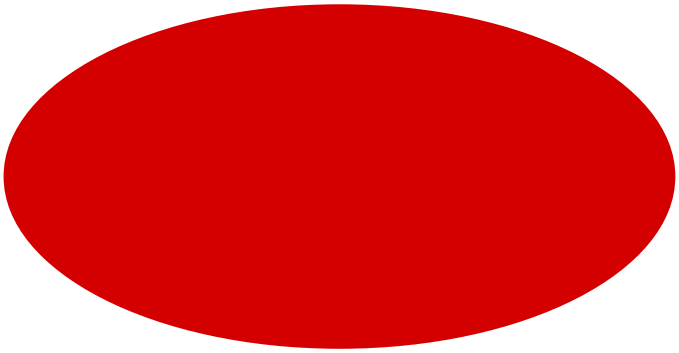


Obrázek 3.2: Prediktivní model v detekci anomálií v Azure.

3.2.3 Webová služba

Po dokončení prediktivního modelu je třeba experiment vystavit tak, abychom ho mohli používat z vlastní sítě.

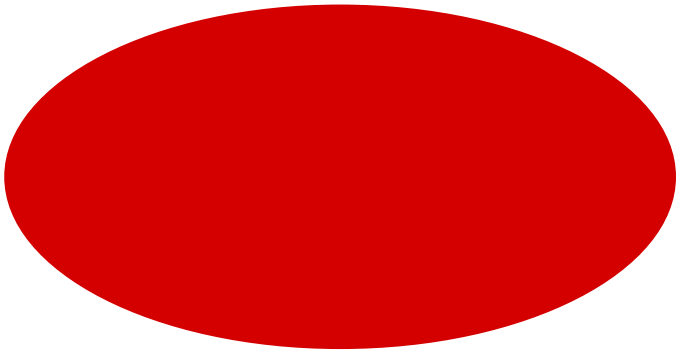
Azure umožňuje takový model spustit jako webovou službu.



Obrázek 3.3: Vytvoření webové služby pomocí stisku tlačítka.

Po vytvoření webové služby získáme takzvaný „API key“. Tento řetězec bude sloužit pro přihlášení se do Azure, při dotazování se na konkrétní službu.

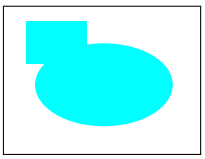
Také je možné službu otestovat. Otevře se nám okno s očekávanýma políčkama (obr. 3.4). Po vyplnění políček se zobrazí odpověď z prediktivního modelu. Tímto způsobem můžeme otestovat funkčnost nebo pár vzorků. Jiná použití by byla velmi časově a zdrojově nevýhodná.



Obrázek 3.4: Zobrazené okno pro otestování prediktivního modelu jako webové služby

3.3 MongoDB

[[Celkově z této sekce jsem rozpačitý]] Pro potřeby aplikace je nutné v co nejrychlejším možném čase ukládat jednotlivé requesty. Při ohromném provoze, který se na integrační platformě vyskytuje to je nutná podmínka pro to, aby bylo možné v reálném čase jednotlivé požadavky zpracovávat.



Obrázek 3.5: Logo MongoDB. [1]

Rozhodl jsem se za tímto účelem využít MongoDB [1], protože se očekává, že bude třeba ukládat v mimořádném případě až 30 záznamů za sekundu, zpravidla bude docházet k více zapisům než čtením.

3.3.1 NoSQL

MongoDB patří do takzvaných NoSQL databází [1]. NoSQL v angličtině znamená „Not Only SQL“ [47], v překladu „Nikoliv pouze SQL“. Jde o skupinu

nerelačních databází. Takové databáze nejsou primárně postavené na principu tabulek a zpravidla nepoužívají SQL pro práci s daty [47].

3.3.2 O MongoDB

MongoDB je licencovaná pod GNU AGPL v3.0 [48] licencí. Data jsou ukládána ve formátu BSON. BSON je binárně zakódovaná JSON [49].

V MongoDB se vytvářejí kolekce, každá kolekce obsahuje soubory. Soubory mají parametry [1]. Soubory a jejich parametry lze v čase libovlně měnit nebo přidávat. Což je výhoda, pokud zjistíme, že aktuální návrh není finální, vyhneme se problémům s migrací do nového schématu.

V rámci souboru je možné definovat čítač, který se využije k tomu, aby automaticky generoval jednoznačný identifikátor k souborům nebo lze využít parametr souboru **__id**. Ten vygeneruje jednoznačnou identifikaci, ze které jsme schopni například získat i čas vložení dokumentu do kolekce.

3.3.3 Využití v práci

3.3.3.1 Kolekce terms

V práci využívám databázi k ukládání všech slov, ze kterých se tvoří vektor, jenž reprezentuje konkrétní požadavek (více v kapitole 2.7). Tím není potřeba je mít v paměti a při připadném výpadku je znova vypočítávat.

V kolekci *terms* ukládám soubory jejichž struktura je automatický identifikátor, slovo pro konstrukci vekotru a timestamp přidání dokumentu do kolekce.

3.3.3.2 Kolekce messages

Další kolekcí je kolekce *messages*. V té jsou uložené veškeré požadavky, které byly přečteny z logů integrační platformy. Protože ještě před uložením do kolekce dochází v Azure k vyhodnocení, je zpráva uložené i s informací, která určuje zdali je požadavek vyhodnocen jako bezpečností riziko nebo není.

Struktura každého souboru je:

- **__id** - automaticky generovaný identifikátor
- **timestamp** - čas uložení souboru
- **original-message** - původní požadavek, tak jak byl převzat z logu integrační platformy
- **normalized-message** - požadavek ve znormalizované podobě
- **platform-id** - jednoznačný identifikátor v rámci integrační platformy
- **assignment** - informace od Azure s výsledkem přiřazení katogire

[[Lépe vysvětlit assignment]] [[Konfigurační kolekce]]

3.3.4 Práce s MongoDB v Javě

V implementaci jsem vytvořil třídu *MongoClientService* (aby bylo možné třídy využívat i v jiných modulech, musí se taková třída skládat z interfacu a jeho implementace, v textu se budu bavit o celku implementace a interfacu dohromady například jako o třídě MongoClientService). Tato třída umožňuje distribuci konkrétní databáZe napříč celou aplikací.

V jednotlivých modulech si vyvoláme instanci konkrétní databáze a nad tou jsme schopni pracovat. Ovladače pro MongoDB nám umožňují jak data v kládat, tak je číst.

3.4 Čtení dat z logů

[[Popis]] Při návrhu zisku jednotlivých požadavků z integrační platformy jsem vycházel z toho, že nový aplikace musí minimálně, či spíše vůbec nezatěžovat Unify [11]. Vzhledem k tomu, že přes integraci proudí veškerý provoz, je sama o sobě dosti vytížená a v případě, že by touto aplikací byl způsoben výpadek došlo by k silnému ztížení veškerých bussiness procesů, což si nelze dovolit.

Unify veškeré požadavky ukládá do logovacích souborů. Některé, převážně rizikové, služby se zároveň ukládají Oracle databáze. Ale vzhledem k tomu, že nejde o všechny dostupné služby rozhodl jsem se toho nevyužít.

Princip získání dat proudicích přes integrační platformu je založen na čtení jednotlivých logovacích souborů. Jako vhodný nástroj jsem vybral Java třídu Tailer z dostupné knihovny org.apache.commons.io [50].

Třída *Tailer*, po implementaci listeneru, se chová stejně jako linuxový příkaz tail [51]. Průběžně kontroluje čtený soubor a každou nově zapsanou řádku zpracovává.

Tímto řešením získáváme data z integrační platformy, aniž bychom jí zatěžovali.

3.5 Předzpracování a odeslání do Azure

[[Možná rozdělit na dvě sekce]]

Protože jsou data odesílána do cloudu, předzpracováváme je lokálně a přímo do Microsfot Azure odesíláme už jen identifikátor zprávy a vypočtený vektor.

Po přečtení zprávy z auditového logu Unify je zpráva předzpracována (2.6) a následně je z ní vytvořen vektor (2.7).

3.5.1 Start aplikace

První start aplikace je komplikovanější v tom, že pro výpočet finálního vektoru ještě nemáme známá vhodná slova, pro která se budou TF a IDF vypočítávat.

Pro případ, kdy je databáze zcela prázdná jsou nejdříve načteny nějaké zprávy (dle konfigurace), z těch jsou vypočetent vhodné termy. V případě, že databáze nějaké zprávy již obsahuje je možné využít je. Nedoporučovaný způsob je vložení slov přímo do databáze. Tato metoda může být vhodná v případě, že například chceme databázi migrovat a nechceme se zdržovat znovu výpočtem.

3.5.2 Získání dat z logu

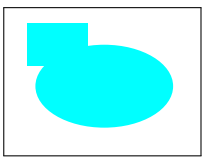
Samotný zisk dat z logu je řešen pomocí Java knihovny Tailer [52]. Jednou z věcí, které bylo potřeba vyřešit byla situace, kdy třída Tailer během čekání na nový přírustek logovacího souboru zcela zablokovala program. Situaci jsem vyřešil tím, že procesu, který čte z logu integrační platformy jsem pomocí *ExecutorService* [53] umožnil běžet na pozadí aplikace. Tím jsem neblokoval řízení programu.

3.5.3 Předzpracování dat

Celý proces předzpracování zpráv probíhá v implementované třídě *LogListener*. Po získání dat, jako textového řetězce, jsou uložena do struktury třídy *AuditLogMessage* (obr 3.6).

Následuje proces vytvoření vektoru, který je odesílán do MS Azure. Vektor se vytváří dle pravidel uvedených v sekci 2.7 včetně všech procesů předzpracování . Metody pro výpočet TF a IDF jsou implementované ve třídě *WeightsCounterService*.

Vektor samotný je reprezentován jako seznam *Double* čísel.



Obrázek 3.6: Struktura třídy AuditLogMessage.

Pro odeslání dat bylo třeba vytvořit třídu *AzureWebService*. Vytvoření vhodného požadavku na Azure je podmíněno přihlášením se do služby. Proto do hlavičky je přidána *Basic Access authentication* (jednoduché ověření přístup).

Na požadavek ihned dostaneme synchroní odpověď s výsledkem. Výsledek je zpracován a přiložen k datům načteným z logu.

3.6 Uložení dat

3.6.1 Kolekce

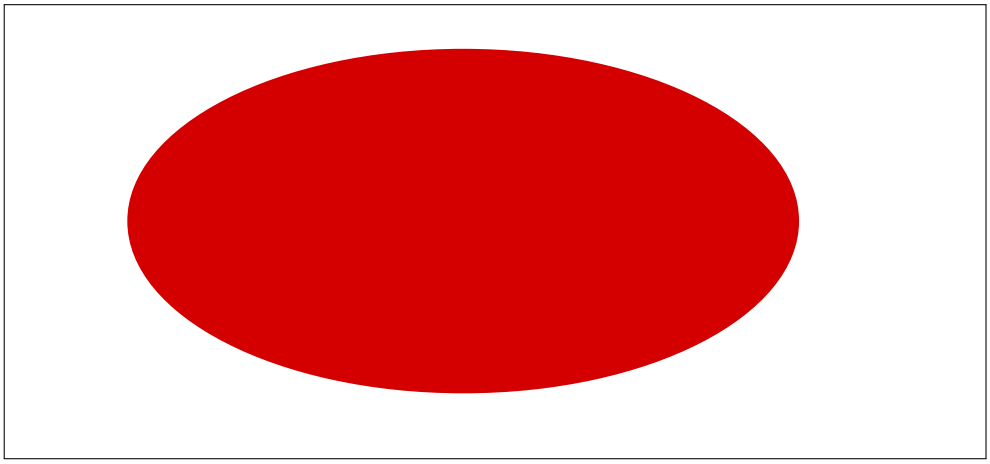
Do databáze se dvě kolekce:

- terms
- messages

Tyto kolekce (znázornění také na obrázku 3.7) jsou využívány při běhu programu.

Kolekce *terms* slouží k uchování výrazů, které se používají pro výpočet TF-IDF (kapitola 2.7). **[[Popsat strukturu kolekce]]**

Kolekce *messages* uchovává veškeré požadavky a to jak v jejich originální podobě (parametr original-message), tak její normalizovanou metodu (normalized-message). Podstatným identifikátorem každého požadavku je jeho unikátní ID v rámci platformy Unify. Pro tento účel je použit parametr platformId. Posledním parametrem této kolekce je assignment. Tento parametr uchovává informaci o tom, do jaké (rizikové) kategorie byl dle algoritmu běžícím v MS Azure zařazen.



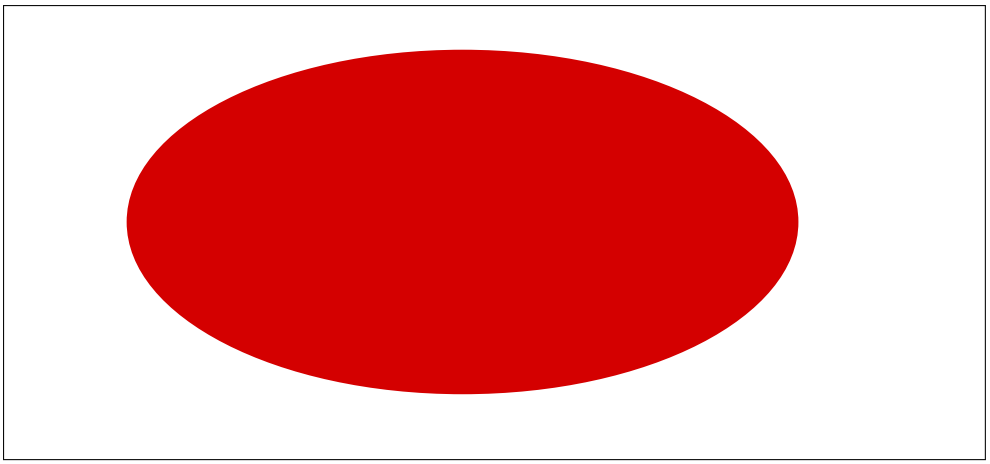
Obrázek 3.7: Kolekce v MongoDB využité pro běh programu.

3.6.2 Proces ukládání dat

Do kolekce terms jsou výrazy ukládán ihned po tom, co je určeno, že jsou správným termem. **[[Lépe popsat dle kodu]]**

Kolekce messages je plněna teprve po té, co se vrátí odpověď z MS Azure. Celá zpráva je ve třídě *LogListener* ukládána do formátu *AuditLogMessage*.

Ta kopíruje svým obsahem právě kolekci messages. *AuditLogMessage* je vlastníkem metody *toDBObject*, která obsah třídy vráti ve vhodném formátu pro uložení do MongoDB.



Obrázek 3.8: Sekvenční diagramy ukazující proces uložení dat do kolekcí terms a messages.

3.7 Napojení na Google Charts

Google vystavuje dokumentaci k produktu Google Charts na adrese **[[Dodat adresu]]**. Samotné grafy jsou generovány pomoc javascriptu. Z knihovny grafů lze vybrat nepřeberné množství různorodých grafů.

3.7.1 Rest API

Aby grafy byly generovány s daty z databáze bylo vytvořeno REST api. Toto api je postavené na míru Google Charts. Ovšem zpravidla není problém postavit další aplikaci tak, aby dokázala přijímat data z REST stejně popřípadě se je dokázala samostatně transformovat.

Za tímto účelem vznikl kompletně celý modul aplikace. Tento modul má dovětek *web-services*. Je složen ze dvou tříd:

- DataProviderService
- ChartsProviderService

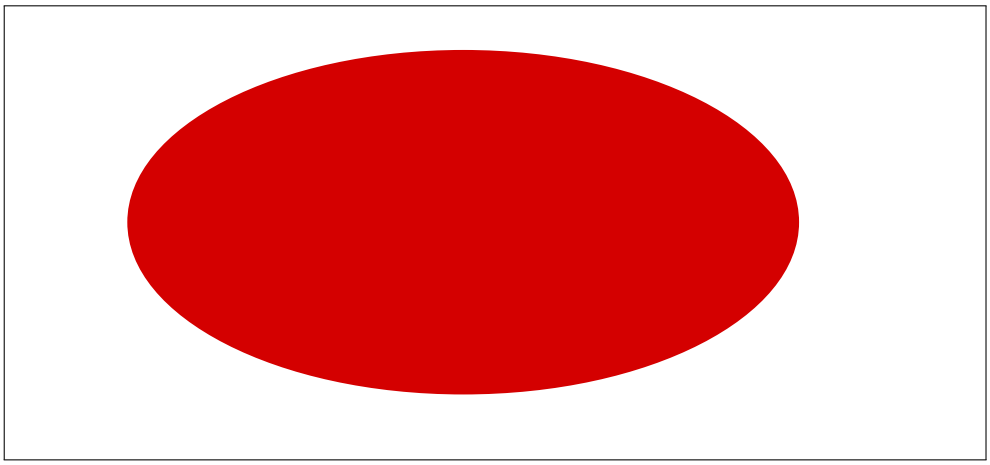
Každá třída je chápana jako samostatná služba.

DataProviderService vystavuje webové služby na konkrétních URL adresách. Tím také poskytuje na jednotlivé GET **[[Definovat zkratku GET]]**

požadavky na zmíněné URL [[**Definovat zkratku URL**]] odpovědi s daty z databáze.

Data z databáze ve správné struktuře vybíra služba promítnutá ve třídě ChartsProvderService.

Jak bylo zmíněno, grafy z Google Charts očekávají konkrétní odpověď v předem určeném formátu JSON [[**definovat JSON**]]. Ukázka projekce dat ve formátu JSON pro použití v grafu druhu Gauge [[**Bud vlozit zdroj nebo url**]] je na obrázku 3.9.



Obrázek 3.9: Data ve formátu json pro použití v Google Charts.

Kapitola 4

Analýza a vyhodnocení dat

[[Sem vepsat nějaký úvod k této kapitole]] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

[[Popsat systém, na který to bylo nasazené]] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4.1 Analýza K-Means

V této části je popsán celkový proces vytváření nejlepšího možného nastavení pro clusterovací k-means algoritmus. Jako základ jsem vybral trénovací data. U těchto dat jsem se rozhodl určit 1000 trénovacích vzorků. Protože cílem je detekovat bezpečnostní rizika a to jak ve formě špatných requestů tak i například podvržených zpráv, vybral z testovacích dat 500 vzorků, které do-

padli chybou a 500 vzorků, které proběhli v pořádku. První množina slouží jako reprezentanti skupiny zpráv, které chceme rozpoznávat. Druhá množina 500 prvků je reprezentantem klasických zpráv, které by měli přes integrační platformu proudit.

Pro poměr 50:50 jsem se rozhodl, aby byla dostatečně velká pravděpodobnost, že shluky v k-means najdou správná místa. Pokud bych zvolil poměr ve kterém by se podezřelé zprávy vyskytovali v malém množství nebo dokonce vůbec, algoritmus k-means by mohl pomocí shluků například rozpoznávat jednotlivé služby nebo najít zcela jiné vzory v datech. Takové vzory by se nemuseli hodit pro analýzu bezpečnostních rizik.

Trénovací data jsem rozdělil náhodně poměrem 80:20 na trénovací a testovací. Testovací budou použita k odhadu správnosti shlukování. Vzhledem k tomu, že jsem jednoznačný identifikátor podezřelých zpráv označil, je možné na první pohled dokázat odlišit takovou zprávu od zprávy korektní.

Následující sekce se zabívají jednotlivýma možnostma, jak ovlivnit běh algoritmu k-means. Pro všechny sekce byla použita stejná vstupní data.

[[Když vznikne prosto, mohl by se vyzkoušet nějaký reálnější poměr.]]

4.1.1 Počet shluků

K-means shlukuje data do k shluků vhledem k podobnosti jednotlivých dat. V experimentu bezpečnostních rizik lze očekávat, že by rozdělení mohlo být na dvě skupiny:

- data v pořádku
- podezřelá data

I přes tuto myšlenku byl proveden experiment na různá množství shluků. Konkrétně na 2, 3, 4, 10 a 20. Všechny experimenty byly spuštěny současně na totožných vstupních datech. Hodnoty ostatních konfiguračních parametrů (samozřejmě mimo počtu shluků) byly zvoleny následující:

- **výběr prvotních středových bodů:** náhodný
- **metrika:** eukleidova vzdálenost
- **počet běhů:** 100

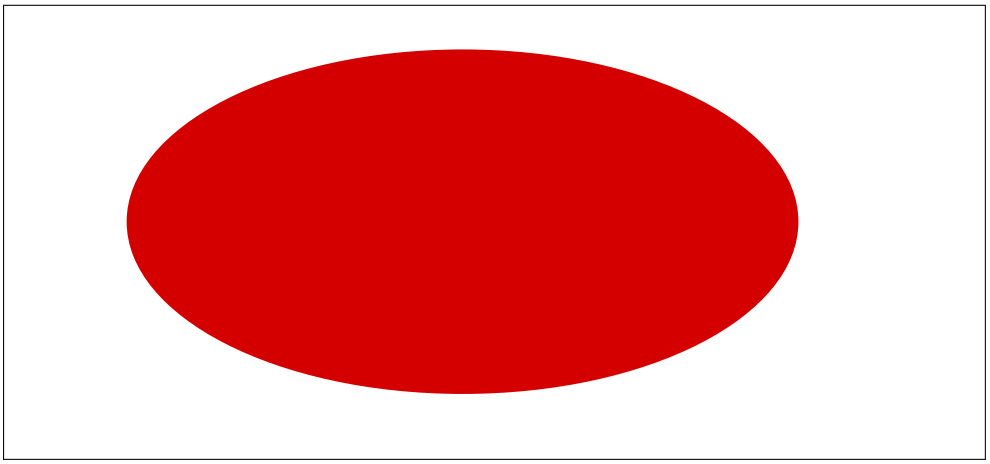
Výsledky jsou hromadně ilustrovány na obrázku 4.1. Na ose x vždy pozorujeme shluky označené přirozeným číslem. Osa y zaznamená počet zpráv ve shluku.

4.1.1.1 2 shluky

Pro volby dvou shluků vznikly dva na počet zpráv téměr shodné shluky. Protože poměr vstupních dat 50:50 jde očekávaný výsledek [[**očekávaný?**]]. Při kontrole konečných přiřazení shluků jsem narazil na zprávu, která by měla být označená jako podezřelá, ale bylo vyhodnocena jako zpráva korektní. V celé množině šlo o jedinou zprávu. Tato zpráva oznamovala, že došlo ke špatnému zadání hesla k jedné ze služeb. Jiná chyba nebyla nalezena [[**asi ne uplne nazyvat chybou**]]. Lze tedy říci, že dochází k rozdělení na podezřelé zprávy a zprávy korektní. [[**zkusit shrnout proč vznikla chyba**]]

4.1.1.2 3 shluky

Vizualizace výsledků přiřazení clusterů pro $k = 3$ nám ukazuje, že tři vzniklé shluky jsou o velikostech 35, 409 a 355 vzorků. Zde je rozdělení shluků takové, že jeden obsahuje pouze zprávy podezřelé. Další shluk obsahuje pouze zprávy v pořádku a stejně jako v případě dvou shluků podezřelou zprávu o špatném zadání hesla. Poslední, nejmenší shluk je kombinací. Po detailnějším rozboru vyplynulo, že zprávy v tomto shluku jsou oproti ostatním zprávám kratší. Normalizovaná zpráva se skládá zpravidla z několika slov. Výsledkem rozdělení trénovací množiny byly shluky podezřelých zpráv, korektních zpráv a velmi krátkých zpráv.



Obrázek 4.1: Zobrazené grafy s počtem zpráv v jednotlivých shlucích.

4.1.1.3 4 shluky

Čtyři shluky ukázaly pro jejich velikost podobný výsledek jako když $k = 3$. Největší cluster obsahuje zprávy většinu korektních zpráv a již zmiňované krátké chybové zprávy. Druhý shluk obsahuje pouze chyby. Charakteristikou

jde spíše o kratší zprávy převážně způsobené komponentou notifikačního enginu. Dále je přítomen shluk, jehož obsahem jsou jen zprávy chybové. Převážně jde o zprávy, které jsou delší a nesou například informaci o tom proč chyba nastala, nikoli pouze upozornění, že nastala. Poslední, obsahově nejmenší, shluk obsahuje pouze několik málo korektních požadavků.

4.1.1.4 10 shluků

Experiment s deseti shluky ukazuje na trend, kdy se k-means přizpůsobuje jednotlivým službám. U dvou shluků došlo ke kolizi korektních a podezřelých požadavků. Oba tyto clustery shlukovali převážně kratší zprávy. U těch je větší pravděpodobnost, že jejich vektor bude převážně nulový. Ostatní shluky obsahují buď chyby nebo správné požadavky, jen se přizpůsobují ke konkrétním službám. Je zde možné například rozpoznávat shluk, který obsahuje chyby hlásící špatný KMS [[**zkratka KMS**]] překlad a stejně tak shluk s chybami špatného přihlášení.

4.1.1.5 20 shluků

K-means s $k = 20$ potvrzuje vývoj, který se udál již při $k = 10$. Pouze jeden shluk je ze smíšených korektních zpráv a z podezřelých. Ten obsahuje pouze jeden záznam z podezřelých. Jednotlivé clustery více rozpoznávají jednotlivé služby a chyby. U chyb, jejichž druhů je ve vstupních dat méně než služeb se to projevuje více.

4.1.1.6 Shrnutí

Z celého vývoje dat, která byla použita na testování, plyne, že nejlepším je nechat dva shluky. Tím dokážeme rozlišit podezřelé zprávy od korektních. To zcela naplňuje požadavky detekce bezpečnostních rizik. Další možností je vytvořit shluků více a pokusit se i detekovat různé druhy chyb. Při této volbě bude vyžadováno více starostí organizováním, které clustery patří do jaké skupiny. Při velkém počtu shluků by mohlo docházet k přeučení, kdy by sice byly rozpoznávany konkrétní chyby konkrétní služby, ovšem s dalším nasazením nové služby by byly potíže.

4.1.2 Metrika

Pro porovnání různých způsobů měření vzdálenosti jsem zvolil na MS Azure dostnou eucleidovu vzdálenost (1.5.1.1) a cosinovu vzdálenost (1.5.2). Protože celá analýza ideálního počtu shluků byla založena na eucleidově vzdálenosti, další měření stačí provést už pouze na cosinově vzdálenosti.

Pro následující výsledky platí, že velikosti shluků jsou průměrovány z 10 spuštení programu. Nastavení k-means je následovné:

- **výběr prvotních středových bodů:** náhodný
- **metrika:** cosinova vzdálenost
- **počet běhů:** 100

Při zkoumání výsledků se objevil jev, kdy některé zprávy nebyly přiřazený do žádného clusteru. Po hlubší analýze jsem zjistil, že jde o zprávy, které pochází z B2B brány. Tyto požadavky po normalizaci mají velikost nula. To vede k tomu, že jejich vektor je nulový. S takovým vektorem má cosinova vzdálenost problém. Nelze jej vypočítat, tak zpráva není označena žádným shlukem.

Kromě těchto nulových dat dochází k falešně pozitivnímu označení některých výrazně menších zpráv jako bezpečnostní rizika. Konkrétně pro případ, kde $k = 3$ došlo rozdělení dat žádného shluku (prázndé zprávy), podezřelé zprávy, korektní a další skupinu korektních. V této skupině dominovali zprávy z REST rozhraních. Ty se od ostatních požadavků liší, že nejsou v XML zápisu ale v JSON.

Při náhledu, jak byly zprávy rozděleny do clusterů, při 4 shlucích lze opět pozorovat, že podezřelé zprávy jsou všechny z testovacích dat zahrnuty v jednom shluku. Ostatní shluky jsou složeny už pouze ze zpráv z běžné komunikace. Jejich rozdělení je pak závislé na charakteru zprávy.

Deset shluků také vykazuje dokonalé rozdělení testovacích na rizika a ostatní zprávy. Obě skupiny jsou rozloženy do stejného počtu shluků.

V případě $k = 20$ jsou data opět rozdělena. Ovšem narozdíl od testovaných $k < 20$ se zde objevují clustery, které obsahují například jen jednu zprávu.

Z výsledků analýzy, kdy porovnáváme metriku euikleidovu a cosinovu je pozorovatelná lepší přesnost u cosinovy délky. Problém zde nastává se zprávy, které mají nulový vektor. Tyto zprávy je třeba ošetřit nebo nahradit normalizační funkci tak, aby nulový vektor negenerovala.

4.1.3 Inicializace

Vliv na běh algoritmu má i počáteční nastavení středů jednotlivých shluků. MS Azure ve svém modulu pro k-means nabízí možnosti [**?**]:

- **Prvních N**. $N \in \mathbb{N}$ prvních vzorků z dat je určeno jako střed shluku.
- **Náhodné**. Jsou vybrány náhodné body ze vstupních vzorků jako středy.
- **K-Means++**. Algoritmus definovaná Davidem Arthurem a Sergeiem Vassilvitskii [**?**]
- **K-Means++Fast**. Optimalizovaná verze K-Means++.
- **Rovnoměrně**. Středy jsou rozmístěné ve stejné vzdálenosti v prostoru.

- **Dle popisku sloupce**. Algoritmus, který na základě popisu sloupe dat rozhoduje o tom, jak budou středy umístěny.

Z experimentu jsem se rozhodl vyřadit K-Means++Fast, jenž je optimalizovanou verzí K-Means++ a inicializaci dle popisku sloupce, neboť jsem nedohledal oficiální dokumentaci k metodě, jak funguje.

Pro experiment jsem zvolil následující nastavení:

- **výběr prvotních středových bodů:** výběr pomocí komponenty Sweep K-means **[[Přidat sweep k-means do volby počtu shluků.]]**
- **výběr prvotních středových bodů:** předmětem experimentu
- **metrika:** eukleidova i cosinova vzdálenost
- **počet běhů:** 100

[[kmeans++ U kosinu 1 FP, ale ve skutecnosti slo o chybu, ze dms nenasla soubor!]] [[PrvníchN eukl - 30 shluku]] [[PrvníchN cosi - 30 shluku]] [[Náhod. e - 11]] [[Náhod. c - 12]] [[even. e - 11]] [[even. c - 22]]

Výsledky experimentu ukazuje tabulka 4.1. V tabulce se vyskytují počty zpráv, které byly v jednotlivých metodách označeny falešně pozitivní (dobrá zpráva označená jako podezřelá) a falešně negativní (podezřelá zpráva označená jako korektní). U všech běhů pro cosinovu metriku bylo 20 vždy 20 zpráv neoznačeno žádným číslem shluku. Jde o nulové vektory, které cosinova vzdálenost nedokáže vypočítat.

Z tabulky (4.1) se jeví nejlepší metoda prvních N. Porovnáme-li výsledky této tabulky s obrázkem 4.2, který ukazuje graf závislosti počtu shluků na konkrétní metodě inicializace (výpočítané pomocí metody sweep clustering) je vidět, že metoda prvních vytváří velký počet shluků. Maximální počet byl stanoven na 30. Až na této hranici se prvních N zastavila. Vzniklo tím mnoho méně obsazených shluků. Jednotlivé zprávy byly rozděleny korektně na podezřelé a nepodezřelé. Velká počet shluků pak vedl i k většímu počtu rozeznaných chyb. Problém by mohl nastat s novou, v trénovacích datech neexistující, chybou. Proto i přes skvělý výsledek neshledávám tuto metodu ideální.

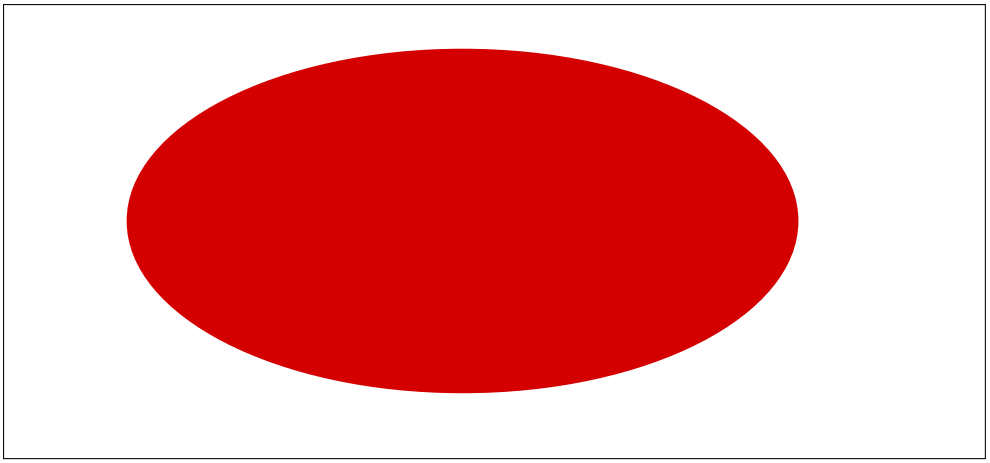
Ve všech ostatních metodách inicializace se při měření eukleidovou metrikou objevila jedna stejná podezřelá zpráva, která byla označena jako nepodezřelá. Jde o chybu, která se svého druhu v trénovacích datech objevuje jediná. Celkově to ukazuje na slabost při zjišťování nových chyb.

Cosinova metrika si vedla značně lépe, než eukleidova. Ve vstupních datech měla jen jeden případ špatného určení. Šlo o případ, kdy korektní zpráva byla označená za chybnou. Při bližším zkoumání jsem zjistil, že jde zprávu, která hlásí nenalezení dokumentu v uložišti. Ve skutečnosti algoritmus našel a

označil jako chybu zprávu, která nesla informaci o chybě. Z pohledu administrátora platformu může jít o zajímavou informaci, neboť dotazuje-li se někdo na neexistující soubor jde také o bezpečnostní riziko.

Tabulka 4.1: Zobrazení chyb při shlukování 800 vzorků, při rovnoměrném rozdělení korektních a podezřelých zpráv.

	Prvních N		Náhodné		K-Means++		Rovnoměrně	
	FP	FN	FP	FN	FP	FN	FP	FN
Eukleidova metrika	0	0	0	1	0	1	0	1
Cosinova metrika	0	0	0	0	1	0	0	0



Obrázek 4.2: Počty clusterů, které jako ideální vyhodnotila metoda sweep k-means.

4.2 Analýza detekce anomálie

[[Link do 2. kapitoly]] Při detekci anomálie jsem kromě experimentů s ideálním nastavením vytvořil i experiment mezi dvěma metodami. Jedna je detekce anomálie založená na principu PCA, jejíž princip je přiblížen v kapitole 1.6.2. Druhou metodou je One-Class Support Vector Machine **[[Udělat kapitolku.]]**, dále bude označována zkratkou SVM.

4.2.1 Popis vstupních dat

Rozdíl v přístupu mezi shlukováním a detekce anomálie je v trénovacích datech. Při detekování anomálií byla jako testovací data použita pouze data, která byla manuálně určena jako nezávadná. Dále byla připravena i množina

dat testovacích. Testovací data obsahují v poměru 87:13 korektní zprávy a podezřelé.

Velikost obou množin je 1000 požadavků. Vstupy byly vybrány náhodným výběrem z téměř 50000 požadavků velké množiny na ESB. Zprávy, které integrační platformou v pořádku prošli byly hodnoceny jako korektní. Zprávy, které vyvolávali chyby byly označené jako podezřelé. Nevýhodou tohoto přístupu může být přehlédnutí rizikové zprávy, která běžným pohledem se může jevit v pořádku.

4.2.2 Ohodnocení klasifikace

Pro dosažení nejlepšího možného výsledku používám MS Azure komponentu nazvanou „Tune Model Hyperparameters.“ Jde o komponentu, která se snaží optimalizovat parametry modelu, tak aby našla ideální nastavení [34]. Na základě toho je nutné být schopen rozeznat, jestli předchozí ohodnocení parametrů je horší než stávající. K tomu poslouží tyto možnosti nastavení [**?**]:

- **Přesnost:** lze vyjádřit vzorcem $\frac{tp+tn}{tp+tn+fp+fn}$
- **Správnost:** lze vyjádřit vzorcem $\frac{tp}{tp+fp}$
- **Podíl:** lze vyjádřit vzorcem $\frac{tp}{tp+fn}$
- **F-score:** **[[todo powers-precision]]**

Experiment s ohodnocením klasifikace na základě přesnosti je zobrazen v tabulce 4.2. Lze pozorovat, že ani při jedné z použitých metod nedošlo k označení podezřelé zprávy ve zprávu korektní. Z hlediska bezpečnosti je lepší prověřovat správné zprávy, než ignorovat podezřelé. Jediný rozdíl je v tom, že SVM hodnotí méně korektních zpráv jako podezřelé. Při bližším pohledu na takové zprávy jde převážně o kratší požadavky.

Tabulka 4.2: My caption

	PCA		SVM	
	P-M: korektní	P-M: podezřelé	P-M: korektní	P-M: podezřelé
Korektní	525	165	656	34
Podezřelé	0	102	0	102

Po změně hodnotícího vzorce na Správnost se výsledná data příliš nezměnila. V tabulce 4.3 je vidět, že hodnoty pro SVM zůstali totožné, jako v případě použití Přesnosti. U PCA došlo k lehkému zhoršení. Toto zhoršení je rovno 14 korektním zprávám, které byly označené jako podezřelé.

Změna hodnocení na Podíl pro výsledek příliš změnu neznamenal (hlavně v porovnání s hodnocením Správnost). Tabulka 4.4 ukazuje, že jediná změna oproti hodnocení Správnost je o jednu zprávu víc, která byla chybně označena

Tabulka 4.3: My caption

	PCA		SVM	
	P-M: korektní	P-M: podezřelé	P-M: korektní	P-M: podezřelé
Korektní	511	179	656	34
Podezřelé	0	102	0	102

jako podezřelá v případě použití PCA. SVM si stejně jako v předchozích dvou experimentech drží stejné hodnoty.

Tabulka 4.4: My caption

	PCA		SVM	
	P-M: korektní	P-M: podezřelé	P-M: korektní	P-M: podezřelé
Korektní	510	180	656	34
Podezřelé	0	102	0	102

Použití F-Score vedlo k totožným výsledkům jako experiment s Přesností. Více ji znázorněno v tabulce 4.5

Tabulka 4.5: My caption

	PCA		SVM	
	P-M: korektní	P-M: podezřelé	P-M: korektní	P-M: podezřelé
Korektní	525	165	656	34
Podezřelé	0	102	0	102

Při porovnání výsledků klasifikace F-Score a Přesnosti došlo ke zcela totožným výsledkům. Obě tyto metody byly z výše uvedených i nejvýše úspěšné z pohledu PCA. SVM nezaznamenalo žádnou změnu na hodnocení klasifikace.

[[Popsat vstupní parametry]] [[Ukázat výsledky na Prod logu]] [[Dojít k závěru]]

Kapitola 5

Závěr

Conclusion

Literatura

[1] *MongoDB*. Available from: www.mongodb.com

[2] Available from: https://www.nbu.cz/cs/pravni-predpisy/1091-zakon-o-kyberneticke-bezpecnosti-a-o-zmene-souvisejicich-zakonu-zakon-o-kyberneticke-bezpecnosti/

[3] ISO 27001. Available from: http://www.rac.cz/rac/homepage.nsf/CZ/BS7799

[4] Online. Available from: http://www.businessdictionary.com/definition/information-system.html

[5] Bilge, L.; Dumitras, T. Investigating Zero-Day Attacks. Online, 08 2013.

[6] ZETTER, K. Online, 01 2016. Available from: https://www.wired.com/2016/01/hacker-lexicon-what-are-dos-and-ddos-attacks/

[7] Petukhov, A.; Kozlov, D. Detecting Security Vulnerabilities in Web Applications Using Dynamic Analysis with Penetration Testing. *Computing Systems Lab, Department of Computer Science, Moscow State University*, 2008. Available from: https://www.owasp.org/images/3/3e/OWASP-AppSecEU08-Petukhov.pdf

[8] Corona, I.; Giacinto, G. Detection of Server-side Web Attacks. *Department of Electrical and Electronic Engineering, University of Cagliari, Italy*, 2010.

[9] Ahn, S.-H.; Kim, N.-U.; et al. Big Data Analysis System Concept for Detecting Unknown Attacks. *Department of Electrical and Computer Engineering, Sungkyunkwan University, College of Information and Communication Engineering, Sungkyunkwan University*, 2014.

[10] Rouse, M. security information and event management (SIEM). Online. Available from: http://searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM

[11] *Unify integration platform*. Available from: https://www.physter.com/unify/

[12] Available from: https://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb

[13] Available from: http://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html

[14] Available from: https://www.liaison.com/2016/02/22/b2b-integration-survival-of-the-fittest-4/

[15] Available from: http://www.oracle.com/technetwork/java/javaee/tech/javaee6technologies-1955512.html

[16] Paul, S. K.; Agrawal, M.; et al. An Information Retrieval(IR) Techniques for text Mining on web for Unstructured data. *International Journal of Advanced Research in Computer Science and Software Engineering*, volume 4, 02 2014, ISSN 2277 128X. Available from: https://www.ijarcsse.com/docs/papers/Special_Issue/icadet2014/Lord_18.pdf

[17] Witten, I. H. Text mining. *Computer Science, University of Waikato, Hamilton, New Zealand*, 0. Available from: http://www.cos.ufrj.br/~jano/LinkedDocuments/_papers/aula13/04-IHW-Textmining.pdf

[18] Ghosh, M. S.; Roy, M. S.; et al. A tutorial review on Text Mining Algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, volume 1, 06 2012, ISSN 2278 – 1021. Available from: https://pdfs.semanticscholar.org/5fc6/b674cde1f39847b8783349af200eb68c9d48.pdf

[19] Sankar, K.; Babu, D. G. N. K. S. A Study of Text Mining For Web Information Retrieval System From Textual Databases. *International Journal of Emerging Technology and Advanced Engineering*, volume 3, 12 2013, ISSN 2250-2459.

[20] Ramos, J. Using TF-IDF to Determine Word Relevance in Document Queries. *Department of Computer Science, Rutgers University*, 0. Available from: https://pdfs.semanticscholar.org/b3bf/6373ff41a115197cb5b30e57830c16130c2c.pdf

[21] Vijayarani, D. S.; Ilamathi, M. J.; et al. Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science & Communication Networks*, volume 5, 2016, ISSN 2249-5789. Available from: http://www.ijcscn.com/Documents/Volumes/vol5issue1/ijcscn2015050102.pdf

[22] Zhao, Q.; Bhowmick, S. S. Association Rule Mining: A Survey. Online. Available from: http://www.lsi.upc.edu/~bejar/amlt/material_art/assrules20zhao03association.pdf

[23] Agrawal, R.; Imielinski, T.; et al. Mining Association Rules between Sets of Items in Large Databases. Online. Available from: http://www.almaden.ibm.com/cs/quest/papers/sigmod93.pdf

[24] Tan, P.-N.; Steinbach, M.; et al. *Introduction to Data Mining*. Addison-Wesley, first edition, 2005, ISBN 978-0321321367. Available from: http://www-users.cs.umn.edu/~kumar/dmbook/index.php

[25] Bhusare, B. B.; Bansode, S. M. Centroids Initialization for K-Means Clustering using Improved Pillar Algorithm. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, volume 3, 04 2014. Available from: https://pdfs.semanticscholar.org/b278/74148c4af4d3eedc64909b0b738e5b1c73cf.pdf

[26] Deepthi, A. S.; Rao, D. K. Anomaly Detection Using Principal Component Analysis. *IJCST*, volume 5, 10 2014, ISSN 0976-8491. Available from: http://www.ijcst.com/vol54/1/28-Adathakula-Sree-Deepthi.pdf

[27] Anand, P. R.; Kumar, T. K. PCA Based Anomaly Detection. *International Journal of Research in Advent Technology*, volume 02, 2014, ISSN 2321-9637. Available from: http://www.ijrat.org/downloads/feb-2014/paper20id-222014114.pdf

[28] Breunig, M. M.; Kriegel, H.-P.; et al. LOF: Identifying Density-Based Local Outliers. *Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX*, 2000. Available from: http://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf

[29] Abdi, H.; Williams, L. J. 2010. Available from: http://www.utdallas.edu/~herve/abdi-awPCA2010.pdf

[30]

[31] *Google Charts*. Available from: https://developers.google.com/chart/

[32] Johnson, C. H. High Level Design Distributed Network Traffic Controller. 02 2005. Available from: https://people.ok.ubc.ca/rlawrenc/research/Students/CJ_05_Design.pdf

[33] Available from: https://www.microsoft.com/cs-cz/

[34] Microsoft, . *Microsoft Azure*. Available from: https://azure.microsoft.com/cs-cz/

[35] *Microsoft Azure Machine Learning Studio*. Available from: https://studio.azureml.net/

[36] *DEPLOY ANYWHERE WITH RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM*. Available from: https://www.redhat.com/cms/managed-files/mi-deploy-anywhere-jboss-eap-datasheet-inc0405103lw-201605-en.pdf

[37] *Apache Log4j 2*. Available from: https://logging.apache.org/log4j/2.x/

[38] *Class PatternLayout*. Available from: https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

[39] Sproat, R.; Bedrick, S. CS506/606: Txt Nrmlztn. 2011. Available from: http://www.csee.ogi.edu/~sproatr/Courses/TextNorm/

[40] Li, W. Automatic Log Analysis using Machine Learning. 11 2013. Available from: http://uu.diva-portal.org/smash/get/diva2:667650/FULLTEXT01.pdf

[41] Neto, J. L.; Santos, A. D.; et al. Document Clustering and Text Summarization. *Pontificia Universidade Catolica do Parana Postgraduate Program in Applied Computer Science Rua Imaculada Conceição 1155 Curitiba - PR, 80215-901. Brazil*, 0. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.4634&rep=rep1&type=pdf

[42] Ramos, J. Using TF-IDF to Determine Word Relevance in Document Queries. *Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 08855*, 0. Available from: https://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf

[43] *CETIN*. Available from: https://www.cetin.cz/

[44] Constantine, C. SIEM and Log Management - Everything you need to know but were afraid to ask, Part 1. 2014. Available from: https://www.alienvault.com/blogs/security-essentials/everything-you-wanted-to-know-about-siem-and-log-management-but-were-afraid

[45] Work?, H. D. S. Written by Colton Bachman. 2016. Available from: https://www.integritysrc.com/blog/313-how-does-siem-work

[46] w3.org. Cross-Origin Resource Sharing. 2014. Available from: https://www.w3.org/TR/cors/#introduction

[47] Moniruzzaman, A. B. M.; Hossain, S. A. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, volume 6, 04 2013. Available from: https://arxiv.org/ftp/arxiv/papers/1307/1307.0191.pdf

[48] Available from: https://www.mongodb.com/community/licensing

[49] Available from: http://bsonspec.org/

[50] *Tailer class*. Available from: https://commons.apache.org/proper/commons-io/javadocs/api-2.4/org/apache/commons/io/input/Tailer.html

[51] *Tail*. Available from: https://www.gnu.org/software/coreutils/manual/html_node/tail-invocation.html

[52] *Class Tailer*. Available from: https://commons.apache.org/proper/commons-io/javadocs/api-2.4/org/apache/commons/io/input/Tailer.html

[53] *Interface ExecutorService*. Available from: https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ExecutorService.html

Příloha A

Acronyms

GUI Graphical user interface

XML Extensible markup language

Příloha B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format