VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA LOGŮ A UTILIZACE HARDWARE

LOG ANALYSIS AND HARDWARE UTILIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ KUCHYŇKA**
AUTHOR

VEDOUCÍ PRÁCE **Mgr. Ing. PAVEL OČENÁŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Kuchyňka Jiří**
Program: Informační technologie
Název: **Analýza logů a utilizace hardware**
Log Analysis and Hardware Utilization
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se se službami pro sběr logů a jejich analýzou.
2. Analyzujte možnosti pro vytvoření událostí na základě získaných dat se zaměřením na utilizaci hardware.
3. Po konzultaci s vedoucím práce navrhněte informační systém pro zpracování nasbíraných dat a jejich vizualizaci.
4. Implementujte navržený systém.
5. Ověřte implementovaný systém v praxi na reálných datech.
6. Zhodnoťte získané výsledky a diskutujte možnosti rozšíření.

Literatura:

- Kurose, J. F. Computer networking: A top-down approach. Pearson, Essex, 2017, ISBN 978-1-292-15359-9.
- Stallings, W. Network security essentials: Applications and standards. Hoboken, 2016, ISBN 978-0-13-452733-8.
- Bishop, M. Computer security: Art & Science. Addison-Wesley, Boston, 2003, ISBN 0-201-44099-7.
- Ahmed, M., Mahmood Naser, A., Hu, J. A survey of network anomaly detection techniques. Journal of network and computer applications. Elsevier, 2016, 60(C), s. 19-31. ISSN 1084-8045.
- Buczak, A., Guven, E.. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications surveys and tutorials. IEEE, 2016, 18(2), s. 1153-1176.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Očenášek Pavel, Mgr. Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 27. října 2020

Abstrakt

Tato práce se zabývá návrhem a implementací systému k dlouhodobému monitorování stavu linuxových systémů umístěných v produkčním prostředí. Práce se zaměřuje zejména na situaci, kdy dané systémy nemají možnost odesílat sesbíraná data k analýze po síti, proto sběr dat musí probíhat zcela automaticky a data musí být jinou cestou přenášena z daných systémů na centrální systém ke shromažďování, analýze a vizualizaci. Podstatná část práce se věnuje návrhu a implementaci webové aplikace sloužící k exportu dat z monitorovaných systémů na přenosové médium a importu z něj na systém ke shromažďování dat. Výsledné řešení má za cíl zjednodušit sběr dat ze systémů, dříve vykonávaný přímo správci daných systémů tak, aby ji dokázal provádět kdokoliv, kdo se k monitorovanému systému může fyzicky přiblížit a snížit tím náklady spojené s monitorováním těchto vzdálených systémů.

Abstract

The goal of this thesis is to design and implement a system for long-term monitoring of the state of Linux systems located in a production environment. The thesis focuses mainly on situation in which the system does not have the ability to send the collected data for analysis over the network, so data collection must be completely automatic and data must be transferred from monitored systems to a central system for collection, analysis and visualization. A substantial part of the work is devoted to the design and implementation of a web application used to export data from monitored systems to the transmission medium and import them from it to the system for data collection. The resulting solution aims to simplify the collection of data from systems, previously performed directly by system administrators, so that it can be performed by anyone who can physically approach the monitored system and thus reduce the costs associated with monitoring these remote systems.

Klíčová slova

monitorování systému, Linux, logování, logy, utilizace hardware, zatížení systému, ELK stack, Elasticserch, Logstash, Kibana, Apache Kafka, zabezpečení dat, analýza dat, vizualizace dat

Keywords

system monitoring, Linux, logging, logs, hardware utilization, system load, ELK stack, Elasticserch, Logstash, Kibana, Apache Kafka, data security, data analysis, data visualization

Citace

KUCHYŇKA, Jiří. *Analýza logů a utilizace hardware.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Ing. Pavel Očenášek, Ph.D.

Analýza logů a utilizace hardware

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Mgr. Ing. Pavla Očenáška, Ph.D. Další informace mi poskytl také pan Ing. Petr Chmelař a pan Ing. Michal Pustka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Kuchyňka
10. května 2021

Poděkování

Na tomto místě bych rád poděkoval všem, kteří při mně během studia stáli a podporovali mě. Také bych rád poděkoval panu Ing. Petru Chmelařovi, panu Ing. Michalu Pustkovi a panu Mgr. Ing. Pavlu Očenášekovi, Ph.D. za rady a připomínky, které mi během realizace této práce poskytli.

Obsah

Kapitola 1

Úvod

Se zvyšující se složitostí systému vzniká stále větší potřeba monitorovat jeho chování za účelem odhalit možné zdroje problémů. V dnešní době k tomuto účelu existuje mnoho nástrojů ke sběru logů a jiných dat, která mohou být následně analyzována a jinak zpracována například k detekci anomálií.

Dokud má člověk ve správě pouze pár systémů, není důležitost sběru logů a jiných dat příliš zřejmá. Většinu problémů, které přímo ovlivňují výkonnost či chybovost systému, je možné odhalit i pomocí nástrojů zabudovaných v operačním systému. Ostatní anomálie, které se nikterak viditelně neprojevují, způsobené například průniky do systému, zůstanou neodhalené. S přibývajícím množstvím systémů se ale začíná složitost jejich správy a monitorování zvyšovat. Málokdo by chtěl každých pár dní kontrolovat na stovce systémů, zda vše běží jak má a čekat, až některý ze systémů přestane fungovat. tento přístup se silně nevyplatí, problémy spojené se ztrátou dat či odstávkami provozu způsobenými tímto přístupem silně převažují nad případnými nevýhodami či náklady na zprovoznění třeba i jen jednodužšího systému k monitorování a sběru dat. V minulosti tak začal vznikat různorodý software, který umožňoval centrálně shromažďovat data z dalších systémů a s těmito daty dále pracovat.

Cílem takovéhoto software bylo zesnadnit proces získávání, filtrování, transformování a vizualizace dat, tak aby bylo hledání potřebných informací v datech jednodušší a efektivnější. Toho je často dosahováno vizualizací získaných dat pomocí grafů a jiných grafických znázornění.

I přes rostoucí pokrytí světa internetovým připojením a jinými sítěmi, stále existují situace, při kterých není možné s některými zařízeními udržovat spojení. V některých specifických situacích se například firma může rozhodnout dané zařízení nepřipojit do sítě a jediný způsob, jak z takovéhoto zařízení sbírat data, je pak limitovaný na fyzickou návštěvu zařízení s flash diskem v době, kdy zařízení není aktivně využíváno. Během této návštěvy jsou pak na daný flash disk přenesena data ze systému, která v době od poslední návštěvy sesbíral. Pokud ale takovýto proces není nijak automatizovaný, může tato činnost zahrnovat několikahodinovou práci lidí, kteří musí data manuálně zpracovat a umístit ne předem domluvené místo. Zároveň takovýto přenos dat nemusí být bezpečný, data ze systému v nešifrované podobě na flash disku či jiném médiu mohou představovat únik citlivých dat z daného systému.

Aby k takovýmto situacím nemuselo docházet, rozhodl jsem se, že v této práci zmapuji všechny dosavadní možnosti a zpracuji, jak je možné danou problematiku v současné době vyřešit. Práce se zaměřuje zejména na konkrétní situaci GNU/Linux systému umístěného u výrobní linky zákazníka bez přístupu k síti.

Převážná většina řešení se v dnešní době zaměřuje pouze na přímou komunikaci s monitorovaným zařízením po síti. Není tak jednoduché dosáhnout řešení tohoto problému za využití pouze existujících technologií. Tento fakt mě vedl k rozhodnutí zvolit si jako hlavní náplň bakalářské práce implementovat systém pro GNU/Linux zařízení, který by zajišťoval sběr, přenos, analýzu a vizualizaci logů, utilizace hardware a jiných dat. Zde zpracovaný systém je určen pro využití v zařízeních, která nemají možnost pomocí síťového spojení odesílat sesbíraná data cílovému systému pro shromažďování a vizualizaci dat a sesbíraná data tak musí být předávána jinými prostředky. Při využití tohoto systému může sběr dat ze systémů zajišťovat kdokoliv. Například s využitím USB flash disku může kdokoliv s přístupovými údaji stáhnout data ze zařízení a následně je nahrát na cílový systém pro shromažďování a vizualizaci dat.

Probíraný systém má za cíl usnadnit práci firmám, které potřebují sbírat data z většího množství zařízení a přitom nemají možnost tato data získávat přes síťové spojení za zařízeními. Zvýšením automatizace tohoto procesu umožní firmám jednodužšeji odhalit a opravit nedostatky jimi vyvíjeného software, čimž docílí kvalitnějšího a efektivnějšího vývoje, aby jimi vyvíjený produkt byl lépe konkurenceschopný.

Práce se nejprve věnuje v kapitole 2 principům práce s logy, rozebírá typy logů a existující technologie pro práci s nimi. Kapitola 3 se zabývá ochranou dat při přenosu, věnuje se základům zabezpečení dat, využití asymetrické a symetrické kryptografie k ochraně dat tak, aby taková data nebylo možné číst jinými zařízeními při přenosu. Také se věnuje elektronickým podpisům jako nástroji k dosažení autentizace a celistvosti dat. V následující kapitole 4 je probrán návrh systému, jsou diskutovány cíle řešení a požadavky na výsledný systém. Poté se kapitola 5 věnuje implementaci navrženého systému. Nejprve jsou probrána vybraná existující řešení a důvody jejich volby, poté se kapitola zabývá implementací všech částí systému a nakonec implementací nástroje pro export a import dat. Poslední kapitola 6 se věnuje analýze dat sesbíraných během testování implementovaného systému na reálných datech, zaměřuje se na monitorování výkonu systému, operačního systému a kritických procesů.

Kapitola 2

Principy práce s logy

Tato kapitola se zabývá logy a jednotlivými kroky sběru a analýzy dat v současné praxi. Obsahuje průzkum existujících řešení v této oblasti a uvádí čtenáře do této problematiky.

Aby bylo možné pro systémové administrátory sledovat, co se v systému děje, mnoho programů zaznamenává svoji činnost – události, které za běhu nastaly. Takováto posloupnost událostí se nazývá log, který lze následně využít k pochopení aktuální činnosti systému a k případné diagnostice problémů.

Mnoho techniků zná logy pouze v podobě textových souborů, které různé systémové služby generují. V prostředí distribuovaných systémů jsou ale tyto logy méně časté. Mnohem častěji se setkáváme s logy, které jsou strukturované a místo pouhého zaznamenávání činnosti programu mohou nést informace o událostech, popisovat změny v databázi, atd. Takovéto logy pak velice často nejsou čtené lidmi, ale dalšími programy, které na ně různými způsoby reagují. Například databázový systém může číst log jiného databázového systému a replikovat operace, které provádí, a tím vytvářet druhou kopii dané databáze na dalším systému.

2.1 Typy logů

Existuje více typů logů. Většina lidí si pod pojmem log nejspíše představí klasický textový log – soubor obsahující mnoho řádků textu s informacemi o požadavcích, chybách či jiných zprávách.

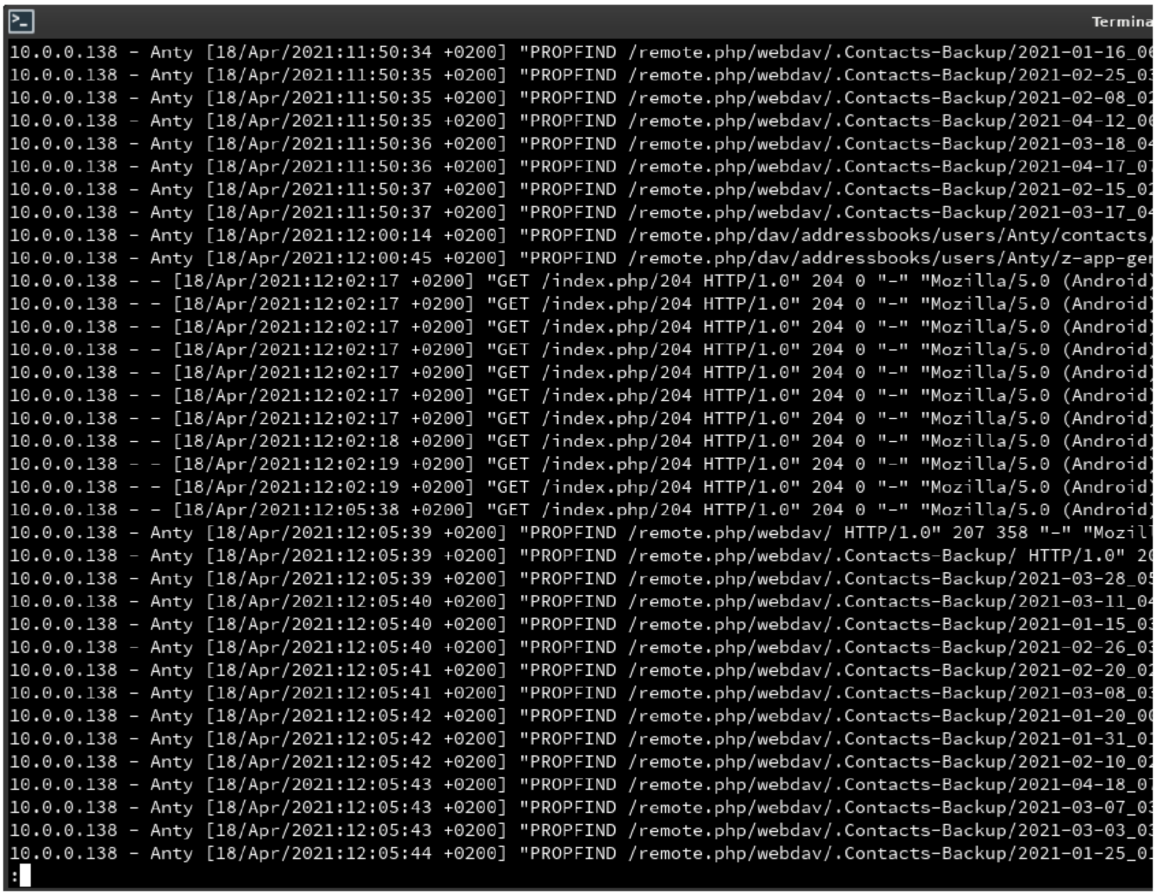
Tento typ logu je ale určen převážně pro čtení lidmi a není jednoduše strojově analyzovatelný. Existuje ale i jiný, zcela odlišný druh logů. Tento druh logů spíše připomíná databázi, nežli text. Takovéto logy většinou nejsou přímo analyzovány či jinak využívány lidmi, dokonce většina takovýchto logů je cíleně určena pro jiné programy a k lidem se vůbec dostat nemusí. V případě, kdy je však potřeba takovéto logy analizovat, povětšinou jsou podle daných pravidel nejprve zpracovány, analyzovány a různými způsoby vizualizovány pomocí k tomu určených počítačových programů, a teprve v této podobě jsou následně prezentovány lidem. Tato kapitola byla inspirována dělením logů z knihy [18].

Textové logy

Mnoho programů umožnuje generovat log: webové servery vypisující historii požadavků a chyb, či operační systém vypisující postup initializace hardware při spuštění systému. Všechny tyto logy mají jednu věc společnou, jejich formát je dobře čitelný pro lidi. Například takto vypadá textový log přístupů k Nginx serveru 2.1.

Při řešení problémů se serverem mohou být tyto logy velice užitečné - pouhým pohledem do takovéhoto logu v době kdy server nefungoval lze velice často nalézt možný zdroj problému. Tento druh logů ale ztrácí svoji váhu v prostředí distibuovaných systémů. S přibývajícím množstvím oddělených instancí nabízející určitou službu přibývá i množství proměnných, kterými je při snaze odhalit zdroj problému nutné se zabývat.

Zatímco tento druh logů je velice známý mezi většinou techniků, při správě větších systémů je potřeba pracovat i s jinými druhy logů. S takovými logy, které je možné strojově zpracovávat, agregovat a vizualizovat.



Obrázek 2.1: Vzor textového logu z Nginx serveru.

Strojové logy

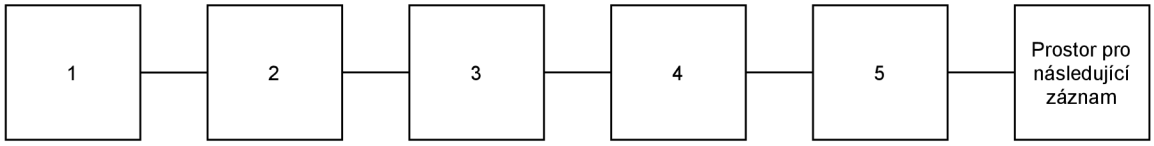
Zvané též žurnálové logy. Tento druh logů spíše připomíná databázi, nežli text, od databáze se ale výrazně liší. Můžeme si ho představit jako sekvenci záznamů, řazenou podle času, do níž lze pouze přidávat nové záznamy. Pro znázornění, jak tento typ logů funguje, obrázek 2.2 ukazuje strukturu takového logu.

Čas v této struktuře ale nepředstavuje čas vytvoření, nýbrž pořadí záznamu od vzniku sekvence. Každému záznamu je přiděleno unikátní pořadové číslo, které následně funguje jako unikátní klíč, takovýto záznam je následně uchován v pořadí, ve kterém byl vytvořen v databázi. Obsah každého záznamu může být různorodý. Například databázové systémy mohou do takovéhoto logu zaznamenávat veškeré akce provedené nad databází.

Každý takovýto log tak má vlastní čas. Místo ukládání jednotlivých záznamů podle reálného času se řídí tyto logy časem, který se posune o jednu jednotku při každém nově přidaném záznamu. Na první pohled se může zdát tento přístup práce s časem jako s pořadím záznamů neintuitivní, ale umožnuje jednodušeji s takovými logy pracovat. Zároveň nic nebrání v rámci struktury záznamu uchovávat i reálný čas, kdy byl záznam vytvořen a následně tento reálný čas při zpracování tohoto záznamu využít.

Strojový log není natolik rozdílný od souboru či tabulky. Jedná se pouze o jiný pohled na problém. Soubor reprezentuje pole bajtů, tabulka reprezentuje pole řádků. Log pak je pouze druh tabulky či souboru, který reprezentuje pole záznamů, které jsou řazené podle času, kdy byly vytvořeny.

Ani tento druh logu není ve svém základu příliš rozdílný od klasických textových logů. Strojové i textové logy jsou tvořeny sekvencí záznamů, do které lze pouze přidávat nové záznamy. Zásadním rozdílem však je, že záznamy strojových logů mohou mít jakoukoliv strukturu. Každý záznam může například reprezentovat objekt s mnoha atributy. Naproti tomu textové logy reprezentují záznamy pomocí řádku textu.



Obrázek 2.2: Struktura strojového logu v databázi. Čísla jednotlivých záznamů odpovídají jejich času.

2.2 Zdroje logů

Logy mohou pocházet z mnoha zdrojů. V každém prostředí mohou být důležité rozdílné zdroje logů a klíčovým krokem před prací s logy tak je nalézt ty správné zdroje logů k monitorování pro danou situaci. Všechny dostupné zdroje v dané situaci se mohou zdát důležité, ale dokázat vybrat z nich ty nejrelevantnější může velice zjednodušit jejich následné monitorování.

Následující seznam ani zdaleka nezahrnuje všechny možné zdroje logů. Uvádí pouze nejčastější zdroje, nad kterými stojí za to se zamyslet. Tato kapitola byla inspirována seznamem zdrojů logů na webové stránce [19].

Systémové logy

Není žádnou zvláštností, že logy ze systémů v síťové infrastruktuře nabízí mnoho informací o svém stavu. Operační systémy Windows i Linux neustále generují mnoho logů, které mohou pomoci při snaze pochopit proč se chovají tak jak se chovají. Mnoho událostí může nastat za běhu operačního systému a s ním spojených aplikací. Rozpoznání které logy mohou být důležité a mohou potřebovat okamžitou reakci a které pouze upozorňují na běžnou událost může být složité, ale i přesto stojí za to logy z tohoto prostředí sbírat.

Logy webových serverů

Zpracování logů z webových serverů sice může být pracný proces, ale při snaze pochopit jak uživatelé pracují s webovou stránkou můžou tyto logy být jedním z nejlepších zdrojů infor-

mací. Apache, Nginxm, Tomcat, Web Sphere, ale i jiné webové frameworky nabízí určitou úroveň logování aktivity webového serveru. V závislosti na požadavcích může i informace, kdy a odkud lidé chodí na danou webovou stránku,pomoci pochopit potřeby uživatelů a tím uvést budoucí směr vývoje stránky na správnou cestu. Bohužel log aktivity webového serveru patří mezi často přehlížené logy v době, kdy organizace pracuje na své strategii logování.

Logy autentizačních serverů

Bez ohledu na to zda k autentizaci je použit nástroj Active Directory, některá z implementací rozhraní OpenLDAP, či některé z jiných řešení, vždy je z bezpečnostního hlediska užitečné vědět, kdo co dělal v rámci dané infrastruktury. Každý autentizační server nabízí nějakou úroveň logování, klíčem ale je pochopit, co v těchto záznamech hledat. Mezi často sledované patří požadavky na získání tokenů, požadavky na zrušení oprávnění a záznamy o selhaných pokusech o přihlášení. Tyto typy logů mohou pomoci při řešení problémů spojených se selhanými pokusy o přihlášení z důvodů prošlých přihlašovacích údajů, při izolování zdroje potenciálního útoku, či poukázat na problematické oblasti infrastruktury, které by mohly potřebovat lépe zabezpečit.

Logy aplikací

Téměř každá aplikace nabízí nějakou podobu logů, které generuje za běhu. I přesto, že některé aplikace využívají nástroje pro zaznamenávání a správu logů z operačního systému, existují i aplikace, které vytváří logy vlastní cestou například do logových souborů. Proto jsou velice často i důležité logy ukládány jen tak na disk do souboru a běžné nástroje na sběr logů je mohou přehlédnout. Při řešení problému s konkrétní aplikací mohou být takovéto logy dostačující, ale při spojení více takovýchto modulů dohromady, například backend a frontend aplikace, může být složitější korelovat informace mezi těmito logy. Každý takový modul může logy sbírat rozdílným způsobem a člověk při pohledu na takový log nemusí příliš rozumět, co špatného se stalo. Proto je i z takovýchto aplikací sběr logů důležitý. Kombinace těchto logů společně se systémem k jejich analýze může pomozi odhalit problémy a pomoci s optimalizací takovýchto aplikací.

Zařízení síťové infrastruktury

Tato zařízení jsou hlavní křižovatkou síťové infrastruktury. Směrovače, přepínače, mosty, přístupové body, ale i jiné síťové prvky mohou být nakonfigurovány tak aby nabízeli statistiky využití a informace o svém stavu. Záznamy z těchto zařízení mohou pomoci při řešení celé řady problémů, od vynechávajícího spojení s přístupovým bodem, až po hardwarová selhání.

Pravděpodobně nejzásadnější jsou ale události upozorňující na změny konfigurace zařízení či na jiné nečekané události z jejich systému. Informace příčině změny konfigurace může pomoci vyřešit problémy od chybné konfigurace zaměstnancem, až po napadení systému tohoto zařízení.

Logy datového toku síťové infrastruktury

Tento zdroj možná není zcela běžně k vidění na seznamech důležitých zdrojů logů, protože se dnešní systémy často přesouvají na virtualizovaná prostředí či do cloudu, ale sběr logů

datového toku síťové infrastruktury, která zajišťuje systémům přístup k datům a propojuje je navzájem, je stále důležitý. Pokud přepínač připojený do sítě tvořené optickými kabely ztratí spojení se serverem, server přichází o přístup k datům. V dnešní době mají tyto linky velice často redundantní cesty, přes které mohou vést data, spojení proto nemusí být při prvním selhání ztraceno, ale i v těchto případech je potřeba selhání řešit. I při více než dvou redundantních spojeních v rámci série nečekaných událostí po dobu několika měsíců může většina redundantních spojení selhat. V takovémto stavu jsou data z pohledu serveru stále dostupná, ale propustnost linky může být silně limitována. Žádná služba v tomto stavu nemusí havarovat, protože spojení stále funguje, ale rychlost přenosu dat je silně zpomalena. Ve výsledku takovýto stav může vést ke zpomalení systému, který se může zdát v očích běžného člověka nepochopitelný, přitom logy datového toku z síťové infrastruktury by mohly pomoci při hledání problémů spojených se snížením propustnosti linky.

Logy bezpečnostních zařízení

Se stále přibývajícími společnostmi které přesouvají svoji infrastrukturu do cloudu, zařízení na okraji sítě se mohou stát mnohem důležitějšími z pohledu bezpečnosti sítě. Firewally a jiné bezpečnostní zařízení se musí starat o kontrolu stále více průchozí aktivity sítě spojené s rozšiřováním infrastruktury. Záznamy z těchto bezpečnostních zařízení mohou obsahovat mnoho užitečných informací. Statistiky blokovaných požadavků, stav a zdraví VPN služeb, události ze systémů na detekci a prevenci proniknutí do systému a další událoasti spojené s neobvyklou aktivitou uživatelů. Tyto SIEM (Security Information and Event Management) logy patří velice často k prvnímu zdroji informací při obraně proti útoku, jako takové pomáhají pochopit, jak útok fungoval a pomáhají izolovat, jaký vliv měl útok na síť. Nástroje jako například SolarWinds umožňují monitorovat logy událostí a detekovat podezřelou aktivitu, s pomocí těchto informací je pak možné reagovat na potenciální hrozby v reálném čase.

Logy hypervizorů

Hypervizory jakožto prvky síťové infrastruktury jsou velice užitečné. Místo odděleného hardware pro každou službu je možné více služeb provozovat na stejném hardware, aniž by hrozili bezbečnostní komplikace spojené během více služeb v rámci jediného systému. V případě napadení jedné služby běžící ve virtuálním stroji v hypervizoru je možnost této nákazy ovlivňovat běh ostatních služeb silně limitována. Správce infrastruktury tak může odvádět lepší práci ochrany, balancování zátěže a využití zdrojů na celém stroji. Skupina hypervizorů se pak může starat o stovky i tisíce současně běžících virtuálních strojů. Protože ale velká část práce hypervizorů probíhá srytě v rámci operačního systému a systémových služeb, mohou různé problémy zustat skryté a bez prohledání logů a jiných záznamů aktivity neodhalené.

Hypervizor neustále pracuje se zdroji systému na kterém běží, přealokování zdrojů z jednoho virtuálního stroje na jiný, alokovat nové místo pro data virtuálního stroje a jiné operace neustále prováděné k udržení všech virtuálních strojů v provozu. Záchytávání a monitorování logů z hypervizorů tak může jedna z nejlepších cest jak pochopit co vše hypervizory skrytě dělají. Optimalizace využití zdrojů a odstranování hardwarových problémů pak je s logy po ruce jednodušší.

Logy kontejnerů

I přesto, že kontejnery patří spíše mezi nové techonologie v porovnání s ostatními zdroji logů v tomto seznamu, zažívají v dnešní době vysoký nárůst využití a v mnoha společnostech se stávají kritickým bodem pro fungování infrastruktury. Na vyšší úrovní nad kontejnery mohou být různé systémy správy kontejnerových služeb jako například Kubernetes, Docker Swarm či Apache Mesos. Tyto systémy představují v prostředí kontejnerů podobnou roli jako hypervizory v prostředí virtuálních strojů, od hypervizorů se ale liší v mnoha ohledech.

Pochopit, proč se systém správy kontejnerových služeb rozhodl snížit či zvýšit počet replikací kontejneru, které se starají o daný deployment, se může zdát bez logů obtížné. Při snaze takovéto chování optimalizovat může být klíčové chápat, co k danému chování systém správy kontejnerů vedlo. Všechny tyto informace lze z logů kontejnerů a systémů správy kontejnerů získat.

Logy ze strojů u klientů

Přestože se může zdát jednodužší prodávat klientům hotová řešení a v případě problémů nechat řešení na zákazníkovi, v praxi je ve firemním prostoru mnohem častější přístup, kdy dodavatel nabízí k dodanému řešení nějakou míru podpory. Problém nastává v situaci, kdy je potřeba určit, co se pokazilo. Může být možností odeslat k zákazníkovi technika, který celý problém zanalyzuje a někdy je takovéto řešení i nutné, ale vždy se vyplatí, když je možné problém vyřešit vzdáleně. Problém může být způsobený uživatelem, hardwarovým problémem i chybou v kódu. Není nutné sbírat každý log, který lze ze stroje sbírat, dokonce takový přístup může být i kontraproduktivní, ale sbírání vybraných logů ze zařízení u klientů může poskytovat mnohem lepší přehled o situaci při následném řešení některých častých a klíčových problémů.

Ostatní zdroje logů

Existují i další zdroje logů, od proxy serverů a vyrovnávačů zátěže až po systémy správy cloudových služeb. Bez ohledu na to, zda se jedná o server, aplikaci či hardwarové řešení, vždy mohou logy z tohoto zařízení pomoci k získání lepšího přehledu o síti. Vždy je dobrý nápad přemýšlet nad tím, jaké logy sbírat a jaké už, ne a rozhodně existuje i mnoho zdrojů logů, které v tomto seznamu nejsou uvedeny.

2.3 Vytváření logů

Logy mohou reprezentovat mnoho druhů informací. Podle této reprezentace existují různé způsoby, jak takové logy mohou vznikat. Zatímco záznamy chyb mohou vznikat vyvoláním kódu k zachycení výjimky, jiné záznamy tvořené vzorkováním stavu programu, systémů či využití zdrojů, mohou vznikat kontrolami daného stavu v nekonečném cyklu. Tato sekce se věnuje několika způsobům vytváření logů a popisuje je z vlastní zkušenosti, tak jak je možné je vnímat v praxi.

Záznamy aktivity

Tento způsob vytváření logů patří mezi nejznámější způsoby. Jo to dáno tím, že mnoho programátorů jej již od začátků programování používá k hledání chyb v programech. Stačí k pár důležitým místům v kódu aplikace připsat příkaz, který vypíše na výstup aplikace

informaci, že daný kód právě proběhl a záznam aktivity je na světě. Jednoduchý záznam aktivity by mohl vypadat například takto 2.3, kdyby byl přepsán do lidsky čitelné podoby.

Mezi záznamy aktivity se také řadí logy selhání programu. Webové servery často takovéto logy nabízí a zaznamenávají do nich záznamy o selhání při zpracování požadavku, pokud byla chyba způsobena na straně serveru.

Záznamy aktivity mohou být užitečné k mnoha druhům analýzy. Lze z nich například vyvodit, jak uživatelé používají aplikaci či webovou stránku – stačí se podívat na seznam posledních stránek, které danný uživatel navštívil, či jaké ovládací prvky využil. Při takovéto analýze lze kombinovat data mnoha uživatelů a vizualizovat, jak se uživatelé chovají.

Při práci s těmito logy lze také měřit výkonnost či zatížení služeb. Kolik obsloužil tento server uživatelů za poslední hodinu? Jak dlouho dané službě trvalo zpracovat požadavek z jiné služby? Tyto všechny dotazy mohou záznamy aktivity pomoci zodpovědět.

Právě byl zpracován požadavek na cestu /index.php z adresy 99.88.77.66, kód výsledku je 200.
Právě byla vytvořena záloha serveru.
Při čtení souboru /etc/mojekonfigurace.conf došlo k chybě: Nedostatečná oprávnění.
Sloučen duplicitní blok dat na disku o velikost 128kb.
Načtena konfigurace ze souboru /etc/mojekonfigurace.conf.

Obrázek 2.3: Vzor záznamu aktivity programu v lidsky čitelné podobě.

Žurnál změn

Běžný člověk se s tímto způsobem vytváření logů pravděpodobně nesetká. I přesto se ale jedná o velice zajímavý způsob, jak logy vytvářet. Moderní souborový či databázový systém zastává mnohem více funkcí nežli pouhé uchovávání dat. Jednou z těchto funkcí je zachování integrity systému i v případě nečkané ztráty napájení, či při selhání operačního systému. Existuje více způsobů, jak této funkcionality dosáhnout, ale důležité k pochopení vytváření žurnálových logů pro nás bude pouze jedno řešení, a to pomocí žurnálu změn s transakcemi. Aby nedošlo k porušení integrity systému, musí být systém schopný případné nedokončené změny vrátit či zpětně dokončit. Mnoho systémů k tomuto účelu využívá takzvaného žurnálu. Před každou operací nad daty je nejprve do žurnálu zapsána informace o tom, jaká změna se chystá, a po dokončení změny je tato změna potvrzena v žurnálu. Pokud při spuštění systém zjistí, že je v žurnálu změna, která není označena jako dokončená, pokusí se vrátit případné změny, které byly provedeny, ale nebyly dokončeny.

Využití žurnálových logů ale zdaleka nekončí u detekcí a oprav chyb. Co kdyby bylo možné takovýto log změn odesílat na jiné zařízení, kde běží stejný systém se stejnými daty a nastavit takovýto systém tak, aby tyto změny také aplikoval na svá data? Takto nejspíše přemýšleli lidé při zrodu distribuovaných systémů. Žurnálové logy tak představují

popis změn dat distribuovaných databázích a souborových systémech a jejich využití sahá od replikování dat na více zařízeních po celém světě až po analyzování způsobů úprav dat, které v daném systému nejčasteji nastávají. Pro názornost obrázek 2.4 ukazuje, jak by mohl žurnál změn vypadat, kdyby byl přepsán do lidsky čitelné podoby. Obor distribuovaných systémů se zabývá i mnoha dalšími, často i složitějšími problémy, nežli je replikování dat pomocí žurnálových logů, ale těm se již tato sekce nebude věnovat.

Obsah bloku na pozici 123 změněn na hodnotu 99887766.
Vytvořen nový rádek v tabulce stromy s hodnotami ...
Soubor /etc/mojekonfigurace.conf byl ozančen jako smazaný.

Obrázek 2.4: Vzor žurnálu změn v lidsky čitelné podobě.

Log událostí

Na první pohled se může zdát, že logy událostí jsou velice podobné záznamům aktivit i přesto, že oba tyto způsoby vytváření logů na svém výstupu vytváří pomyslný log záznamů říkajících „teď se něco stalo“, způsob, jak vzinkají je rozdílný.

Ve snaze detekovat nečekané chování existují programy, které prochází jiné logy a hledají v nich různými způsoby anomálie. Při nalezení anomálie je následně tímto programem vygenerována událost, informující o tomto nálezu. Při přepsání takovýchto událostí do lidsky čitelné podoby by vypadaly například takto 2.5.

Některé programi také genrují události kterými oznamují jiným programů, že se stalo něco na co by mohli chtít reagovat. Například registrační modul může generovat událost „Právě se zaregistroval nový uživatel“, na niž mohou jiné moduly reagovat tím, že si například u sebe inicializují konfiguraci pro tohoto uživatele v databázi.

Příkladů vytváření logů událostí existuje mnohem více, například mnoho operačních systémů generuje události při připojení nového hardware. Logy událostí také často doprovází i záznamy aktivity – program zapíše do logu informaci, že právě zaregistroval nového uživatele a hned na to vytvoří událost s touto samou informací. I přesto v rámci logování má tento zdroj logů své místo jako zdroj logů, na který mohou jiné programy přímo reagovat na rozdíl od záznamů aktivity.

Vzorkování stavu

V prostoru záznamů aktivity existuje určitý druh stavů a činností, které nelze nelze klasickým zpusobem zaznamenávat do logů. To je dáno způsobem jakým se vyvýjí. Zatímco běžný program má velice často přesně definovaný stav po celou dobu běhu a tento stav se mění v závislosti na činnosti, kterou vykonává, s využitím prostředků systému je tento problém citelně složitější. Teoreticky by mohlo být možné zapisovat do záznamu aktivity každou alokaci a dealokaci paměti RAM systému, ale kvůli množství takovýchto událostí a problému, který vzniká ve chvíli, kdy k odeslání tohoto záznamu musí dojít k alokaci paměti se rychle dostaneme do prostoru, kdy takovýto log v podstatě není možné sbírat.

Zaregistroval se uživatel abcd.
Byl detekován nečekaný nárůst chybovosti při vyřizování požadavků serverem.
Požadavek od uživatele abcd pochází z nedůvěryhodné IP adresy.

Obrázek 2.5: Vzor logu událostí v lidsky čitelné podobě.

V takovém případě přichází na řadu vzorkování stavu. Místo vytváření záznamů pro každou zmenu paměti, bude na daném systému běžet služba, která se bude starat o periodické kontrolování stavu paměti. Tímto způsobem vznikají logy, které vzorkují stav. Pro názornost obrázek 2.6 ukazuje, jak by takový log vzorkující stav využití procesoru mohl vypadat, kdyby byl přepsán do lidsky čitelné podoby.

Využití procesoru v čase 00:20:00 bylo 2%.
Využití procesoru v čase 00:20:30 bylo 10%.
Využití procesoru v čase 00:21:00 bylo 7%.

Obrázek 2.6: Ukázka logu vzorkování stavu v lidsky čitelné podobě.

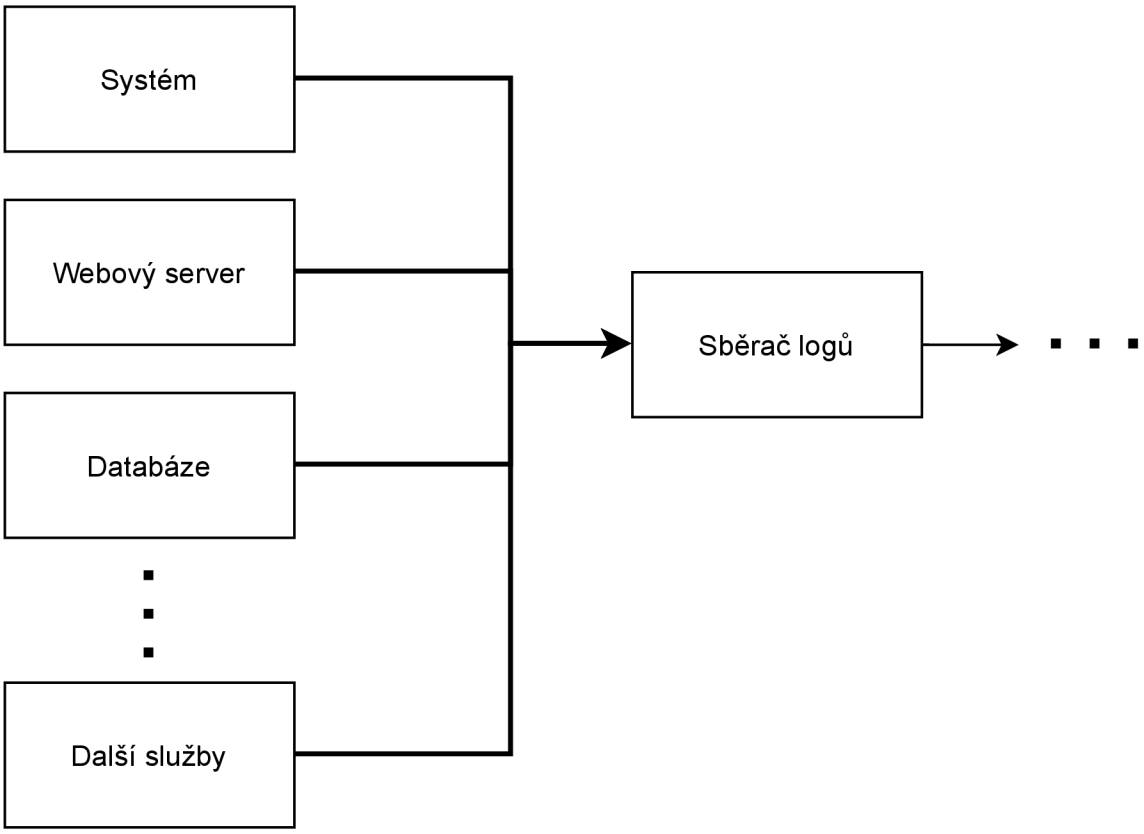
2.4 Sběr logů

Aby bylo možné logy zpracovávat a analyzovat, je potřeba tyto logy nejprve sesbírat a odeslat službě, která se o jejich zpracování a uchování postará. Prvním krokem v k dosažení tohoto cíle je logy získat z mnoha zdrojů a připravit je na přenos. V praxi se k tomuto účelu využívá hned několik způsobů sběru logů v závislosti na způsobu jaké možnosti daný zdroj logů nabízí.

Cíl služby, která se stará o sběr logů je znázorněn na obrázku 2.7, takovýchto služeb pak může běžet více a každá se může například specializovat na odlišný zdroj logů.

Pro účely sběru logů existuje velké množství nástrojů, některé se specializují na sběr logů z logových souborů, jiné se věnují například vzorkování využití hardware systému. Všechny tyto nástroje ale mají společný cíl, a to sesbírané logy převést do předem definovaného formátu (například JSON objektu) a odeslat je vybrané službě k dalšími zpracování.

Následující podsekce se věnují jednotlivým zdrojům logů a existujícím nástrojům, které je dokáží sbírat. Rozdělení do podsekcí je inspirováno existujícími nástroji, tak jak jsou jednotlivé zdroje logů ke sběru v těchto nástrojích rozděleny.



Obrázek 2.7: Práce sběrače logů.

Souborové logy

Za nejjednodušší, nejznámější a zároveň nejvíce podporované řešení se dá jednoznačně označit ukládání logů do souboru. Záznamy činnosti programu jsou touto cestou ukládány do zvoleného souboru, kde každý řádek představuje jeden záznam.

V kombinaci například se službou logrotate [20], která mimo jiné zajišťuje ořezávání délky těchto logů, tak aby jejich velikost nerostla až do zaplnění veškerého volného místa je pak i tento způsob zaznamenávání logů bez problému použitelný v praxi.

Logy tohoto druhu dokáže sbírat vícero existujících nástrojů, více o několika vybraných nástrojích v následujících podsekcích.

Filebeat

Filebeat zastává pozici jednoduché malé služby s jediným cílem – sbírat logy z textových souborů a odesílat je další službě. Po nainstalování na server Filebeat monitoruje souborové logy specifikované v konfiguraci a sbírá nové záznamy. Tyto záznamy jsou následně odeslány vybrané službě ke zpracování. Mezi podporované služby na které lze nové záznamy odesílat patří Elasticsearch 2.6, Logstash 2.5, Kafka 2.5 a mnoho dalších. Více o tomto nástroji lze nálzt na webových stránkách [12].

Azure Log Analytics agent

Za zmínění stojí i nástroje specifické pro cloudová řešení. Azure Log Analytics je nástroj, který se stará o sbírání logů z virtuálních strojů s OS Windows či Linux. Sesbírané logy odesílá do pracovního prostředí pro analýzu logů v nástrojí Azure Monitor. Azure Log Analytics mimo souborové logy dokáže sbírat i záznamy z Windows logu událostí, Syslogu 2.5, IIS Logů či historii využití zdrojů. Mimo klasické logy také dokáže sbírat informace o dalších Azure službách, jako například Azure Secuity Center či Azure Automation. Přehled informací o tomto nástroji je také dostupný na webové stránce [6].

Journald

Součástí známého open-source správce Linux systému systemd [34] je i nástroj journald, jehož hlavním úkolem je zaznamenávání logů systému, běžících služeb a jiných systémových událostí. Na rozdíl od souborových logů pracuje se záznamy systému jako s jedním celkem. Log vytvořený pomocí Journald využívá stejného formátu jako Syslog. Pro uživatelské pohodlí nabízí příkaz `journalctl` k prohlížení zaznamenaných událostí a jejich zjednodušené agregaci. Více informací o Journald je možné nalézt také na webové stránce [36].

Následující podsekce uvádí některé existující řešení ke sběru logů s Journald k jejich přenosu na jiný systém.

Journalbeat

Journalbeat, jak již název napovídá, má za cíl sbírat logy ze systemd journálu a odesílat je jiné vybrané službě. Po spuštění služby začne Journalbeat monitorovat cesty k logům zadané v konfiguraci a sbírat záznamy. Ty poté odesílá vybrané službě ke zpracování. Journalbeat dokáže odesílat logy do mnoha služeb, jako například Elasticsearch 2.6, Logstash 2.5 či Kafka 2.5. Pro více infromací o tomto nástroji lze také navštívit oficiální webové stránky [14]

Utilizace hardware

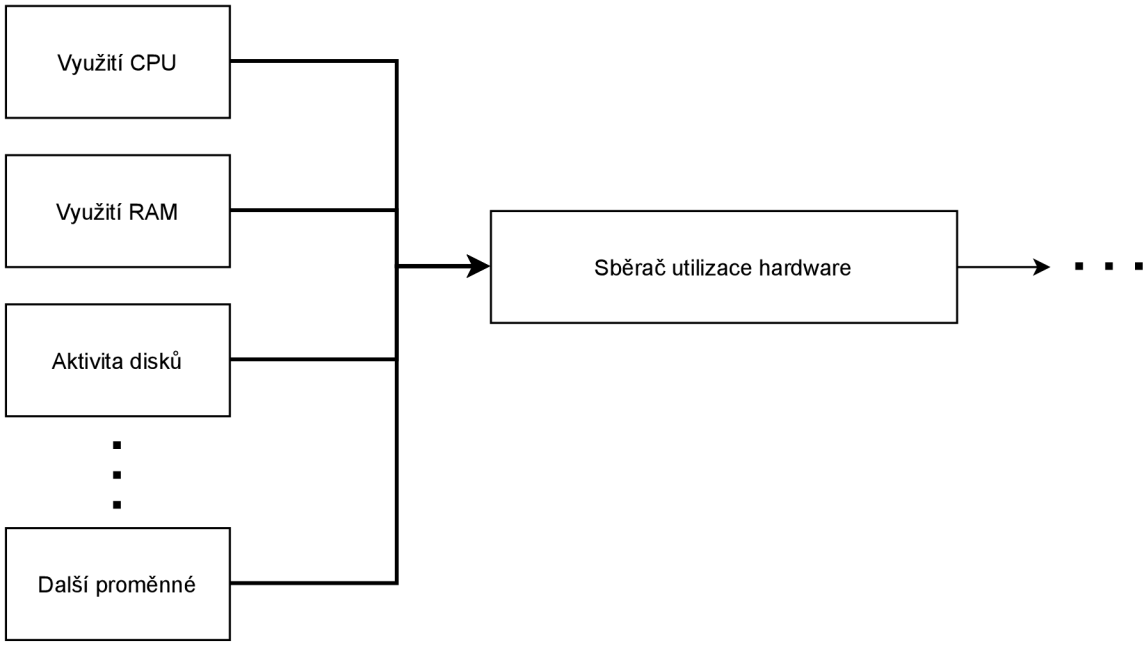
Podobně jako sběr jiných logů i sběr historie utilizace hardware může pomoci při snaze odhalit zdroj problémů. Na rozdíl od logů v běžném linuxovém systému není historie utilizace hardware zaznamenávána. Existují ale nástroje, které umí tuto funkcionalitu doplnit, většina těchto nástrojů ale zároveň i obstarává předávání sesbíraných dat jiné službě.

Cílem sběru dat o utilizaci hardware je zaznamenat průběh proměnných stavů systému v čase. Může se jednat o různé parametry, od zatížení procesoru, využití paměti RAM, až po aktivitu sítě či disků. Všechny tyto informace mohou hrát důležitou roli při hledání zdroje problémů. 2.8

Následující podsekce se věnují několika existujícím nástrojům ke sběru historie utilizace hardware. V praxi existuje i moho jiných nástrojů, kde každý takový nástroj má své cíle a vlastnosti, které jej dělají lepší volbou v kontrétních situacích.

Metricbeat

Ke sběru a následnému odesílání logů o utilizaci hardware, ale i jiných statistik spojených s ostatními službami běžícími na vybraném systému může velice dobře posloužit Metricbeat. Jakožto jeden z mnoha takzvaných Elastic Beatů, které vznikají po boku sady nástrojů Elastic stack.



Obrázek 2.8: Práce sběrače dat o utilizaci hardware.

Metricbeat si klade za cíl fungovat jako velice nenáročný nástroj k periodickému vytváření záznamů o stavu mnoha vybraných proměnných z operačního systému a služeb běžících na zařízení, na kterém sběr dat probíhá, jako například Apache, Nginx, MySQL, MongoDB, PostgreSQL, Zookeeper či Redis. Podobně jako ostatní Elastic Beaty umožňuje i Metricbeat odesílat vytvořené záznamy k dalšímu zpracování vice možným službám. Mezi podporované služby pro výstup logů patří Elasticsearch 2.6, Logstash 2.5, Kafka 2.5 a mnoho dalších. Více informací o tomto nástroji lze nálzt na webových stránkách [22].

Netdata

Nástroj Netdata se zaměřuje na sběr historie stavu mnoha proměnných ze systému, hardware, kontejnerů a služeb. Cílí na jednoduchost zprovoznění, kdy je možné nainstalovat a spustit Netdata i bez konfigurace a i přesto získat přístup k mnoha statistikám. Zdrojový kód tohoto nástroje je veřejně dostupný pod licencí GPLv3 na GitHub repozitáři [28]. Součástí nástroje Netdata je i webová aplikace k analýze a vizualizaci sesbíraných dat. Sesbíraná data Netdata agentem je možné odesílat na centrální instalaci Netdata, kde je možné prohlížet data z více strojů. Více informací o tomto nástroji lze nálzt na webových stránkách [27].

Monitorix

Monitorix je nástroj vyvinutý ke sledování sítě počítače. Tento nástroj periodicky sbírá informace o stavu systému a vyzualizuje tyto informace pomocí webového rozhraní. Mezi monitorované informace patří mnoho metrik popisujících výkon systém, s pomocí těchto informací tak může Monitorix pomoci s odhalením, co počítač zpomaluje, co v systému selhává, či jaké operace trvají nepřiměřeně dlouho. Zdrojové kódy tohoto nástroje jsou dostupné pod licencí GPLv2 na GitHub repozitáři [25].

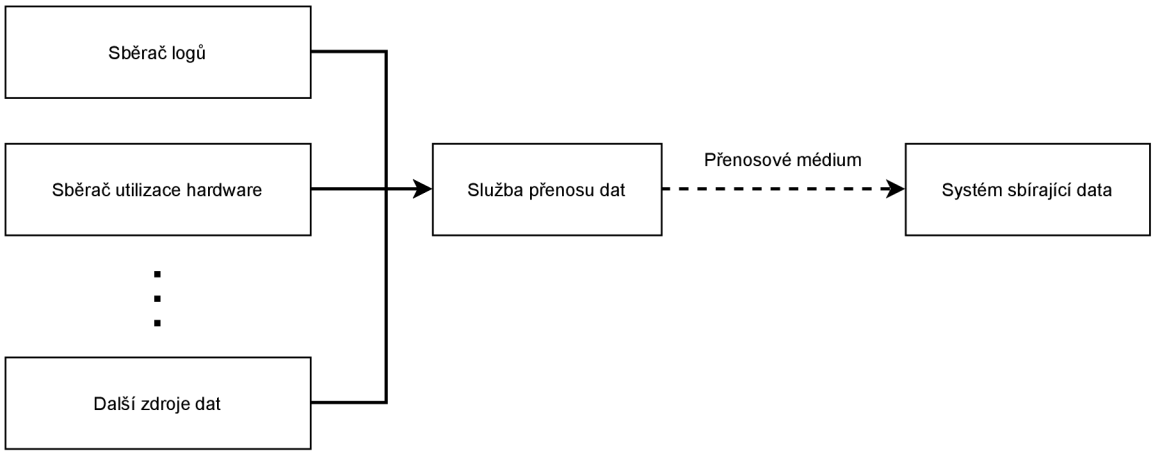
Tento nástroj je tvořen několika programy. První program je tvořen službou, která se stará o sběr dat a jejich uchovávání. Druhá část je pak tvořena webovou aplikací, která data zpracovává a vizualizuje uživatelům pomocí webového rozhraní. Více informací o tomto nástroji lze nálzt na webových stránkách [24] a [26].

2.5 Přenos sesbíraných dat

Aby bylo možné sledovat vícero systémů z jediného centrálního systému, sesbíraná data je potřeba přenést z těchto systémů na centrální systém. V praxi má tento problém více řešení. Některé služby na sběr logů dokáží i sesbíraná data přímo odesílat centrálnímu systému, jiné služby spoléhají na jiné způsoby doručení dat centrálnímu systému. Tato sekce se zaměřuje právě na situace, kdy sesbíraná data jsou na centrální systém doručována oddělenou službou.

Jak znázorňuje obrázek 2.9, software pro přenos sesbíraných dat má za cíl doručt všechna jemu předaná data centrálnímu systému ke zpracování a uchování k budoucí analýze.

Následující podsekce se věnují několika existujícím nástrojům, které se zabývají přenosem logů a jiných dat. V praxi je možné tyto nástroje kombinovat a řetězit k dosažení kýženého výsledku.



Obrázek 2.9: Práce služby přenosu dat.

Systemd netlogd

Systemd netlogd [35] je open-source nástroj, který přeposílá události, které byly zaznamenány nástrojem Journald, o němž více pojednává podsekce 2.4, na zadané IP adresy podle syslog 2.5 protokolu (RFC 5424). Tento nástroj podporuje odesílání událostí na unicastové i multicastové adresy. Události odeslané tímto nástrojem mohou být přijimány jakoukoliv službou, která podporuje příjem událostí ve formátu syslogu.

Syslog

Syslog narozdíl od ostatních nástrojů v této sekci zmíněných není přímo nástrojem, ale jedná se o standard k přenosu logů (RFC 5424). Díky tomuto standardu může více programů implementovat identický protokol a přeposílat přes něj záznamy logů, aniž by navzájem

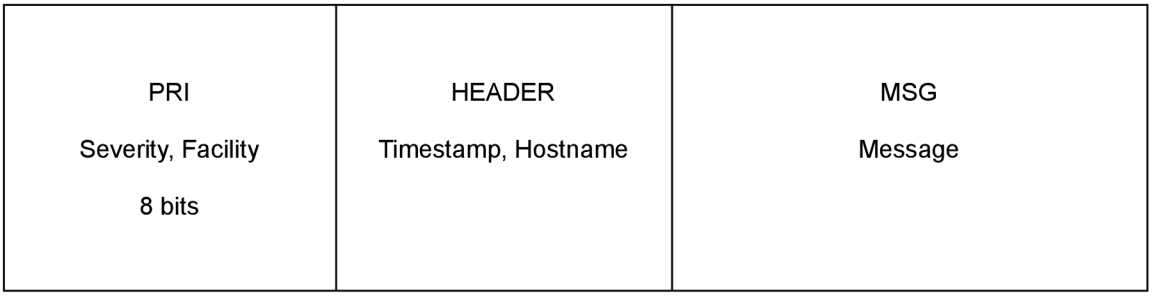
museli vědět o jaký konkrétní program se jedná. Více informací o tomto protokolu lze nálzt na webových stránkách [32].

Každá zpráva zaslaná ve formátu syslogu je označena takzvaným facility kódem, který určuje, jaký druh software danou zprávu vygeneroval. Za facility kódem následuje kód úrovně důležitosti zprávy. Pro názornost obrázek 2.10 ukazuje jak vypadá formát zprávy zaslané protokolem syslog.

Protokol syslogu lze využívat k vícero účelům. Při správě systému ke sběru logů, ke kontrolování bezpečnosti k odesálání zpráv o detekovaných událostech či při ladění různých softwarových problémů k přenášení zpráv z ladícího software.

Mnoho zařízení jako například tiskárny, síťové směrovače a přepínače využívají syslog standardu k zasílání svých logů, pokud jsou k tomu nakonfigurovány.

Implementací syslogu existuje mnoho v mnoha různých podobách, od integrovaných nástrojů v síťových zařízenách až po implementace sbírající systémové logy. K práci se syslog standardem existují například nástroje syslog-ng [33] a rsyslog [29].



Obrázek 2.10: Formát zprávy zaslané protokolem syslog.

Logstash

Logstash [21] je open-source nástroj, který sbírá data z vícero zdrojových systémů a před předáním dál je transformuje. Jeho cílem na rozdíl od ostatních nástrojů pro přenos sesbíraných dat v této sekci není pouze data doručit z jednoho programu jinému programu. Logstash nabízí mnoho možností jak přenášená data transformovat a jinak zpracovávat. Při transformování dat lze například doplňovat do dat polohu podle IP adresy, rozkládat a zjednodušovat strukturovaná data či anonymizovat data odebíráním citlivých informací.

Logy a jiná data jsou často vytvářena v rozdílných formátech, v takové podobě je ale těžké s nimi pracovat. Logstash v takovýchto situacích může pomoci data, která přes Logstash prochází, normalizovat a transformovat do podoby, kdy s nimi bude možné pracovat lépe.

Logstash je často využíván v kombinaci s Elasticsearch 2.6 a Kibana 2.7 jako plnohodnotný toolkit ke sběru, analýze a vizualizaci dat. Podporuje mnoho zdrojů dat, jako například Kafka 2.5 či Elastic Beats [8].

Apache Kafka

Apache Kafka [1] implementuje framework softwarovou linku využívající proudové zpracovávání dat, které jí procházejí. Apache Kafka je vyvíjena společností Apache Software Foundation a je šířena pod open-source licencí Apache 2.0. Cílem tohoto nástroje je nabídnout jednotnou platformu pro zpracovávání datových zdrojů v reálném čase. Apache

Kafka je optimalizována tak, aby jí mohlo procházet mnoho dat současně a aby procházející data měla jen minimální zpoždění přidané průchodem Apache Kafkou. Více informací o základech fungování nástroje Apache Kafka lze nálzt na webových stránkách [3] či [2].

Připojení k externím systémům zajišťuje Kafka Connect. Pomocí tohoto nástroje je možné do přidávat i číst data z Apache Kafky programy, které nebyly vytvořeny s podporou Apache Kafka.

Ke komunikaci Apache Kafka využívá protokolu TCP, přes který přenáší data v definované podobě. Při přenosu Apache Kafka pracuje se skupinami zpráv, které spojuje dohromady, čímž se snižuje náročnost jejich přenosu, tímto přístupem Apache Kafka vytváří větší současně zpracované bloky, větší přenesené pakety a větší sekvenční zápisy na disk. Díky tomu dokáže Apache Kafka při zátěži zpracovávat mnohem více průchozích dat, protože data jsou zpracována ve skupinách, místo po jedné zprávě.

Streamování dat pomocí Apache Kafky je možné přirovnat k práci lidského nervového systému. Při práci v prostředí distribuovaných systémů se Apache Kafka stará o předávání událostí mezi jednotlivými instancemi. V takovýchto prostředích je možné dosáhnout stavu, kdy systémy běží bez přestávky a je možné za běhu měnit a jinak řídit jejich běh.

Mimo mnohá jiná využití Apache Kafky v praxi je možné ji využít i k přenášení logů ze zdrojů do centrálního systému. Centrální systém při tomto přístupu nemusí být nakonfigurován pro každé zdrojové zařízení, stačí aby četl proud dat z Apache Kafky a aby zdrojová zařízení do tohoto proudu zapisavala sesbírané logy.

2.6 Uchování sesbíraných dat

Než je možné sesbíraná data analyzovat a jinak v ních vyhledávat, je potřeba tato data nejprve uložit a připravit k vyhledávání. Dosažení tohoto cíle většinou obnáší zapsání dat do různých databází a vytváření různých indexů napříč datay.

Pro názornost obrázek 2.11 ukazuje posloupnost operací s daty tak, aby bylo možné je uložit a následně analyzovat.

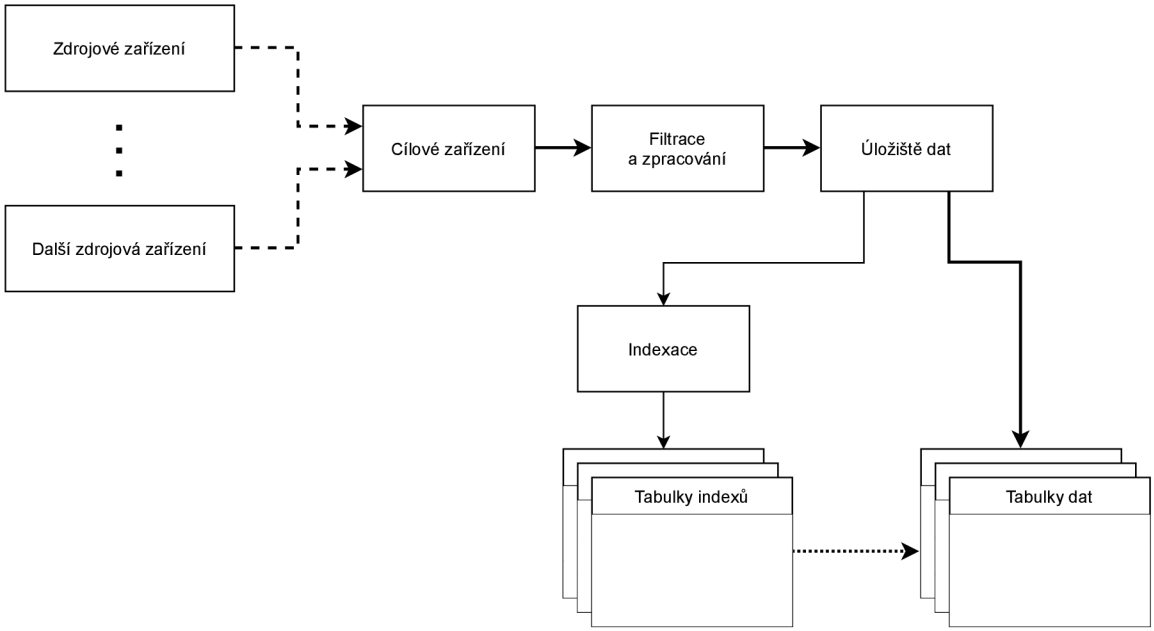
Tato sekce se věnuje existujícím řešením k uchovávání sesbíraných dat. V praxi existuje i mnoho řešení typu „vše v jednom“, která zahrnují nejen nástroje k uchování sesbíraných dat, ale i nástroje k jejich sběru, analýze a vizualizaci. Těmto řešením se ale tato sekce nevěnuje.

Elasticsearch

Jeden ze zástupců tohoto typu software je Elasticsearch. Jedná se o open-source nástroj nabízející distribuované textové vyhledávání v logách JSON dokumentů. K těmto účelům nabízí Elasticsearch REST API, pomocí něhož je možné popsat hledaná data či zadat Elasticsearchu jiné příkazy. Pro více infromací o tomto nástroji lze také navštívit oficiální webové stránky [9] nebo [10].

K práci s elastic search, mimo officiálního webového rozhraní Kibana 2.7, je možné i využít oficiálně podporovaných klientských knihoven pro jeden z vybraných jazyků, jako například Python, Java, .NET, PHP, Apache Groovy či Ruby.

Elasticsearch lze využívat k vyhledávání v mnoha různých dokumentech. Hledání pomocí Elasticsearch lze škálovat díky distribuovanému přístupu. Každá sada dat je rozdělena do oddělených částí a tyto části mnohou následně být replikovány mezi více instancemi Elasticsearchu. Každá instance Elasticsearchu spravuje několik částí dat a koordinuje operace



Obrázek 2.11: Průběh zpracování a uchování dat.

nad těmito částmi. K zachování rychlosti a spolehlivosti, Elasticsearch se automaticky stará o balancování a přesměrovávání dat, která spravuje.

Elasticsearch je velice často využíván v kombinaci s Logstash 2.5 a Kibana 2.7 jako plnohodnotný toolkit ke sběru, analýze a vizualizaci dat.

2.7 Vizualizace dat

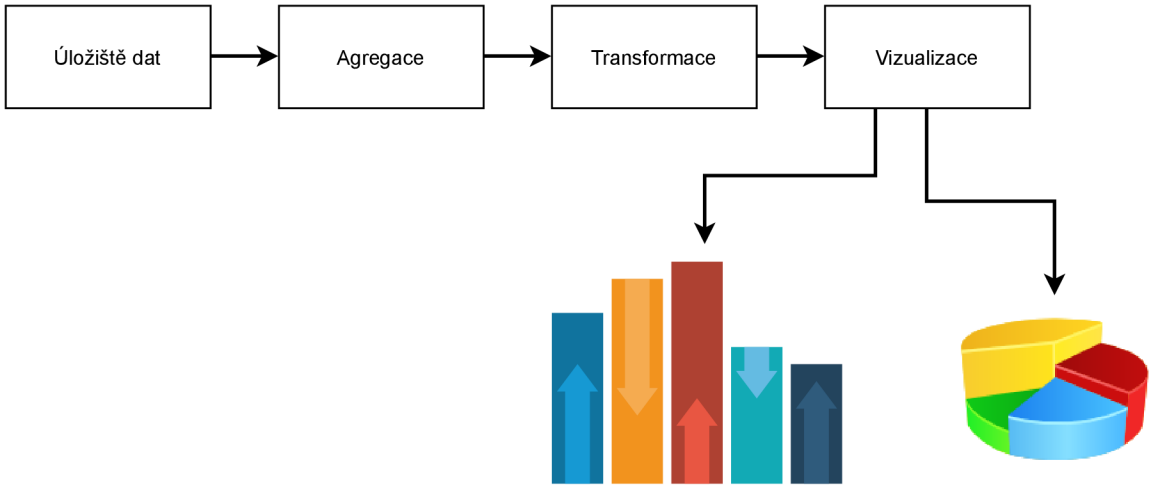
Posledním a zároveň cílovým krokem při práci se sesbíranými logy je jejich vizualizace. Cílem vizualizace je reprezentovat údálosti logů tak, aby byly čitelné pro člověka – tak aby z nich vyčetl co nejvíce informací. Toto je zejména důležité v situacích, kdy je k dispozici velké množství dat a není tak možné pro člověka se v takovýchto datech zorientovat pouhým pročítáním zaznamenaných dat.

Data k vizualizaci je potřeba před vizualizací agregovat a transformovat, tak aby odpovídala formátu, ve kterém mají být vyobrazena. Obrázek 2.12 znázorňuje postup přípravy dat k vizualizaci.

Tato sekce se věnuje existujícím řešením k vizualizaci sesbíraných dat. V praxi existují i řešení, která se kromě vizualizace dat zabávají i jich sběrem, analýzou a vizualizací. Těmto řešením se ale tato sekce nevěnuje.

Kibana

Jedním z známých nástrojů k vizualizaci dat je Kibana. Umožňuje vizualizovat zadaná data z Elasticsearch 2.6 do různých grafů, které lze seskupovat do přehledných panelů. Kibana nabízí vizualizace dat pomocí sloupcových, hranových, bodových a výsečových grafů či pomocí mapy k zjednodušení orientace v datech. V Kibaně je také možné vytvářet vlastní vizualiazace dat pomocí Canvasu, který umožňuje užiavtelům vytvářet vlastní reprezentace dat získaných z Elasticsearch.



Obrázek 2.12: Process vizualizace dat.

Kibana je velice často využívána v kombinaci s Logstash 2.5 a Elasticsearch 2.6 jako plnohodnotný toolkit ke sběru, analýze a vizualizaci dat. Pro více infromací o tomto nástroji lze také navštívit oficiální webové stránky [16] nebo [17].

Kapitola 3

Ochrana dat při přenosu

Software musí implementovat mnoho mechanismů které podporují zabezpečení systému. Ať už se jedná o ochranu dat či ochranu systému před napadením. O některé mechanismy se stará operační systém, zatímco o jiné se musí aplikace postarat sami. Zabezpečení systému se věnuje kniha [7] a rozebírá téma zabezpečení velice podrobně. Věnuje se podrobně i tématům probíraným v této kapitole.

Tato kapitola slouží jako zjednodušený úvod do tématu ochrany dat za využití asymetrické a symetrické kryptografie. Věnuje se také ve zkratce elektronickým podpisům, jakožto nástroji k ověření původu a celistvosti dat.

3.1 Základy zabezpečení dat

Tato sekce byla inspirována popisem základů počítačové bezpečnosti z knihy [30]. Více informací je také možné nalézt na webové stránce [13].

V jádru se cíl zabezpečení dat točí kolem několika základních atributů:

- Důvěrnost
- Celistvost
- Dostupnost

Důvěrnost představuje ochranu dat proti jejich čtení neoprávněnou stranou. Zajišťuje, že důvěrná data mohou být přečtena pouze vybranými jednotlivci.

Celistvost odpovídá ochraně dat proti neoprávněné modifikaci. Zajišťuje, aby data bylo možné modifikovat pouze specifikovaným způsobem, například vybraným programem na jendom z vybraných systémů.

Dostupnost se věnuje zajištění přístupu k datům a ochraně zdrojů dat, tak aby sloužily uživatelům a nebylo možné, aby někdo neoprávněný jejich dostupnost narušil.

Kromě výše zmíněných tří základních atributů existují i další méně známé, ale stále důležité atributy:

- Autentičnost
- Nepopiratelnost
- Autorizovanost

Autentičnost se stará o ověření že data pochází od daného subjektu, tím se zvyšuje důvěra ve validitu dat a zajišťuje se, že data pochází ze známého důvěryhodného zdroje.

Nepopiratelnost zajišťuje nemožnost subjektu popřít, že provedl v minulosti nějakou operaci. Jedná se o možnost ověřit tento fakt z dat, která byla od daného suběktu obdržena.

Autorizovanost určuje, jaké operace je daný subjekt schopný provádět nad daty podle jeho oprávnění.

3.2 Asymetrická kryptografie

Asymetrická kryptografie, též zvaná kryptografie s veřejným klíčem je druh kryptografických algoritmů, které využívají páru klíčů. Pár klíčů je tvořen veřejným a soukromým klíčem. Veřejný klíč je dostupný ostatním, zatímco soukromý klíč zná pouze jeho vlastník.

V prostředí, ve kterém každý má svůj soukromý klíč a zná veřejné klíče všech ostatních, je každý schopen zašifrovat zprávu pomocí veřejného klíče příjemce zprávy. Takovou zprávu pak může přečíst pouze příjemce i v případě, že by ji v zašifrované podobě zachytil i někdo jiný.

Asymetrická kryptografie ale nebývá většinou využívána o samotě. Často je kombinována se symetrickou kryptografií. Šifrovat data asymetricky je více náročné, nežli je symetrické šifrování. K vyřešení tohoto problému je možné nejprve vygenerovat náhodný klíč, který je následně zašifrován pomocí asymetrické kryptografie. Poté jsou data zašifrována pomocí sysmetrické kryptografie za použití vygenerovaného náhodného klíče. Takto zašifrovaná data je pak schopen dešifrovat pouze jejich příjemce.

Více o tomto tématu například pojednává webová stránka [4].

3.3 Symetrická kryptografie

Symetrická kryptografie, je druh kryptografických algoritmů, které využívají k šifrování i dešifrování dat stejný klíč. Algoritmy pro symetrickou kryptografii bývají v zásadě rychlejší, než algoritmy pro asymetrickou. Klíč u symetrické kryptografie představuje tajnou informaci, kterou musí znát jak tvůrce zprávy, tak její příjemce, v praxi je toto nejzásadnější problém, protože předávání klíčů není jednoduchý úkol.

Více o tomto tématu například pojednává webová stránka [31].

3.4 Elektronický podpis

Elektronický podpis představuje sekvenci dat, která jsou vázána na jiná data takovým způsobem, že je možné ověřit, že takovouto sekvenci vytvořil daný subjekt. Elektronický podpis by se dal přirovnat k ručnímu podpisu na papír, pouze adaptovanému pro svět počítačů.

Na straně příjemce podepsaných dat je možné ověřit, jaký subjekt data podepsal a tím stvrdil jejich správnost. Tím může být zajičtěna autentičnost, celistvost a nepopiratelnost dat.

Více o tomto tématu například pojednává webová stránka [11].

Kapitola 4

Návrh systému

Tato kapitola se věnuje návrhu systému k dlouhodobému monitorování stavu linuxových systémů umístěných v produkčním prostředí. Se zaměřením zejména na situace, kdy s danými systémy není možné živě komunikovat a sběr dat tak musí probíhat zcela automaticky.

Data z takovýchto systémů musí být možné stahovat manuálně v případě potřeby a musí být možné taková data importovat do daného odděleného systému, kde budou uživatelům dostupná k analýze.

Pro zajištění bezpečnosti dat musí být znemožněno komukoliv prohlížet data, která byla získána manuálním stažením ze zařízení. Mohou být čitelná pouze pro systém pro shromažďování a vizualizace dat při importu.

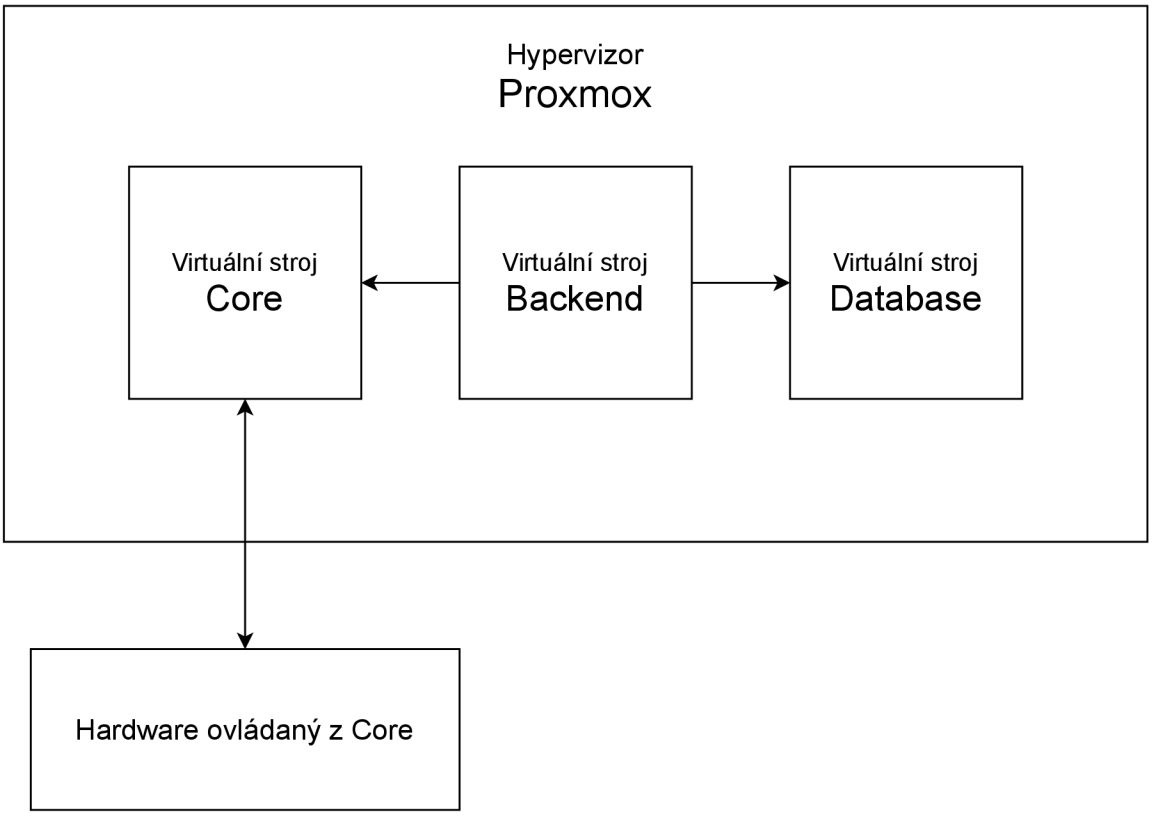
Navrhovaný systém pomůže firmě, která poskytuje zákazníkům hotové systémy k montážním linkám s několika různými zaměřeními. Z důvodů zvyšujícího se počtu zakázek firma cílí na zjednodušení řešení problémů s dodanými stroji k zákazníkům. Z důvodů rozdílných podmínek u mnoha zákazníků je nutné, aby výsledné řešení fungovalo zcela odděleně od firemní infrastruktury a nepotřebovalo ke své funkčnosti komunikovat s jiným systémem.

4.1 Prostředí navrhovaného systému

Základem pro navrhovaný systém je stroj pro zákazníka, který má definovanou vnitřní strukturu. Jedná se o počítač vybavený x64 procesorem. Na tomto počítači běží hypervizor Proxmox, který slouží k rozdělení prostředků nezi několik oddělených částí systému. Každá část systému běží v odděleném virtuálním stroji. Počítač je také vybaven výpočetním modulem (například grafickou kartou) a komunikačním rozhraním, pomocí kterého komunikuje s ostatním hardware, který v rámci výrobní linky ovládá. Tento hardware je pomocí IOMMU a PCI-E passthrough předán virtuálnímu stroji, který jej následně využívá. Na obrázku 4.1 je model tohoto systému znázorněn pro jednodušší představu umístění každé části v systému.

Systém je v dnešní podobě složen ze tří virtuálních strojů:

- Core – Jádro systému, komunikuje s hardware, zajišťuje kritické funkce,
- Backend – Služba obstarávající webové rozhraní ke správě a validaci činnosti jádra systému,
- Databáze – Databáze pro Backend uchovávající data o činnosti Core spravovaná Backendem.



Obrázek 4.1: Model systému pro zákazníka.

Pro shromažďování a vizualizaci dat z těchto strojů bude sloužit oddělený systém běžící ve firemní infrastruktuře. Tento systém nebude příliš speciální, pouze bude mít úložiště pro nashromážděná data a bude vybaven větším množstvím operační paměti a výpočetního výkonu, tak aby byl schopný po něm požadované služby zajišťovat.

4.2 Cíl řešení

Cílem navrhovaného systému je běžet na systému představeném v předchozí kapitole a monitorovat jeho stav zejména v souvislosti se zatížením systému a pomoci tak odhalit, jaké monitorované vlastnosti nejvíce brzdí běh systému.

Data sesbíraná monitorováním cílového systému bude možné analyzovat a vizualizovat na stroji ve firemní infrastruktuře.

Výsledný systém by měl být modulární, tak aby jej bylo možné v budoucnoti jednoduše rozšířit. Návrh počítá se třemy částmi systému – systémem generujícím záznamy, systémem k uchování dat před přenosem a systémem ke shromažďování a vizualizaci dat.

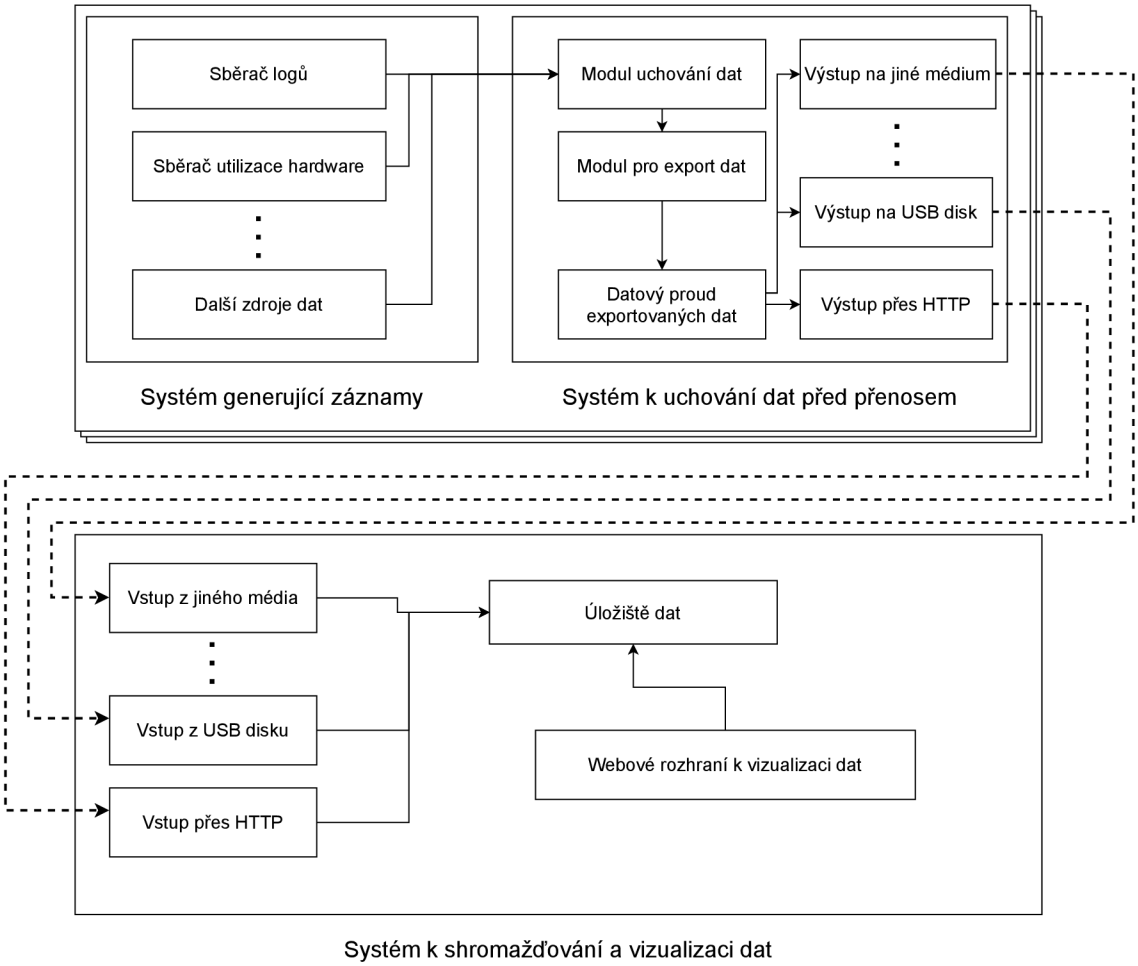
V následujících sekcích je nastíněno, co bude úkolem každé části systému i jednotlivých modulů, ze kterých jsou složeny.

4.3 Architektura

Následující diagram 4.2 znázorňuje vazby jednotlivých modulů systému.

Na monitorovaných zařízeních bude zajišťovat sběr dat sada modulů, které budou následně předávat sesbírané logy modulu pro uchovávání logů na zdrojovém zařízení. S modulem pro uchovávání logů bude komunikovat modul pro export dat. Vyvoláním exportu dat vznikne datový proud který bude následně zapsán na zvolené médium.

Na zařízení pro sbírání a vizualizaci dat lze získaná data importovat také více způsoby. Podle zvoleného způsobu jsou data předána modulu pro import dat, například zapojením USB disku s daty do USB portu na zařízení pro sbírání a vizualizaci dat a zvolením, která data z tohoto disku importovat pomocí UI či spuštěním scriptu v terminálu. Data jsou následně importována do úložiště dat, odkud mohou být vizualizována pomocí webového rozhraní.



Obrázek 4.2: Architektura systému.

4.4 Systém generující záznamy

Jedná se o sadu modulů, které mají za cíl sbírat různorodá data a logy ze systému na kterém běží. Tato data jsou následně předávána systému k uchování dat před přenosem, který může běžet i vedle tohoto systému.

Sběr by se měl zaměřit na generování logů o utilizaci hardware, sběr souborových i jiných logů ze služeb a systému, a generování auditních událostí.

Generování logů o utilizaci hardware zahrnuje statistiky využití procesoru, RAM a disků systémem, ale i vybranými službami. Události s aktuálním stavem jsou periodicky generovány a zasílány systému pro k uchování dat před přenosem.

Sběr logů ze služeb a systému předává nové záznamy vytvořené ve vybraných souborech systému k uchování dat před přenosem.

Auditní události by měly být generované tímto systémem pro různé změny stavu systému. Například úpravy souborů ve vybraných adresářích, či spuštění programu, který nepatří k žádné službě.

4.5 Systém k uchování dat před přenosem

Tento systém má za cíl přijímat sesbíraná data a logy z jednotlivých modulů systému pro generování záznamů a persistentně je uchovávat po zadanou dobu, či do zadaného limitu velikosti. V případě potřeby následně umožňuje tato data exportovat na externí zařízení, které může následně být přeneseno a připojeno k systému k shromažďování a vizualizaci dat a data mohou být importována z něj. Exportovaná data musí být chráněna proti čtení kýmkoliv kromě systému k shromažďování a vizualizaci dat.

USB

Jedním ze způsobů exportu logů je USB úložiště. Po připojení USB úložiště je možné jej vybrat k exportu. Modul v takovém případě zajistí vytvoření souboru s exportovanými daty na vybraném zařízení a zahájí přenos historie logů. Stav exportu bude možné kontrolovat přes API/UI aplikace.

Síť

Přes webovou aplikaci je možné vytvořit export logů a stáhnout jej. Exportovaná data jsou v proudu přenášena přes síť během vytváření, a není tak nutné expotrovaná data ukládat do dočasného souboru před stažením. Webová aplikace vyžaduje přihlášení předem zadaným účtem, než je umožněn export dat.

4.6 Systém k shromažďování a vizualizaci dat

Tento systém zajišťuje import logů exportovaných systémem k uchování dat před přenosem a zajišťuje jejich zpracování a persistentní uchování. Následně nabízí přístup k těmto logům pomocí API či UI k analýze a vyzualizaci.

Import logů

Soubor získaný exportem logů je možné importovat přes webové rozhraní. Uživatel může zvolit k importu jakékoliv externí úložiště připojené přes USB či zvolit exportovaný soubor ve svém počítači.

Zpracování logů

Aplikační logika na počítači, na kterém běží systém k shromažďování a vizualizaci dat, nabízí různé způsoby, jak logy automatizovaně zpracovávat. V situacích v budoucnosti, kdy tento systém bude v různých verzích umístěn na mnoha zařízeních u zákazníků, tak bude možné importovaná data normalizovat podle verze systému, ze kterého pocházejí.

Analýza a vizualizace logů

Pomocí UI si uživatel může vytvořit různé interpretace logů, různorodé grafické, tabulkové i jiné podoby. Výsledné interpretace logů a utilizace hardware je možné zobrazit ve webové aplikaci. Se sesbíranými daty je možné i pracovat pomocí API z jiných aplikací.

4.7 Případy užití

Systém bude mít tři druhy uživatelů – Admninistrátor systému, Analytik a Nosiče dat. Každý z těchto uživatelů může různými způsoby interagovat s různými částmi tohoto systému.

Systém generující záznamy

Tento systém je jedinou částí, která nevyžaduje téměř žádnou interakci s uživatateli. Po nakonfigurování této části systému administrátorem by měl tento systém fungovat zcela automaticky.

Systém k uchování dat před přenosem

S touto částí systému bude pracovat převážně Nosič dat, který bude mít po přihlášení možnost exportovat data na svůj počítač či externí USB úložiště. V rámci zprovozňování systému bude Administrátor mít možnost konfigurovat tento systém za cílem změnit přihlašovací údaje uživatele či získání veřejného klíče k ověřování podepsaných exportovaných dat. Na obrázku 4.3 jsou tyto vazby zakresleny do diagramu případů užití.

Systém k shromažďování a vizualizaci dat

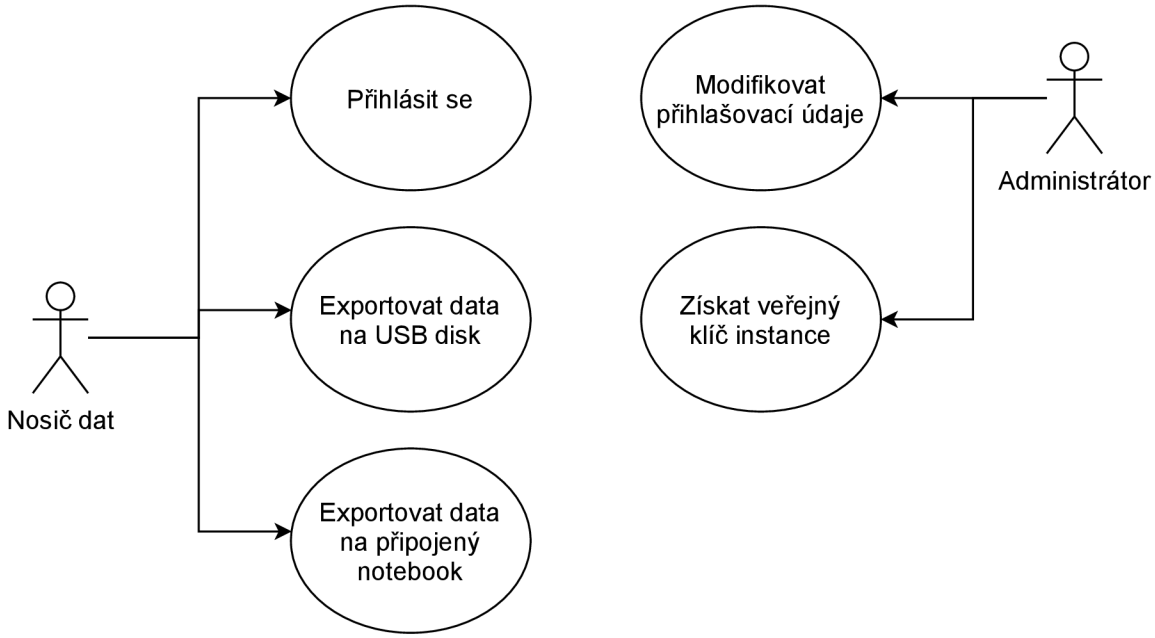
Tato část bude nejvíce interaktivní částí systému. Všechny tři druhy uživatelů zde mají k dispozici určité interakce. Na obrázku 4.4 jsou tyto vazby zakresleny do diagramu případů užití.

Nosič dat bude mít možnost se přihlásit k modulu pro import dat a následně ze svého počítače či externího USB disku importovat data, která získal exportováním dat ze jednotlivých zařízení, na nichž běží systém k uchování dat před přenosem.

Analytik jako jediný pracuje pouze s touto částí systému. Jeho interakce zahrnuje možnost přístupu k IU, ale i API za cílem analyzovat data, která byla do rozhraní importována. Po přihlášení by měl mít několik možností, jak s daty pracovat.

V rámci zjednodušení práce s daty by měl mít Analytik možnost konfigurovat indexaci dat – tak, aby data byla optimálně indexována, ke zrychlení práce s nimi.

K seznámení se s daty bude mít Analytik možnost procházet sesbíraná data. Tato data bude možné i filtrovat za cílem najít určitý druh dat, která následně mohou být využita k analýze a vytváření různých vizualizací.



Obrázek 4.3: Diagram případů užití systému pro k uchování dat před přenosem.

Za cílem zjednodušení hledání důležitých informací v datech bude mít Analytik možnost data agregovat a tato agregovaná data vizualizovat. Vizualizace dat bude možné ukládat pro budoucí použití a vytvářet z nich vizualizační panely tvořené sadou vizualizací zaměřených na konkrétní druh dat.

Administrátor bude mít v této části sytému na starosti doplňování veřejných klíčů pro ověřování podpisů z nových systémů k uchování dat před přenosem. Bude mít také možnost vytvářet a upravovat profily analytiků v modulu k analýze a vizualizaci dat. U modulu k importování dat bude mít možnost měnit přihlašovací údaje pro nosiče dat.

4.8 Datové modely a struktury

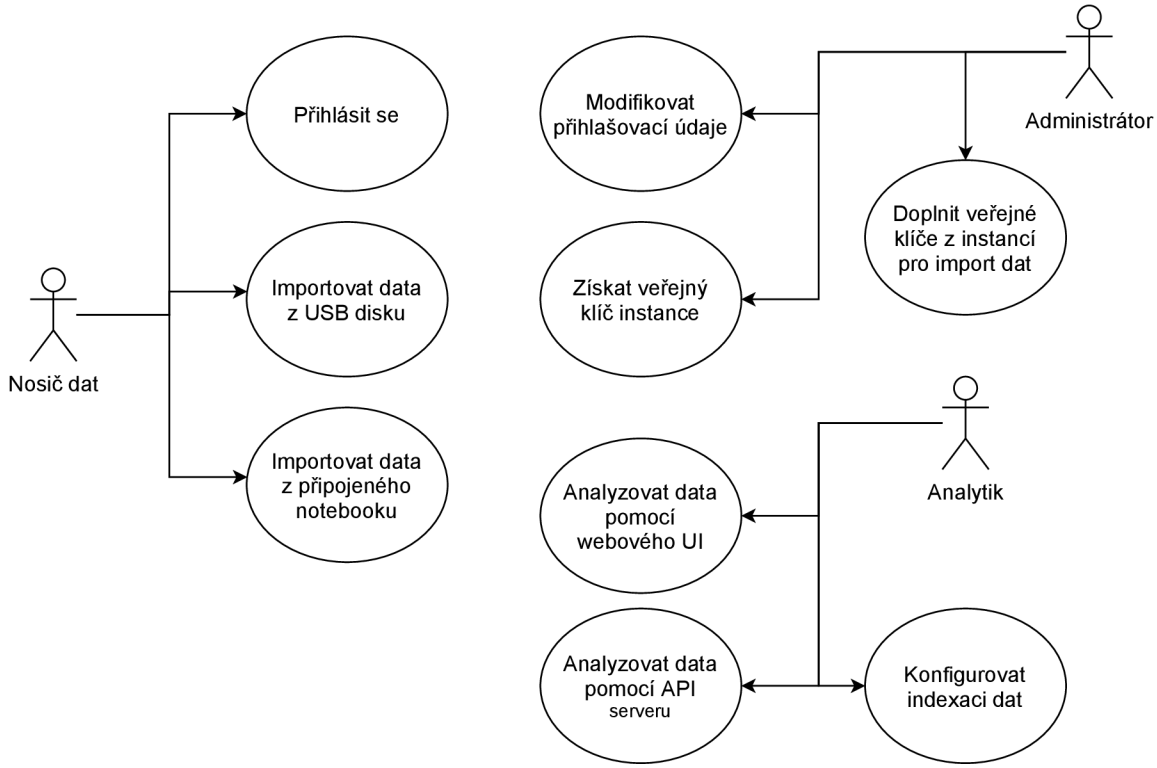
Připravovaný systém bude potřebovat ke svému fungování několik datových struktur.

Moduly pro sběr dat

Každý modul bude možné konfigurovat pomocí konfiguračního souboru. Touto cestou bude možné upřesnit cíle ke sbírání a určit kde lze nalézt API modulu pro uchování dat, kterému mají být logy zasílány.

Modul pro uchovávání dat

K uchovávání zaznamenané historie logů z monitorovaných zařízení před exportem této historie, je nutné použít persistentní úložiště. Jednotlivé záznamy budou uchovávány ve struktuře obdobné databázové tabulky, do které lze přidávat nové řádky se zvyšujícím se sekvenčím číslem.



Obrázek 4.4: Diagram případů užití systému k shromažďování a vizualizaci dat.

Každá monitorovaná oblast systému bude mít vlastní log, do kterého budou zapisovány všechny logu obdržené z modulu, který tyto logy sbírá. Limit množství uchovávaných logů bude konfigurovatelný.

Modul pro export dat

Tento modul bude mít vlastní konfiguraci, ve které bude možné nastavit některé parametry modulu. Bude možné měnit parametry webového serveru, sloužícího webové aplikaci. Nastavení také bude obsahovat seznam logů, které bude možné exportovat z webového rozhraní. Ke správnému fungování ochrany exportovaných souborů bude v konfiguraci specifikován veřejný klíč zařízení k shromažďování a vizualizaci dat pro šifrování exportovaných dat a soukromý klíč tohoto zařízení k uchování dat před přenosem pro podepsání exportovaných dat. Pro účely ověřování podpisu dat bude také v konfiguraci možné nastavit název tohoto systému. Podle tohoto názvu následně modul pro import dat identifikuje správný klíč, podle kterého ověřit podpis dat. V konfiguraci také bude možné volitelně nastavit přihlašovací údaje pro uživatele, který by chtěl provést export dat. V případě, že nejsou přihlašovací údaje uvedeny v konfiguraci, provést export dat je schopný kdokoliv.

Modul pro import dat

Konfigurace tohoto modulu bude obsahovat také různé parametry ke konfiguraci tohoto modulu. Podobně jako u modulu pro export dat bude možné měnit parametry webového serveru, sloužícího webové aplikaci. Také bude v konfiguraci možné volitelně nastavit přihlašovací údaje pro uživatele, který by chtěl provést import dat. V případě, že nejsou při-

hlašovací údaje uvedeny v konfiguraci, provést import dat je schopný kdokoliv. V nastavení také bude možné nastavit cestu ke složce pro dočasné soubory. Do této složky bude modul ukládat exportovaná data v dešifrované podobě před ověřením správnosti jejich podpisu. Ke správnému fungování importování dat, která byla zašifrována během jejich exportu, bude v konfiguraci modulu také uveden soukromý klíč k dešifrování exportovaných dat.

Modul také bude pracovat se složkou, která bude obsahovat veřejné klíče ze všech zařízení pro sbírání záznamů k uchování před přenosem. Názvy souborů těchto klíčů budou odpovídat názvům systémů, ze kterých pocházejí, a to ve formátu `<název_systému>.pem`. Při importu modul podle názvu systému, ze kterého exportovaná data pocházejí, vyhledá v této složce správný veřejný klíč k tomuto systému a ověří jím podpis dešifrovaných dat před započetím importu.

Kapitola 5

Implementace a nasazení systému

Tato kapitola se věnuje implementaci systému navrženého v předchozí kapitole. Popisuje také, jaké nástroje a existující řešení byly využity při implementaci a jejich výhody oproti ostatním řešením. Po srovnání existujících řešení se jednotlivé sekce této kapitoly nejprve zaměřují na architekturu jednotlivých částí a následné podsekce se věnují modulům, ze kterých byla daná část složena. Poslední sekce této kapitoly pak věnuje zvýšenou pozornost nástroji pro import a export dat, který byl pro účely této práce vytvořen.

5.1 Volba existujících řešení

V rámci průzkumu existujících řešení jsem zjistil, že pro velkou část navrženého řešení lze využít již existujících řešení. Rozhodl jsem se proto projít několik kandidátů a nalézt mezi nimi FOSS (Free and Open-Source Software[1]) řešení, které by nejlépe vyhovovalo způsobům a cílům využití výsledného systému.

Analýza a vyzualizace sesbíraných logů

Nejvíce definující podmínkou se brzy ukázala být potřeba, že musí být možné sesbíraná data přenášet ze sledovaných systémů na systém k schromažďování a vizualizaci dat bez možnosti využít k tomuto účelu síťového spojení. Převážná většina existujících řešení ke sběru a analýze dat počítá se síťovým spojením mezi všemi zařízeními. K vyřešení tohoto problému jsem se rozhodl využít taková existující řešení, která jednotlivé činnosti sběru, uchování a analýzy umožňují rozdělit do oddělených modulů a najít mezi těmito řešeními takové, u kterého bude možné sesbíraná data zapsat do souboru a poté importovat z tohoto souuboru do jiného zařízení.

Existujících řešení, která splňují výše uvedené podmínky, nebylo mnoho. Protokol syslog, sada nástrojů ELK Stack a pár dalších drobných nástrojů splňovalo tyto podmínky. Existující řešení jako například Sumo Logic, Splunk a další sice nabízí moho funkcí, ale většinou nenabízí potřebné funkce ve FOSS verzi nebo takovou verzi vůbec nenabízí.

Z existujících řešení jsem se proto nakonec rozhodl využít sady nástrojů ELK Stacku k analýze a vizualizaci dat. ELK Stack se řadí mezi nejpoužívanější nástroje ve svém odvětví ke sběru a analýze dat, v kombinaci s modularitou celého řešení ELK Stacku, která nabízí dynamičnost práce s daty, která je potřeba pro toto řešení, byl ELK Stack solidní volbou.

[1] Software který může být označen současně jako svobodný otevřený. Takový software může kdokoliv volně používat, kopírovat, studovat a měnit jakýmkoliv způsobem. Zdrojové kódy takového software jsou veřejně přístupné komukoliv.

Sběr logů z monitorovaných systémů

Ke sběru dat ze zdrojových zařízení jsem se rozhodl využít Elastic Beatů, které jsou vyvíjeny po boku s ELK Stackem a nabízí proto nejlepší možnosti kompatibility s nástroji ELK Stacku. Rozhodl jsem se sbírat textové logy aplikací, logy ze systémového žurnálu, historii utilizace hardware a pár druhů auditních událostí k zjednodušení detekce případných průniků do systému. Pro sběr všech těcho logů mají Elastic Beats již existující řešení. Sběr textových logů zajišťuje Filebeat 2.4, pro logy ze systémového journálu existuje Journalbeat 2.4, historii utilizace hardware dokáže sbírat Metricbeat 2.4 a o sledování systému a sběr auditních událostí se stará Auditbeat [5].

Přenost dat z monitorovaných systémů na systém ke shromažďování a vizualizaci dat

Ke zprovoznění přenosu dat z monitorovaných systémů na systém ke shromažďování a vizualizaci dat bylo potřebné nalézt řešení, které by umožňovalo uchovávat sesbíraná data na systémech k uchování dat před přenosem a následně by umožňovalo data z těchto sytémů exportovat do souboru a importovat do systému k analýze a vizualizaci dat. Rozhodujícím faktorem také bylo využití prostředků na zdrojových zařízeních. Možnost spustit ELK Stack na těchto zařízeních, sbírat data přímo do Elasticsearch 2.6 přes Logstash 2.5 a exportovat data z Elasticsearch se může z prvu zdát nejlepší, ale kvůli náročnosti nástroje Elasticsearch na systémovou paměť není pro aktuální prostředí použitelná.

Elastic Beaty dokáží sesbíraná data odesílat i jiným službám. Rozhodl jsem se proto prozkoumat i tyto služby, zda by některé z nich nemohly problém přenosu sesbíraných dat ze systému pro uchování dat na systém pro analýzu a vizualizaci dat bez přímého síťového spojení vyřešit.

Elastic beaty podporují výstup na standardní výstup či přímo do souborů. Prvním možným řešením proto bylo nechat Elastic Beaty sbírat data do souborů a tyto soubory následně přenášet a importovat. Ukázalo se ale, že tento přístup má své nedostatky. Například při exportu by bylo nutné implementovat určitý způsob zachovávání poslední přečtené pozice souborů. I přes tyto nedostatky se ale jedná o proveditelné řešení.

Další službou, do které je možné data z Elastic Beatů odesílat je Redis. K předávání dat v rámci sítě systémů by mohla být tato služba užitečná, ale k uchovávání logů se vzhledem ke svému způsobu prace s daty nezdála dobrou volbou v této situaci.

Poslední službou, které se tato podsekce věnuje a do níž Elastic Beaty dokáží odesílat svůj výstup, je Apache Kafka 2.5. Tato služba je ve většině prostředí využívána k přeposílání udalostí mezi programy. Podobně jako sběrnice operačního systému, ale pracující na úrovní celé sítě systémů, místo klasické sběrnice v rámci jediného systému. Z tohoto úhlu pohledu se může zdát tato služba nevhodnou k řešení problému uchování dat a přenosu dat ze systému pro uchování dat na systém pro analýzu a vizualizaci dat bez přímého síťového spojení, ale po delším průzkumu se ukázalo, že z výše zmíněných služeb je pravděpodobně tou nejvhodnější. Apache Kafku je možné nakonfigurovat, aby data, která jí prochází, uchovávala po dlouhou dobu a i při velém množství uložených dat se náročnost Apache Kafky nezvyšuje. Při čtení dat z Apache Kafky je pak možné přesně specifikovat, odkud se data mají číst – od začátku, od poslední čtené pozice, od konce. Kombinace těchto vlastností umožňuje relativně jednoduše vytvořit nástroj, který by dokázal vybraná data z Apache Kafky exportovat do jediného souboru a následně do jiné instance Apache Kafky je importovat z tohoto souboru. Rozhodl jsem se proto Apache Kafku využít k vyplnění velké části

posledního problému a poohlédnout se po existujícím nástroji, který by dokázal exportovat a importovat data z Apache Kafky.

Import a export dat z Apache Kafka

Ukázalo se, že existujících řešení tohoto problému není mnoho. Některé nástroje se specializují na propojení přenost logů jedné Apache Kafka sítě do jiné. Mnoho nástrojů se věnuje napojení existujících zdrojů logů a dat do Apache Kafky. Některé návody se věnují přenesení dat z Apache Kafky na jiné zařízení zkopírováním datových souborů Apache Kafky. Žádné řešení ale nenabízelo vhodné funkcionalitu, která byla potřebná k vyřešení problému exportu dat z Apache Kafka do souboru a následnému importu do jiné instance Apache Kafka.

Nástroj Kafkacat [15], se chvíli zdál být řešením, které by bylo možné využít ke zjednodušení problému. Pomocí tohoto nástroje je možné přečíst všechny logy z Apache Kafky do textové podoby a následně je z této podoby opět zapsat do jiné instance Apache Kafky, jak je ukázáno v tomto blogu [23], stále by ale bylo nutné kolem tohoto nástroje vytvořit software, který by umožňoval export a import dat spouštět a zajišťoval by patřičnou ochranu exportovaných dat.

Rozhodl jsem se proto vytvořit software v podobě webové aplikace a backend serveru, který by umožňoval uživateli spouštět exporty a importy dat z Apache Kafky. Později, při průzkumu existujících knihoven k vytvoření tohoto software, kterému se věnuje sekce 5.5, jsem se rozhodl, že čtení zápis dat do Apache Kafka implementuji pomocí Python knihovny pro komunikaci s Apache Kafkou, místo původně plánovaného nástroje Kafkacat. Využití Python knihovny mi umožnilo jednodušší integraci řešení se zbytkem kódu backendu a usnadnilo případná rozšíření funkčnosti, protože software není limitován dovednostmi nástroje Kafkacat.

5.2 Architektura systému generujícího záznamy

Sběr textových logů

Pro sběr textových logů na monitorovaných systémech byl zvolen nástroj Filebeat. Více informací o tomto nástroji a jeho cílech lze nalézt v kapitole 2.4.

Nástroj Filebeat bude na cílový systém nainstalován společně s ostatními Elastic Beaty pomocí instalačního skriptu `install-beats.sh` přiloženého na paměťovém médiu k této práci.

Ke sběru logů Filebeat využívá modulu `system` ve výchozí konfiguraci. V praxi bude možné jej také nakonfigurovat ke sběru textových logů konkrétních služeb, ale z důvodů technických limitací při testování tohoto řešení na reálných datech, kdy byl využit testovací systém, na kterém nejsou v současné době pravidla umístění logových souborů definována, byl sběr těchto logů v rámci této práce vynechán.

Filebeat je nakonfigurován, aby ke všem sesbíraným záznamům doplňoval název nasazené instance a stroje, na kterém běží. Tyto informace jsou zapsány během instalace do konfigurace pomocí definovatelných vlastních polí, která má Filebeat doplňovat ke každému vytvořenému záznamu.

Výstup z Filebeatu je nasměrován do Apache Kafka 2.5 na zadanou IP adresu systému uchovávání dat před přenosem a je zapisován do topicu s názvem `filebeat`.

Sběr logů ze systémového žurnálu

Pro sběr logů ze systémového žurnálu – Journald 2.4 na OS GNU/Linux se Systemd, na monitorovaných systémech byl zvolen nástroj Journald. Více informací o tomto nástroji a jeho cílech lze nalézt v kapitole 2.4.

Nástroj Journalbeat bude na cílový systém nainstalován společně s ostatnímy Elastic Beaty pomocí instalačního skriptu `install-beats.sh` přiloženého na paměťovém médiu k této práci.

Ke sběru logů Journalbeat nevyžaduje mnoho konfigurace a proto pro jednoduchý sběr logů nebylo nutné výchozí konfiguraci sběru nijak modifikovat. Dle výchozí konfigurace je Journalbeat nastaven aby sbíral záznamy ze systémového žurnálu umístěného ve výchozím úložišti Journald.

Journalbeat je nakonfigurován, aby ke všem sesbíraným záznamům doplňoval název nasazené instance a stroje, na kterém běží. Tyto informace jsou zapsány během instalace do konfigurace pomocí definovatelných vlastních polí, která má Journalbeat doplňovat ke každému vytvořenému záznamu.

Výstup z Journalbeatu je nasměrován do Apache Kafka 2.5 na zadanou IP adresu systému uchovávání dat před přenosem a je zapisován do topicu s názvem `journalbeat`.

Sběr metrik využití hardware a jiných prostředků

Pro sběr historie utilizace hardware na monitorovaných systémech byl zvolen nástroj Metricbeat. Více informací o tomto nástroji a jeho cílech lze nalézt v kapitole 2.4.

Nástroj Metricbeat bude na cílový systém nainstalován společně s ostatnímy Elastic Beaty pomocí instalačního skriptu `install-beats.sh` přiloženého na paměťovém médiu k této práci.

Ke sběru logů Metricbeat využívá modulů `system` a `linux`. Modul `system` je nakonfigurován, aby každé 2 minuty oznamoval aktuální využití procesoru, systémové paměti, zatížení sítě a systému. Také oznamuje každé 2 minuty stav běžících služeb, procesů a přihlášených uživatelů. S delší periodou 5 minut tento modul oznamuje statistiky využití filesystémů a zaplnění volného uložiště na nich. Modul `linux` se stará o sběr zatížení disků a délkách čekání na zápis na disk ve 2 minutových intervalech.

Metricbeat je nakonfigurován, aby ke všem sesbíraným záznamům doplňoval název nasazené instance a stroje, na kterém běží. Tyto informace jsou zapsány během instalace do konfigurace pomocí definovatelných vlastních polí, která má Metricbeat doplňovat ke každému vytvořenému záznamu.

Výstup z Metricbeatu je nasměrován do Apache Kafka 2.5 na zadanou IP adresu systému uchovávání dat před přenosem a je zapisován do topicu s názvem `metricbeat`.

Generování událostí dle předepsaných pravidel

Pro sběr auditních událostí na monitorovaných systémech byl zvolen nástroj Auditbeat. Více informací o tomto nástroji a jeho cílech lze nalézt v na webové stránce [5].

Nástroj Auditbeat bude na cílový systém nainstalován společně s ostatnímy Elastic Beaty pomocí instalačního skriptu `install-beats.sh` přiloženého na paměťovém médiu k této práci.

Ke sběru logů Auditbeat využívá několika modulů – `auditd`, `file_integrity` a `system`. Modul `auditd` je nakonfigurován, aby využíval výchozí sadu pravidel `sample-rules.conf`, která slouží k detekci některých událostí v rámci systému. Modul `file_integrity` je nakon-

figurován, aby monitoroval integritu souborů v několika systémových složkách, ve kterých nejsou běžně očekávány žádné změny obsahu souborů za běhu produkčního software. Modul `system` je nakonfigurován, aby každé 2 minuty oznamoval nově nainstalované, aktualizované a odstraněné balíčky a aby každých 12 hodin oznamoval stav systému, uživatelů, procesů a soketů.

Auditbeat je také nakonfigurován, aby ke všem sesbíraným záznamům doplňoval název nasazené instance a stroje, na kterém běží. Tyto informace jsou zapsány během instalace do konfigurace pomocí definovatelných vlastních polí, která má Auditbeat doplňovat ke každému vytvořenému záznamu.

Výstup z Auditbeatu je nasměrován do Apache Kafka 2.5 na zadanou IP adresu systému uchovávání dat před přenosem a je zapisován do topicu s názvem `auditbeat`.

5.3 Architektura systému uchovávání dat před přenosem

Uchovávání získaných dat

Sesbíraná data z Elastic Beatů putují do nástroje Apache Kafka 2.5, zde jsou uložena do historie logů, kde čekají na export.

Apache Kafka bude na cílový systém nainstalována pomocí přiloženého instalačního skriptu `install-kafka.sh`. Script je přiložen na paměťovém médiu k této práci. Při instalaci je možné upřesnit adresu na které má Apache Kafka běžet a cestu k úložišti dat Apache Kafky a nástroje Zookeeper, který je po boku Kafky nainstalován.

Během instalace je Apache Kafka nakonfigurována, aby zachovávala historii logů 2 roky či do velikosti logů 200Gib. Po instalaci je možné tuto konfiguraci přispůsobit okolnostem. Apache Kafka je také nakonfigurována, aby automaticky nevytvářela nové topicy, takto je možné jednodušeji odhalit případné chyby při konfiguraci programů, které s Apache Kafkou komunikují. Bez této možnosti může být například špatně napsaný nazev topicu jednoduše přehlédnut, což může vést k chybějícím datům.

Export získaných dat

O export dat z nástroje Apache Kafka 2.5 se stará nástroj LogExport, jehož implementaci se věnuje sekce 5.5.

Nástroj LogExport je možné na cílový systém nainstalovat pomocí přiloženého skriptu `install-exporter.sh`. Script je přiložen na paměťovém médiu k této práci. Tento skript vyžaduje k instalaci název nasazené instance, na které bude běžet. Tento název slouží k identifikaci stroje ze kterého exportovaná data pochází a podle tohoto názvu je při importu zvolen správný klíč k ověření podpisu dat. Před instalací je také možné specifikovat IP adresu, na které běží Apache Kafka, cestu ke složce do které budou umístěny konfigurační soubory a cestu k souboru s veřejným klíčem k šifrování logů, kdy by tento klíč měl pocházet existující instance nástroje LogExport nakonfigurované pro import dat.

Během instalace je vytvořen výchozí uživatel pro přístup k webovému rozhraní nástroje LogExport a jsou vygenerovány podepisovací klíče do složky s konfiguračními soubory. Společně s nástrojem LogExport je nainstalován a nakonfigurován i Nginx server, který slouží jako spojovací uzel dynamické a ststické části webové aplikace. LogExport ve výchozím nastavení umožňuje přihlášenému uživateli exportovat všechny čtyři topicy spravované tímto systémem – `filebeat`, `auditbeat`, `metricbeat` a `journalbeat`.

5.4 Architektura systému shromažďování a vizualizace dat

Import získaných dat

O import dat se také stará nástroj LogExport, jehož implementaci se věnuje sekce 5.5. Rozdíl mězi režimem exportování a importování dat je pouze v konfiguraci nástroje a ve vstupním bodu do programu. Webová aplikace i konfigurace Nginx serveru je ale společná s režimem pro export dat, proto se těmto prvkům věnuje pouze předchozí podsekce.

Nástroj LogExport v režimu pro import dat je možné na cílový systém nainstalovat pomocí skriptu `install-importer.sh` přiloženého na paměťovém médiu k této práci. Před instalací je možné specifikovat IP adresu, na které běží Apache Kafka a cestu ke složce do které budou umístěny konfigurační soubory.

Během instalace je vytvořen výchozí uživatel pro přístup k webovému rozhraní nástroje LogExport a jsou vygenerovány šifrovací klíče do složky s konfiguračními soubory. Společně s nástrojem LogExport je nainstalován a nakonfigurován i Nginx server.

Uvnitř konfigurační složky existuje jedna důležitá složka a jeden důležitý soubor. Jedná se o složku `deployments`, do které je nutné vložit veřejné klíče z každého nasazeného systému a pojmenovat je ve formátu `název_instance.pem`. LogExport následně pomocí těchto souborů ověřuje podpis každého importovaného souboru. Důležitým souborem pak je soubor `key_public.pem`, který je nutné mít při instalaci LogExport nástroje v režimu pro export a je potřeba specifikovat instalačnímu skriptu cestu k němu či jej umístit do výchozího umístění.

I při importu dat figuruje Apache Kafka jako prostředník mezi nástrojem LogExport a nástrojem Logstash. Aby nebylo nutné složitě řešit importování dat přímo do Logstashe, je zde nástroj Apache Kafka využit, aby předával importovaná data z nástroje LogExport nástroji Logstash. Toto také umožňuje ve specifických situacích, kdy by bylo mezi monitorovaným systémem a systémem pro shromažďování a vizualizaci dat možné navázat síťové spojení, vynechat nástroj LogExport a zasílat logy přímo z Elastic Beatů na instanci Apache Kafka napojenou na ELK Stack. Případně je možné i na monitorovaný systém nainstalovat ELK Stack a monitorovat systém bez nutnosti data přenášet, přitom ale může vedle ELK Stacku běžet i LogExport připravený k exportování dat v době, kdy nebude možné již využít instance ELK Stacku běžícího na monitorovaném systému.

Instalace a konfigurace nástroje Apache Kafka na systému k shromažďování a vizualizaci dat je i v tomto případě stejná jako na systému uchovávání dat před přenosem, jediným rozdílem je, že může být výhodné upravit konfiguraci Apache Kafka tak, aby nezachovávala historii logů po tak dlouhou dobu, či aby ji nezachovávala vůbec, protože o uchovávání logů se na tomto systému primárně stará nástroj Elasticsearch a uchovávání historie logů nástroje Apache Kafka by pouze vedlo k duplikaci dat na disku. V rámci testování tohoto systému na reálných datech jsem se ale rozhodl historii logů nástroje Apache Kafka ani na tomto systému nezkracovat, pro případy, kdy by bylo nutné ladit problémy s importováním dat.

Zpracování získaných dat

Zpracování logů importovaných na systém k shromažďování a vizualizaci dat obstarává nástoj Logstash. Více informací o tomto nástroji a jeho dovednostech lze nalézt v kapitole 2.5.

Nástroj Logstash je na cílový systém nainstalován společně s ostatními nástroji ze skupiny ELK Stack pomocí instalačního skriptu `install-elk.sh` přiloženého na paměťovém médiu k této práci.

Ke zpracování logů má Logstash oddělenou konfiguraci pro každý topic. V současné chvíli jednotlivé konfigurace zajišťují pouze předávání dat z Apache Kafky do nástroje Elasticsearch. Data jsou v Elasticsearchi dělena do indexů podle data vytvoření.

I přesto, že v současné chvíli Logstash neplní mnoho činností a pouze předává data z Apache Kafka do Elasticsearche, v budoucnosti při rozšiřování tohoto systému hraje klíčovou roli, k tomu aby bylo možné normalizovat sesbíraná data ze systémů se starší konfigurací a nebylo nutné tyto systémy aktualizovat, protože aktualizovat systémy u zákazníků nemusí být vždy možné.

Indexace a uchování získaných dat

Indexaci a uchovávání dat importovaných na systém k shromažďování a vizualizaci dat obstarává nástoj Elasticsearch. Více informací o tomto nástroji a jeho dovednostech lze nalézt v kapitole 2.6.

Nástroj Elasticsearch je na cílový systém nainstalován společně s ostatními nástroji ze skupiny ELK Stack pomocí instalačního skriptu `install-elk.sh` přiloženého na paměťovém médiu k této práci.

Ke svému fungování Elasticsearch nevyžaduje mnoho konfiigurace, proto v rámci tohoto systému je Elasticsearch využívám s jeho výchozí konfigurací.

Vizualizace a analýza získaných dat

Vizualizaci a analýzu dat na systému k shromažďování a vizualizaci dat obstarává nástoj Kibana. Více informací o tomto nástroji a jeho dovednostech lze nalézt v kapitole 2.7.

Nástroj Kibana je na cílový systém nainstalován společně s ostatnímy nástroji ze skupiny ELK Stack pomocí instalačního skriptu `install-elk.sh` přiloženého na paměťovém médiu k této práci.

Ke svému fungování Kibana nevyžaduje mnoho konfiigurace. V rámci tohoto systému je v konfiguraci nástroje Kibana pouze upravena host IP adresa tak, aby bylo možné k rozhraní Kibany přistupovat i z jiných zařízení v rámci sítě.

5.5 Nástroj pro export a import dat

Při snaze vytvořit systém navržený v předchozí kapitole vzniknul problém, který nebylo možné vyplnit za použití existujících řešení. K vyřešení tohoto problému proto vzniknul nástroj LogExport, o jehož implementaci tato sekce pojednává.

Základní myšlenkou tohoto nástroje je využití webového rozhraní k vytváření exportů dat z nástroje Apache Kafka a následně k importování těchto dat na jiném stroji. Všechny ostatní vlastnosti tohoto nástroje pak vznikly kolem této myšlenky.

Jako správná webová služba i nástroj LogExport se skládá ze svou částí backendu a frontendu, které mezi sebou komunikují pomocí REST API. Více o těchto částech v následujících podsekcích.

Mimo webové rozhraní je možné vyvolávat exporty a importy dat i z terminálového rozhraní sereru. Tohoto může být například využito k automatizování exportování dat z Apache Kafky například k zálohování dat. Přestože se tato práce nevěnuje této možnosti využití nástroje LogExport, jeho možnosti rozhodně nekončí u přenášení dat mezi instancemi Apache Kafky.

Backend

Backend představuje dynamickou část nástroje LogExport. Jedná se o aplikaci vytvořenou v programovacím jazyce Python, která za využití knihovny Flask poskytuje frontendu REST API, přes které může frontend komunikovat s backendem za cílem splnění požadavků uživatele.

Backend obstarává přihlašování uživatelů a spouštění exportování a importování dat. Pro frontend také nabízí několik API volání, přes které může frontend ověřovat přihlášení uživatele a dostupné zdroje a cíle pro import či export dat.

Použité technologie

Při implementaci backendu nástroje LogExport bylo využito mnoho existujících knihoven, k dosažení cílové funkcionality.

První přišla na řadu volba programovacího jazyka. Při této volbě bylo nutné vybrat jazyk, ve kterém bude možné jednoduše číst a zapisovat logy do nástroje Apache Kafka a zároveň by mělo v daném jazyce být možné jednoduše vytvořit REST API. Jazyky jako C či PHP proto nebyly k tomuto účelu vhodné. V jazyce C by bylo zbytečně složité pracovat s Apache Kafkou i vytvořit REST API. V PHP je sice jednoduché vytovřit REST API, ale práce s Apache Kafkou v kombinaci s limity jazyka PHP při provádění operací na pozadí, označily i tento jazyk za nevhodný. Jako vhodný se jevil Python a Java, oba tyto jazyky měly dostupné implementace knihovny pro komunikaci s Apache Kafkou a pro oba tyto jazyky existovaly frameworky, ve kterých by bylo možné vytvořit REST API. Výsledná volba nakonec padla na jazyk Python, který jsem se rozhodl použít, protože se mi s ním lépe pracuje a domníval jsem se, že v něm bude jednodušší aplikaci napsat.

Další velice důležitou částí výsledné aplikace je REST API, k tvorbě tohoto API je možné v Pythonu využít vícero knihoven. Při hledání té nejvhodnější, pro účely této aplikace, jsem se zaměřil na jednoduchost práce s danou knihovnou. Planované API nebylo nijak složité a k jeho vytvoření nebylo třeba mnoho speciální funkcionality. Pro Python existují frameworky Flask, Tornado, Sanic, Falcon, Bottle a mnoho dalších a všechny tyto frameworky mají veškerou funkcionalitu potřebnou pro tvorbu REST API. Rozhodnutí zvolit Flask pro tuto aplikaci přišlo až ve chvíli, kdy jsem objevil knihovnu Flask-Login. Kombinace jedné z nejznámějších knihoven k vytváření HTTP rozhraní s již implementovanou autentizací pomocí knihovny Flask-Login se mi zdála být nejlepší možností.

Protože Flask očekává, že v produkci nebude použit jeho vlastní server, ale bude využita jiná produkční implementace WSGI (Web Server Gateway Interface) serveru, musel jsem vybrat, jakou implementaci plánuji využít. Od hledané implementace WSGI serveru jsem neočekával nic jiného, než aby bylo mozné s její pomocí spustit Flask aplikaci. Nepotřeboval jsem kombinovat více serverů, ani nijak speciálně konfigurovat současný server. Servery Gunicorn, uWSGI, CherryPy či waitress a jsou jen pár příklady WSGI serverů, které je možné k provozování Flask aplikace použít. Rozhodl jsem se nakonec použít WSGI server waitress, který se ukázal, že nevyžaduje žádnou dodatečnou konfiguraci a je možné pod ním jednoduše spustit Flask aplikaci.

Ke čtení a zápisu logů z Apache Kafka jsem se rozhodl použít knihovnu kafka-python, toto rozhodnutí nebylo nikterak složité, protože neexistuje mnoho dalších implementací rozhaní Apache Kafky pro Python a Kafka-Python patří mezi nejznámější a nejpoužívanější řešení.

Pro kompresi logů bylo také potřeba nalézt knihovnu, která by se dokázala postarat o kompresi proudu dat. Rozhodl jsem se ke kompresi použít algoritmu zstandard, protože

s mám s tímto algoritmem dobré zkušenosti z pohledu efektivnosti a nenáročnosti na prostředky. Volba knihovny pak závisela na zvoleném algoritmu. Existuje sice více implementací, které umožňují využívat zstandard z Pythonu ale pouze jediná knihovna podporovala proudovou kompresi, a to python-zstandard.

Práce s kryptografií vyžadovala rozsáhlejší průzkum existujících řešení a míry do jaké jsou považována tato řešení za bezpečná. Nakonec se ale ukázala knihovna pycryptodome jako nejznámnější a nejpoužívanější knihovna pro práci s kryptografií v Pythonu, proto jsem se ji rozhodl použít.

K vyřešení pár dalších drobných problémů během vývoje bylo potřebné použít i pár dalších knihoven. K dekódování proudu JSON objektů, který reprezentoval proud záznamů logu, byla zvolena knihovna jsonstream. Pro práci s připojenými USB zařízeními byla zvolena knihovna pyudev. Ke spouštění příkazů z pythonu byla použita knihovna sh, která umožňuje volat příkazy bez nutnosti psát kód k vytváření procesů a práci se vstupy a výstupy programů.

Komunikační rozhraní

Backend poskytuje REST API, které je využíváno frontendem. Toto API je implementováno pomocí Flask frameworku. Za cílem zjednodušení budoucích vylepšení rozhraní, je aktuální verze rozhraní zanořena do cesty `/api/v1`. V případě provedení zpětně nekompatibilních změn je tak možné tyto změny publikovat pod cestou `/api/v2` a stále podporovat starou verzi rozhraní pro aplikace, které novou funkcionalitu nepotřebují. Rozhraní je tvořeno několika příkazy, které je možné vyvolat:

- `GET /login` – slouží k získání aktuálního stavu přihlášení,
- `POST /login` – pokusí se přihlásit uživatele pomocí zaslaných údajů,
- `GET /state` – odpoví aktuálním stavem běžícího importu či exportu,
- `GET /topics` – vrátí seznam exportovatelných Apache Kafka topiců,
- `GET /targets` – vrátí seznam zařízení, pro zápis exportovaných dat,
- `GET /export` – spustí export dat,
- `GET /sources` – vrátí seznam zařízení, pro čtení exportovaných dat,
- `GET /exportFiles` – vrátí seznam souborů na zařízení k importování,
- `POST /import` – spustí import dat.

Ochrana dat

Exportovaná data je nutné chránit, tak aby nebyla čitelná pro jiné stroje nežli pro cílový stroj a aby cílový stroj byl schopný ověřit, z jakého systému data pochází a že nebyla nijak modifikována. Přesněji řečeno je nutné do určité míry zajistit důvěrnost, celistvost a autentizaci exportovaných dat.

K zajištění autentizace a celistvosti má každá instance nástroje LogExport v režimu pro export vlastní RSA klíč a s použitím soukromé části tohoto klíče podepisuje exportovaná data. K vytvoření podpisu je použit algoritmus SHA256, jehož výstupní hash je následně podepsán pomocí PKCS#1 PSS. Během importu následně LogExport v režimu pro import

vyhledá veřejný klíč pro stroj ze kterého soubor pochází a ověří jím podpis souboru, než započne import.

Důvěrnost dat je zajištěna šifrováním dat. Instance nástroje LogExport v režimu pro import dat má vlastní RSA klíč, veřejnou část tohoto klíče pak má k dispozici každá instance nástroje LogExport v režimu pro export dat. Data jsou šifrována náhodným klíčem pomocí AES algoritmu v režimu CTR. Klíč kterým jsou data šifrována je zašifrován veřejným klíčem pocházejícího z instance nástroje LogExport v režimu pro import dat pomocí algoritmu PKCS1 OAEP.

K zajištění celistvosti hlavičky souboru, která není součástí podepsaných dat, je použit HMAC algoritmus v kombinaci se SHA256. Veškerá šifrovaná data včetně hlavičky jsou ověřena tímto algoritmem za cílem znemožnit úpravy hlavičky a šifrovaných dat.

Získávání logů

Logy k exportování je potřeba přečíst z Apache Kafka. K těmto účelům využívá nástroj LogExport knihovny kafka-python. Při implementaci iterátoru logů pomocí třídy KafkaConsumer bylo nutné obejít problém způsobený tím, že KafkaConsumer navazuje spojení se serverem, až když dojde k přečtení prvího záznamu z logu. Problém proto nastává ve chvíli, kdy je potřeba ukončit export po dosažení konce logu, protože není množné ověřit před začátkem čtení logů zda, již není aktuální pozice na konci logu. Tento probém by mohl způsobit zamrznutí exportu, proto jako řešení tohoto problému byla vytvořena funkce, která vyvolá obnovení dat ze serveru, aniž by přečetla jediný záznam z logu. Po zavolání této funkce je již možné ověřit pozici v logu.

Vkládání logů

K importování logů je zapotřebí dokázat zapisovat záznamy logů do Apache Kafka. Narozdíl od čtení logů byl zápis jednoduše implementovatelný a jeho fungování nedoprovázely žádné softwarové nedostatky.

Formát souboru exportovaných logů

Exportované soubory mají vcelku komplikovaný formát, který bylo nutné vytvořit zejména z důvodů spojených s ochranou dat. Formát souboru je tvořen třemi vrstvami, kdy každá následující vrstva obaluje předhozí a daným způsobem ji transformuje.

První vrstva obsahuje pouze samotné záznamy z logů v podobě proudu JSON objektů, které obsahují informace o jednotlivých záznamech. Každý JSON objekt obsahuje informaci o tom, z jakého Apache Kafka topicu pochází, jaký je jeho klíč, hlavička, časové razítko a jaká obsahuje data. Pro názornost obrázek 5.1 znázorňuje strukturu takovéhoto proudu záznamů z logů.

Druhá vrstva se stará o kompresi a podepsání dat z první vrstvy. Data v této vrstvě jsou nejprve komprimována a následně v blocích zapsána na výstup této vrstvy. Za data je pak přípsána délka podpisu a podpis dat. Pro názornost obrázek 5.2 znázorňuje strukturu výstupu této vrstvy.

Třetí a zároveň poslední vrstva se stará o šifrování dat z druhé vrstvy. Na výstup této vrstvy je nejprve zapsána hlavička tvořená magickými bajty[2], názvem nasazené instance

[2]Magické bajty slouží k identifikaci obsahu souboru, operační systém může pomocí této sekvence bajtů na začátku souboru vybrat program, který má být pro vybraný soubor použit.

```
{
   'topic': "<Kafka topic>",
   'value': "<Hodnota>",
   'key': "<Klíč>",
   'headers': "<Hlavičky>",
   'timestamp': "<Časové razítko>"
}
{
   'topic': "<Kafka topic>",
   'value': "<Hodnota>",
   'key': "<Klíč>",
   'headers': "<Hlavičky>",
   'timestamp': "<Časové razítko>"
}
... Tato struktura se opakuje až do konce proudu dat
```

Obrázek 5.1: Struktura první vrstvy souboru exportovaných logů – proud záznamů z logů.

Obrázek 5.2: Struktura druhé vrstvy souboru exportovaných logů – zkomprimovaná a podepsaná data.

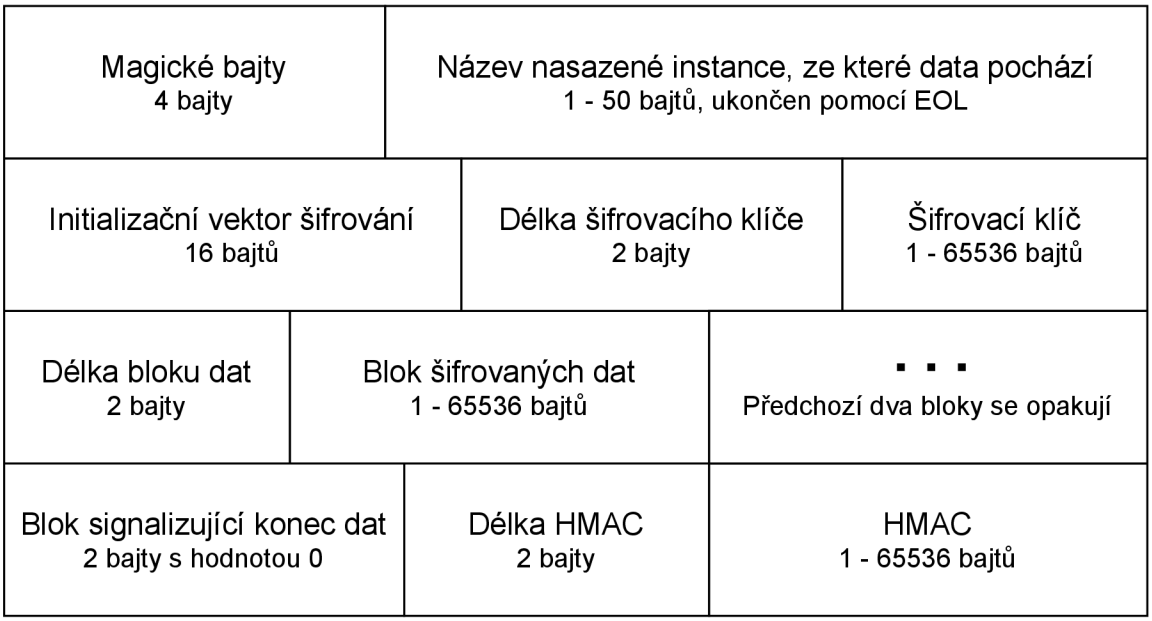
ze které data pochází, inicializačním vektorem šifrování, délkou šifrovacího klíče a šifrovacím klíčem. Poté jsou na výstup zapsána šifrovaná data v blocích. Nakonec je zapsána za data délka HMAC autentizace a HMAC autentizace. Pro názornost obrázek 5.3 znázorňuje strukturu výstupu této vrstvy.

Frontend

Frontend představuje část nástroje LogExport, která běží ve webovém prohlížeči uživatele. Jedná se o webovou aplikaci vytvořenou pomocí knihovny React, která za využití REST API Backendu poskytuje uživateli přehledné UI k přihlášení a exportu či importu logů.

Použité technologie

React jako základní stavební kámen této aplikace byl zvolen, protože již s ním mám zkušenosti z dřívějších projektů a protože splňoval všechny požadavky potřebné pro řešení tohoto problému.



Obrázek 5.3: Struktura třetí vrstvy souboru exportovaných logů – šifrovaná data.

Designové prvky aplikace jsou tvořeny pomocí knihovny Material-UI, která implementuje prvky designu Material.

Navigace v aplikaci je implementována pomocí knihovny React Router, která umožnuje popsat routování aplikace deklarativně pomocí navigačních komponent.

Přihlašovací stránka

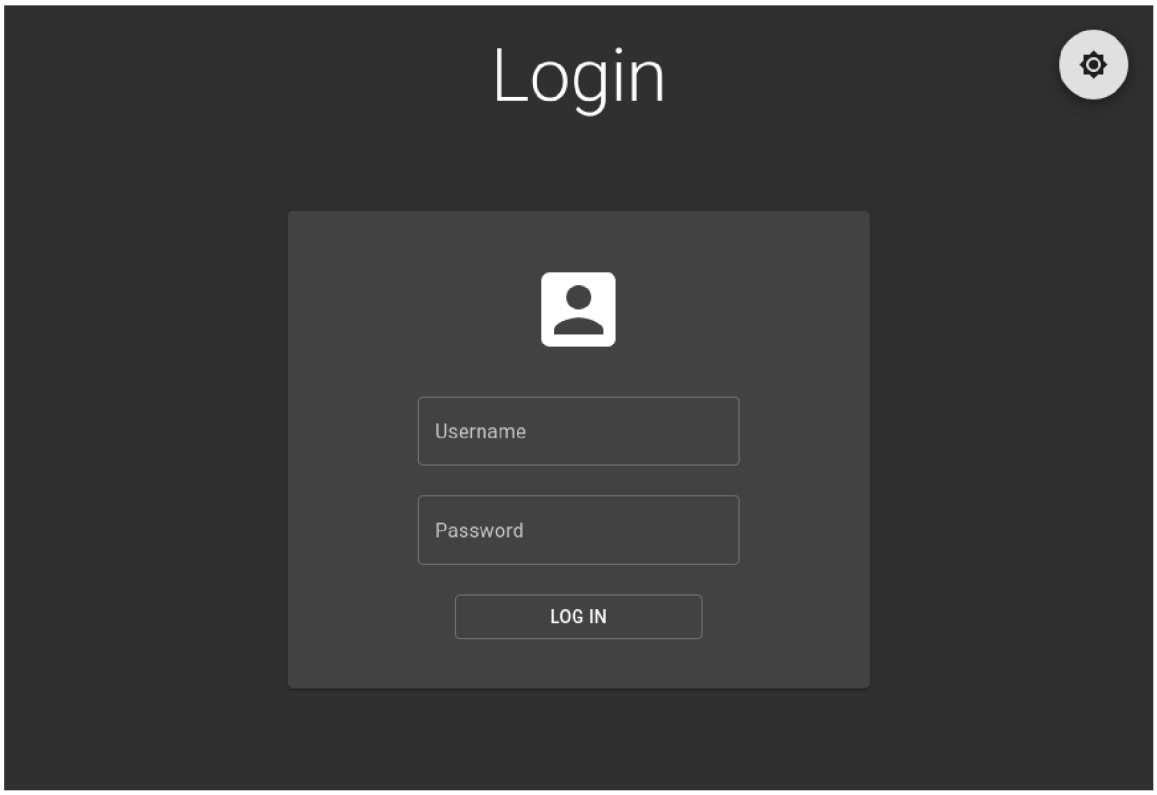
Jak již název napovídá přihlašovací stránka umožňuje uživateli přihlásit se k nástroji LogExport. Teprve po přihlášení bude uživateli umožněno exportovat či importovat data. Snímek obrazovky 5.4 ukazuje jak přihlašovací stránka vypadá.

Stránka k exportu logů

Pokud pracuje nástroj LogExport v režimu pro export logů, po přihlášení bude uživatel uvítán touto stránkou. Na této stránce je možné zvolit Apache Kafka topicy k exportu a vybrat cílové zařízení, do kterého mají být exportovaná data zapsána. V posledním bloku této stránky je také možné monitorovat právě probíhající export dat. Snímek obrazovky 5.5 ukazuje, jak stránka k exportu vypadá.

Stránka k importu logů

Pokud pracuje nástroj LogExport v režimu pro import logů, po přihlášení bude uživatel uvítán touto stránkou. Na této stránce je možné zvolit zdrojové zařízení, ze kterého mají být data importována, poté je možné označit k importu soubory exportů dat nalezené na vybraném zařízení. V posledním bloku této stránky je také možné monitorovat právě probíhající import dat. Snímek obrazovky 5.6 ukazuje, jak stránka k import vypadá.



Obrázek 5.4: Snímek obrazovky přihlašovací stránky.

5.6 Možná rozšíření a vylepšení systému

Během vývoje tohoto systému vzniklo mnoho nápadů, které nakonec nenašly cestu do výsledného řešení. Většina těchto nápadů nebyla špatná, pouze jejich implementace nebyla nutná ke splnění požadavků této práce.

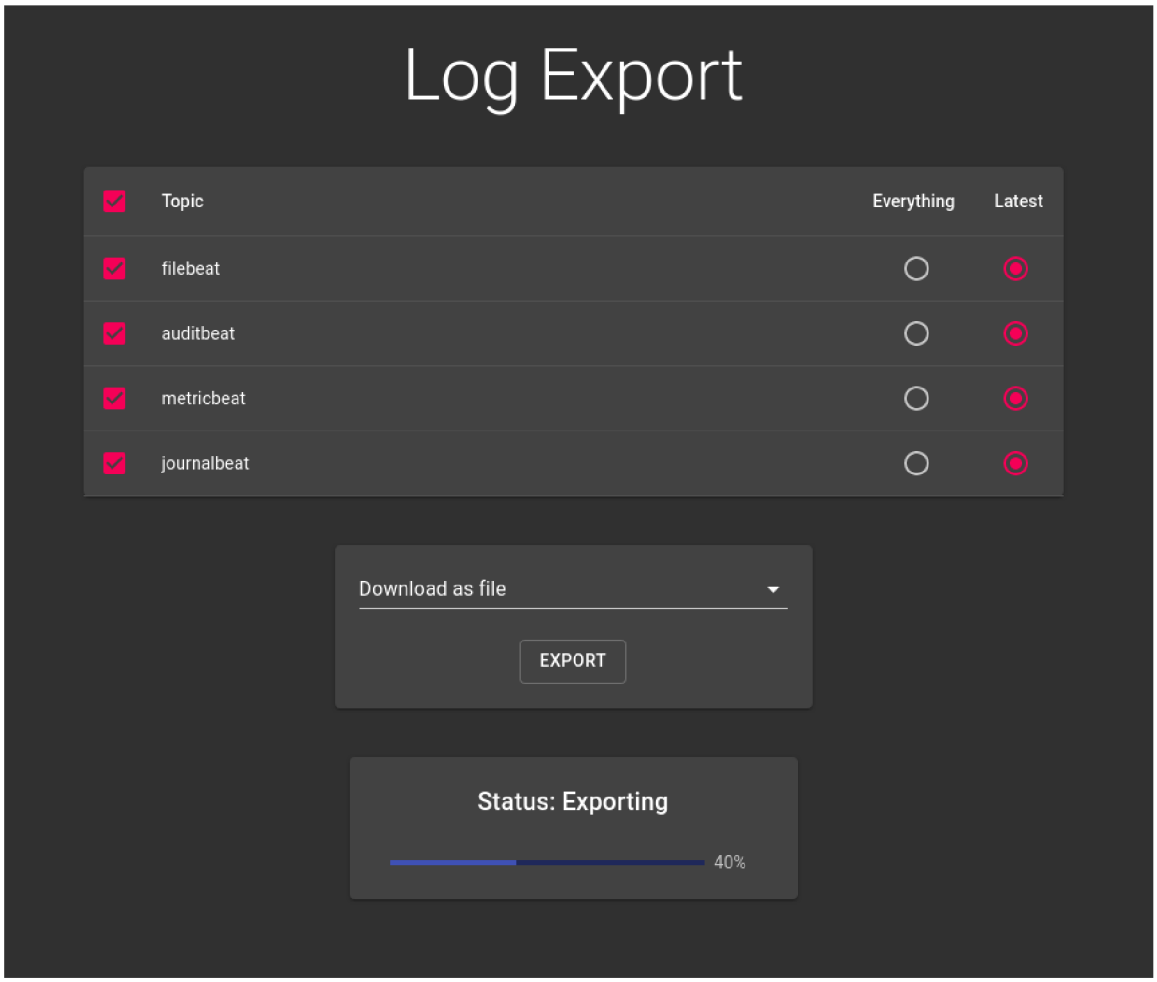
Mnoho možných rozšíření se týká nástroje LogExport, který byl v rámci této práce implementován. Ale i celý systém jako takový je možné v mnoha o vylepšovat a měnit v závislosti na potřebách a okolonostech.

Zakazování vybraných disků k exportu a importu dat

Aktuální implementace nástroje LogExport neumožňuje specifikovat disky, které nemají být nabízeny k importu a exportu dat. Vybrány jsou automaticky všechny disky, které jsou k připojeny přes rozhraní USB. Zatímco v běžném prostředí tento způsob hledání disků přesně odpovídá požadavkům, existují okrajové situace, ve kterých může být vhodné neumožnit export a import dat z vybraných disků. Například pokud systém samotný běží z USB disku, není dobrým nápadem umožňovat exportovat data na tento disk.

Obnovení pozice v logu v případě selhání exportu dat

Při exportování dat je automaticky poslední přečtená pozice ukládána pomocí Apache Kafka a při následujícím exportu je opět automaticky obnovena. Tato vlastnost je užitečná k pokračování čtení logu v místě, kde přestal, v případě selhání exportu dat ale není tato po-



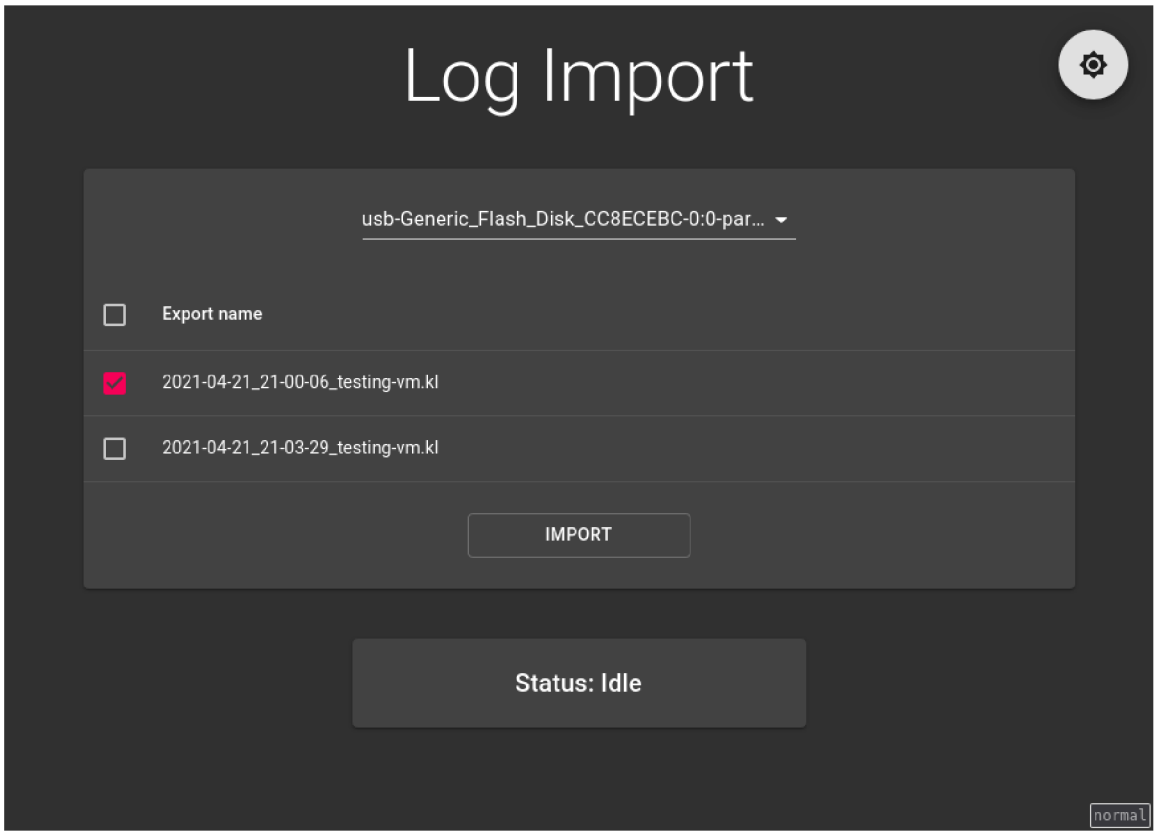
Obrázek 5.5: Snímek obrazovky stránky k exportu logů.

zice v logu obnovena a soubor přerušeného exportu není možně importovat kvůli kontorole podpisu, kterou není možné nad takovým souborem provést.

Pro předejití těmto problémům je možné nástroj LogExport rozšířit o kód, který by v případě selhání exportu dat obnovil pozici v logu na poslední známou před zahájením exportu. Případně by mohlo být zajímavé prozkoumat možnosti odložení odeslání aktuální pozice Apache Kafce a dosáhnout tak podobného výsledku tím, že bude nová pozice v logu uložena pouze až po dokončení exportu dat.

Možnost specifikovat manuálně pozici v logu od které začít export dat

Podobně jako předchozí podsekce i tato se věnuje pozici v logu, od které započne export dat. Ve specifické situaci, například pokud se ztratí již vytvořený export dat před tím, než mohl být importován, může být užitečné mít možnost manuálně zadat pozici začátku či konce exportu. Aktuální implementace tuto možnost ale nenabízí.



Obrázek 5.6: Snímek obrazovky stránky k importu logů.

Podpora více uživatelských účtů

Toto možné rozšíření patří mezi kontroverznější. Nástroj LogExport v aktuální implementaci podporuje pouze jediné přihlašovací údaje, pomocí kterých se všichni uživatelé přihlašují. V jádru je takováto implementace zcela dostatečná, protože není možné na každé instanci u zákazníka mít účet pro každého člověka, který se stará o exportování dat, zároveň i když někdo získá přístup k nástroji LogExport, nemůže získaná data díky šifrování zneužít, takže samotné přihlášení u tohoto nástroje nehraje příliš důležitou roli z pohledu bezpečnosti.

I přesto stojí toto možné rozšíření za zmínku, protože v kombinaci s externím autentizačním serverem se může i jednat o užitečné vylepšení.

Log přihlášení a spuštěných exportů a importů dat

Analýze a řešení problémů s nástrojem LogExport by mohlo pomoci, kdyby server vytvářel log přihlášení a spuštěných exportů a importů dat.

Zbývající čas do dokončení exportu či importu dat

V současné implementaci nástroj LogExport signalizuje průběh exportu či importu dat zobrazováním poměru zpracovaných záznamů vůči zbývajícím záznamům v procentech. Jedním z posledních nápadů je nápat tuto funkcionalitu rozšířit o ukazatel odhadovaného zbývajícího času do dokončení exportu či importu dat.

Přerušení exportu dat

Nástroj LogExport aktuálně nenabízí žádnou možnost přerušení probíhající operace. V praxi by mohlo být užitečné takovou možnost mít. Zároveň by tato možnost mohla ukončit export tak, aby bylo možné i výsledný soubor s částečným exportem dat importovat a nebylo tak nutné začínat export znovu od začátku.

Ukládat RSA klíče šifrované heslem uživatele

RSA klíče, které LogExport využívá jsou na disku uloženy bez šifrování. V případě napadení počítače tak mohou být tyto klíče zneužity útočníkem. Šifrování soukromých RSA klíčů heslem uživatele může snížit dopad tohoto problému za předpokladu, že útočník již toto heslo nezná a není schopen jej hrubou silou zjistit.

Přidávání RSA klíčů instancí zákazníka bez restartu serveru

LogExport v režimu pro import hledá veřejné klíče ostaních instancí pro export dat v zadané složce. V současné implementaci je tato složka prohledána pouze při spuštění programu a všechny klíče jsou načteny do slovníku všech instancí. Tato implementace neumožnuje přidávat a odebírat tyto klíče a je nutné po jejich změně restartovat server.

Zajímavým rozšířením by mohla být úprava této implementace, tak aby byl potřebný klíč načten až při importu dat přímo z dané složky, což by umožnilo dynamicky měnit obsah této složky bez nutnosti server restartovat.

Úspora velikosti exportovaného souboru

Exportovaný soubor ve svém jádru využívá JSON objektu k reprezentaci jednotlivých záznamů logu. Výhodou JSONu je jeho čitelnost pro člověka, k reprezentování takto velkého množství dat ale není příliš vhodný, kvůli zbytečně se opakující struktuře v každém záznamu. Tento problém je v současné implementaci potlačen kompresí nad tímto proudem JSON dat. Tento problém by ale bylo možné zcela odstranit využitím jiného formátu zápisu jednotlivých záznamů.

Doplňovat pole s názvem instance až při importu

V současné implementaci je ke každému záznamu přidáno pole identifikující název instance, na které byl záznam vytvořen, ihned při vytvoření záznamu. Při exportu dat pak ve struktuře každého záznamu je toto pole. Pro úsporu místa a pro zajištění, že všechny záznamy z dané instance používají stejný název, mohl by být tento název instance doplněn až při importu dat buďto nástrojem LogExport či nástrojem Logstash.

Kapitola 6

Analýza shromážděných dat

Pomocí systému, jehož implementaci se věnuje předchozí kapitola, je možné sbírat různé logy a data z vybraných systémů. Ve velkém množství dat ale není snadné nalézt informace. K tomuto účelu slouží analýza a vizualizace dat.

Tato kapitola se zaměřuje na analýzu reálných dat, která byla sesbírána pomocí systému implementováného v přechozí kapitole. Nejprve se první sekce zaměřuje na analýzu využití prostředků systému a popisuje využití informací o vytížení procesoru, využití operační paměti a zatížení disků. Druhá sekce se následně věnuje monitorování vybraného procesu za cílem detekovat úniky paměti. Poté se třetí sekce věnuje monitorování operačního systému a detekci nečekaných jevů v závislosti na modifikaci souborů. Poslední sekce se nakonec věnuje možným rozšířením analýzy data v rámci této práce.

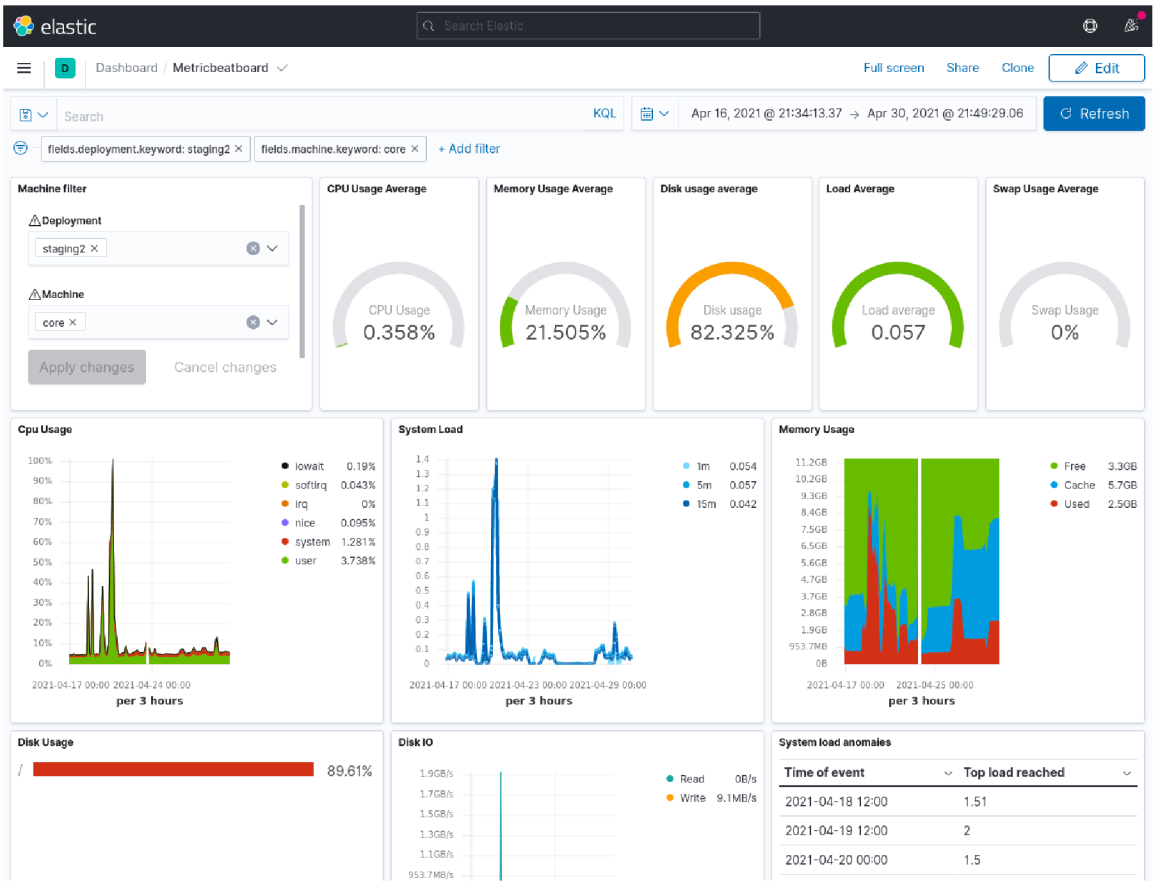
6.1 Monitorování výkonu systému

Důvody monitorování výkonu systému nezahrnují žádné nečekané odůvodnění. V zásadě je vždy dobré využití zdrojů monitorovat a věnovat případným výkyvům pozornost. Monitorování výkonu v případě této práce zahrnuje zatížení procesoru, využití operační paměti, zatížení systému i využití disků. Na obrázku 6.1 domovské stránky pro monitorování výkonu systému je možné vidět grafy vizualizující historii využití různých zdrojů sledovaného systému.

Využití procesoru

Sledování využití procesou může pomoci odhalit, zda daný systém stíhá vyřizovat všechny požadavky. Při vysokém zatížení může mít systém problémy stíhat vyřizovat požadavky včas.

V rámci testování na reálných datech byla tato práce otestována na testovacím systému, kde vyvojáři testují produkční řešení na hardware, který se nejvíce podobá produkčnímu. Software, který je na tomto systému testován využívá nekonečné smyčky k ovládání externího hardware, proto je možné v grafu 6.2 využití procesoru vidět špičky ve chvílích, kdy byl tento software spuštěn k testování, nekteré špičky jsou také způsobeny sestavováním software, kdy kompiler jazyka C++ využívá prostředky daného systému na maximum.



Obrázek 6.1: Kibana dashboard vizualizující využití hardware.

Využití operační paměti

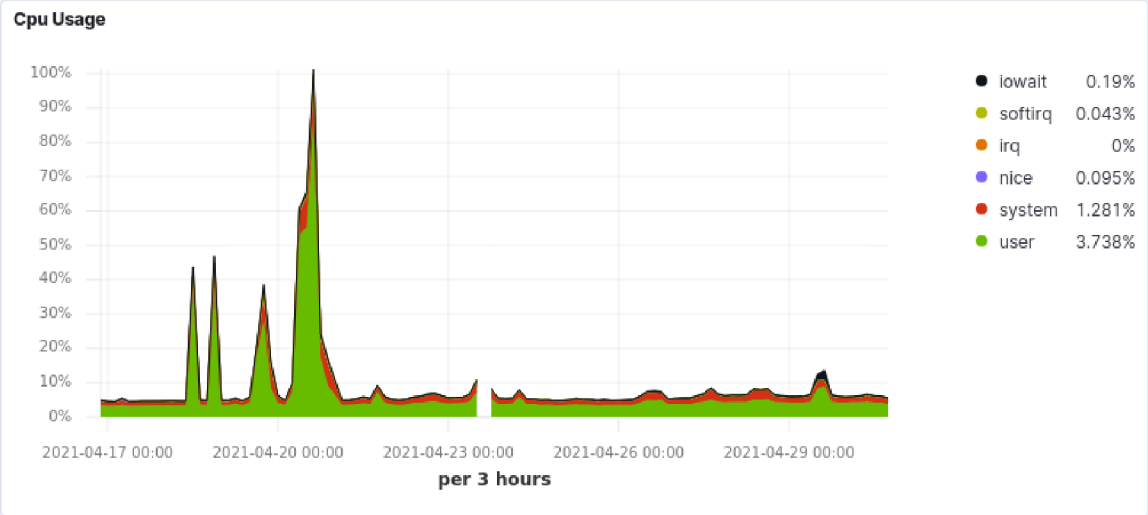
Podobně jako sledování využití procesoru i sledování využití operační paměti může pomoci odhalit problému se sledovaným systémem. Při vysokém využití paměti RAM se může činnost systému výrazně zpomalit a se stále rostoucím využitím opračí paměti může i dojít k selhání služeb.

Podobně jako využití procesoru i u využití operační paměti je možné sledovat v grafu 6.3 špičky způsobené spouštěním testovaného software a jeho sestavováním.

Zpoždění kvůli rychlosti disku

I přesto, že systém není zatížen a má dostatek operační paměti, může se stát, že se začne zpomalovat. Jedním z možných příčin může být čekání systému na data z disku. Tento problém je většinou spojován s mechanickými disky, které musí data před přečtením nejprve mechanicky nalézt na desce a při mnoha požadavcích ke čtení dat z mnoha rozdílných míst na disku je možné, že se zpoždění mezi vznikem požadavku na přečtení dat a samotným přečtením dat prodlouží natolik, že systém přestane zvládat vyřizovat požadavky včas.

Protože reálná data sesbíraná systémem probíraným v této práci pochází z testovacího systému, kde není po většinu času mnoho aktivity, není ani aktivita disků v grafu 6.4 příliš vysoké. V produkci bude ale i tento druh analýzy důležitý. Produkční řešení pracuje s časem, kdy má na vykonání dané operace přesně daný čas a během tohoto času musí



Obrázek 6.2: Kibana graf vizualizující využití procesoru v čase.

dokázat zapsat na disk výsledky dané operace. Data generovaná jedinou operací mohou dosahovat i stovek megabajtů ve velikosti. Pokud by v produkci délka čekání na disk pouze rostla, signalizovalo by to, že systém nestíhá data zapisovat na disk, což by v budoucnu mohlo vést k rostoucímu využití paměti kvůli rosutoucí frontě k zápisu a následně ke ztrátě dat, která nebylo možné včas zapsat na disk.

6.2 Monitorování kritického procesu

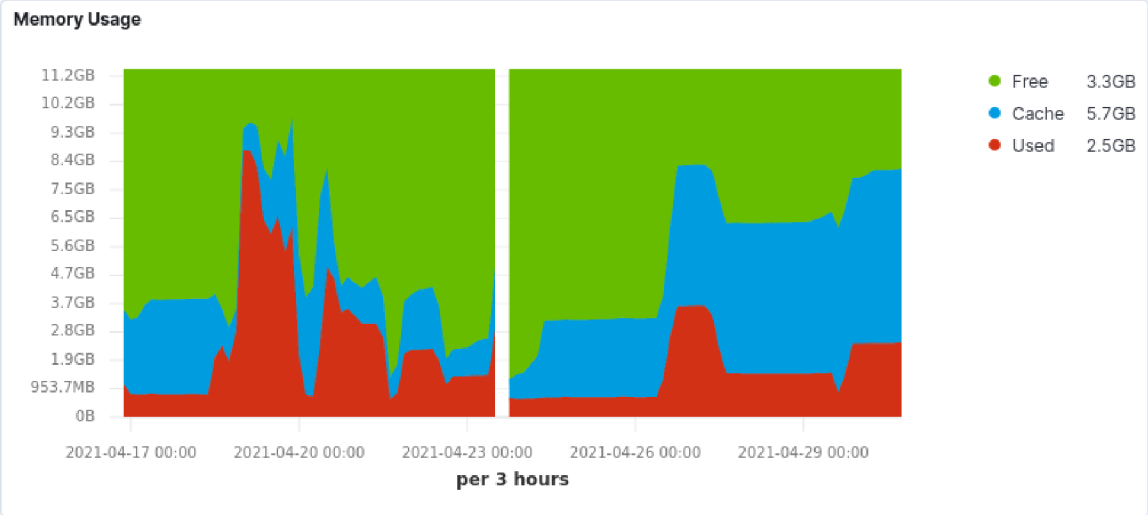
Téměř každý systém je závislý na některých službách k tomu, aby správně fungoval. V mnoha případech je pak těmto službám věnována zvýšená pozornost, co se týče monitorování jejích stavu a analyzování jejich chování za cílem předejít jejich selhání.

Ani systém monitorovaný v rámci testování této práce na reálných datech není jiný. Ve virtuálním stroji s názvem Core běží služba pojmenovaná Core. Tato služba se stará o nejdůležitější činnost celého systému – komunikuje s hardware a provádí oprace, které jsou kritické pro celý systém. Proto tato služba nesmí havarovat ani jinak přestat fungovat.

Detekce úniků paměti

Jedním z častých zdrojů problémů u služeb, které běží dlouhodobě bez restartu, jsou úniky paměti, což jsou situace, kdy si program alokuje paměť, ale poté ji neuvolní. Úniky paměti bývají pozorovatelné tak, že po dobu běhu programu postupně roste využití paměti tímto programem.

Bohužel v testovacím prostředí, kde byla data sbírána, neběží většinou tato služba dlouhodobě, ale je spouštěna opakovaně vyvojáři pro potřeby testování. Z grafu 6.5 za kratší sledovanou dobu se tak dá rozpoznat, že k žádným závažným únikům paměti nedochází, ale případné drobné úniky paměti pomůže odhalit až dlouhodobé monitorování v produkci.



Obrázek 6.3: Kibana graf vizualizující využití operační paměti v čase.

6.3 Monitorování operačního systému

I operační systém je důležité monitorovat. Právě operační systém je prvním zdrojem informací při detekci bezpečnostních incidentů a při hledání nechtěné aktivity. Mimo bezpečnost mohou logy z operačního systému upozorňovat na hardwarové problémy a pomoci tak s jejich řešením. Na obrázku 6.6 domovské stránky pro monitorování operačního systému je možné vidět graf vizualizující množství událostí zaznamenaných v historii sledování systémů.

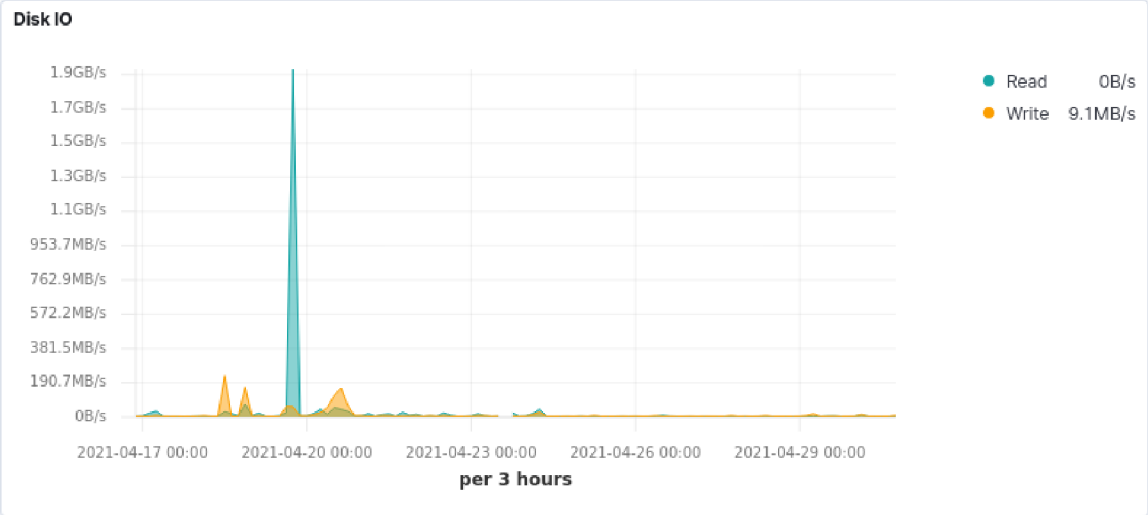
Změny v souborovém systému

Jedním ze způsobů, jak detekovat nežádoucí aktivitu je monitorování změn souborového systému. Zejména ve složkách, jejichž obsah by se za běžných okolností neměl měnit. Informace o změněných souborech mohou prozradit například proniknutí do systému.

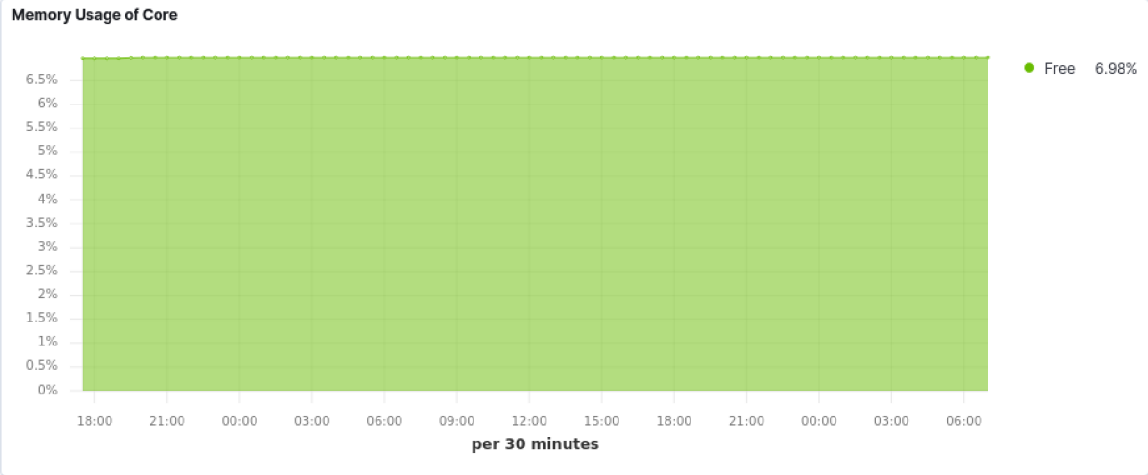
Na obrázku 6.7 je možné vidět historii událostí týkající se změn souborů ve vybraných složkách. Testovací prostředí, ve kterém byla tato data sbírána, je bohužel na rozdíl od produkčního prostředí mnohem častěji aktualizováno a jinak upravováno vyvojáři, a ze sesbíraných dat tak není možné vyčíst mnoho zajímavých informací. Například je vidět, že při instalaci software k testování je velice často modifikována i mezipaměť linkeru.

Pozorování nečekaných jevů

Při analýze logů existuje mnoho technik detekce anomálií. Některé využívají strojového učení, jiné umožnují definovat různorodá pravidla ke klasifikaci dat. Společné mají ale jedno – hledají ve velkém množství dat důležité události a zvýrazňují je jejich uživateli. I bez speciálních programů lze ale některé anomálie detekonvat. Při pohledu na graf na obrázku 6.8 je možné vidět, že ve dnech 14. - 19. 4. 2021 bylo z neznámých důvodů vygenerováno nadměrné množství logů. Po přiblížení na tuto špičku 6.9 lze pak z logů relativně jednoduše pochopit, co se stalo – na jednom z monitorovaných systémů došlo místo na disku a chyby generované programy vytvořily tyto špičku.



Obrázek 6.4: Kibana graf vizualizující čas čekání na data z disků v čase.



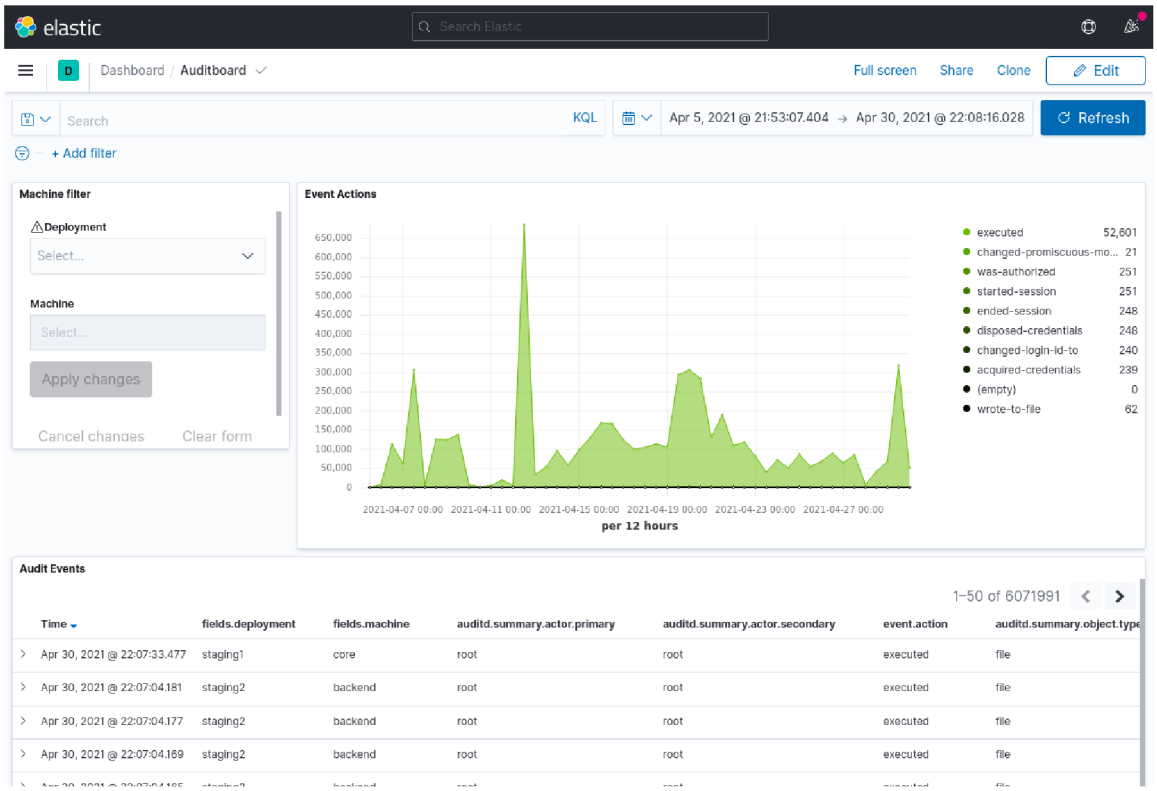
Obrázek 6.5: Kibana dashboard vizualizující využití ram kritickou službou v čase.

6.4 Možná rozšíření a vylepšení analýzy

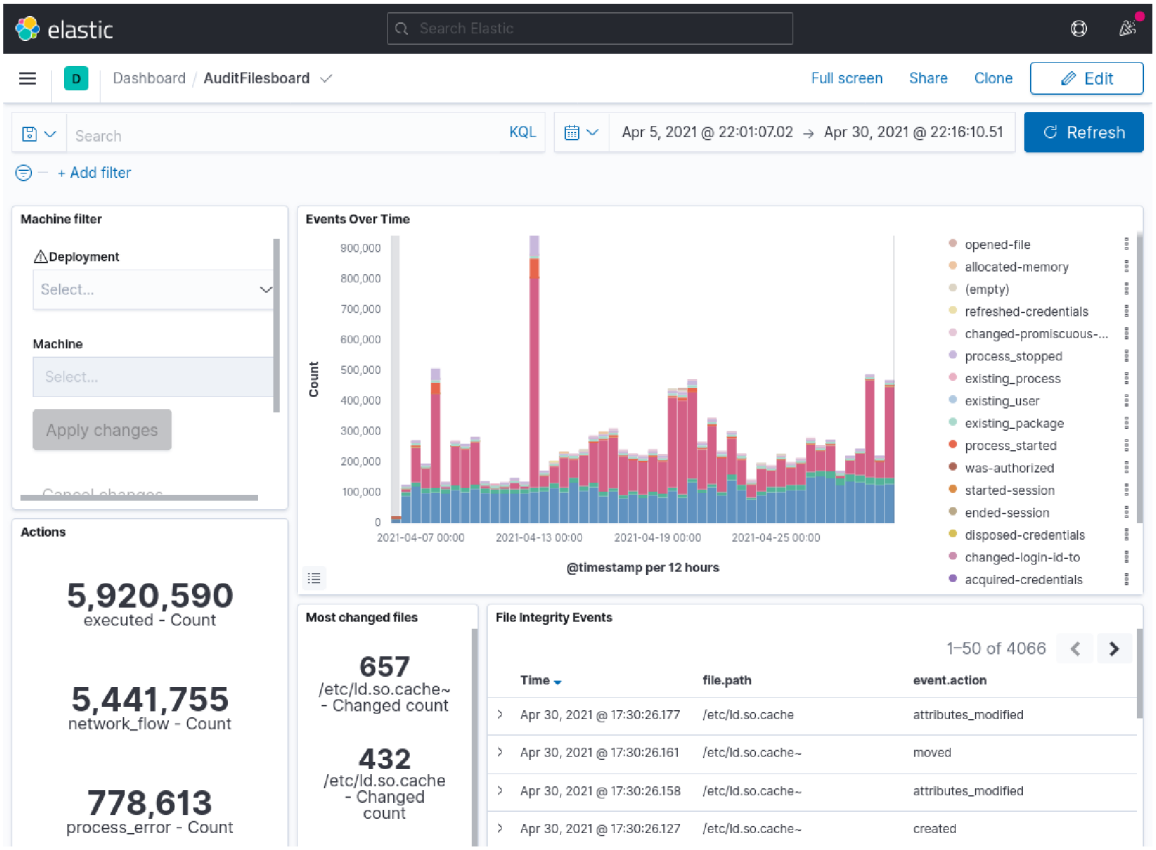
I přesto, že již v tuto chvíli je možné analýzou reálných dat z testovacího systému vyčíst mnohé zajímavosti ohledně chování systému, existuje stále mnoho způsobů, jak tuto analýzu a vizualizaci dat rozšířit.

Při práci s vizualizacemi nad velkým množstvím dat je zřejmé, že stále existuje prostor ke zlepšení výkonu vyhledávání v datech. Správné nastavení indexování, lepší filtrace nepotřebných dat, to vše může zpříjemnit následnou práci s daty.

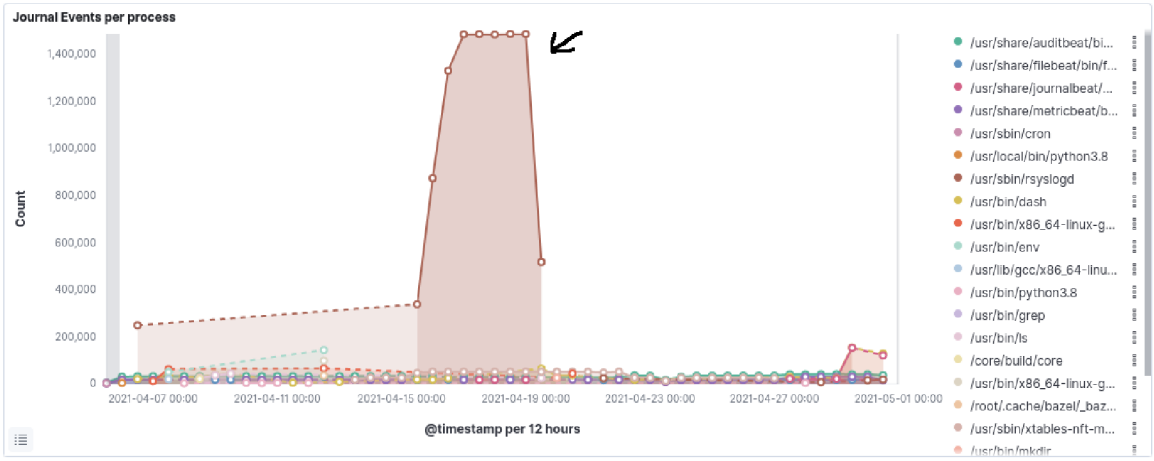
V rámci vizualizace využití hardware lze vizualizovat stav a historii stavu vybraného stroje. V praxi by mohlo být užitečné mít možnost vizualizovat na jednom místě stav všech strojů a zvýraznit v takové vizualizaci stroje, jejichž stav se vymyká běžnému stavu.



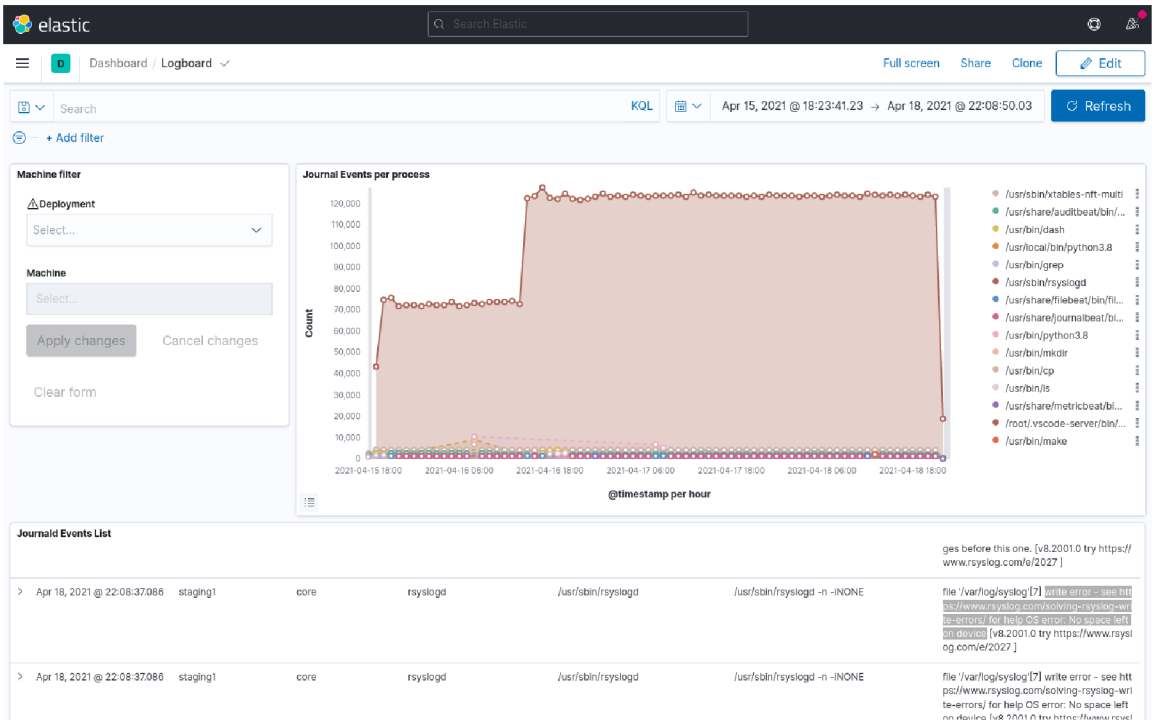
Obrázek 6.6: Kibana dashboard vizualizující četnost výskytů různých auditních událostí v čase.



Obrázek 6.7: Kibana dashboard vizualizující četnost výskytů auditních událostí spojených se soubory.



Obrázek 6.8: Kibana graf vizualizující množství logů ze systémového žurnálu v čase se zvýrazněnou anomálií.



Obrázek 6.9: Kibana dashboard vizualizující množství logů ze systémového žurnálu v čase přiblížený na časový úsek s anomálií.

Kapitola 7

Závěr

Cílem bakalářské práce bylo sbírat data ze systémových logů pro zvýšení bezpečnosti a dostupnosti informačních systémů se zaměřením na utilizaci hardware. Úkolem bylo vytvořit službu na sběr dat a aplikaci na jejich zpracování, analýzu a vizualizaci.

Tato bakalářská práce začala vysvětlením principů práce s logy a rozebráním tématu analýzy a vizualizace dat. Vysvětluje rozdíl mezi klasickými textovými logy a jinými logy, které se využívají například v distribuovaných systémech. Rozebírá i jednotlivé kroky sběru a analýzy dat. Komentuje jejich možnosti využití. Práce se také věnuje existujícím řešením na poli sběru logů a utilizace hardware a analýzy dat.

Za cílem objasnit, jakým způsobem budou chráněna data přenášená mezi systémy, věnuje se tato bakalářská práce základním principům ochrany dat při přenosu. Rezebírá zde téma asymetrické a symetrické kryptografie, po nichž následuje sekce věnující se elektronickým podpisům a jejich úkolu při ochraně dat před modifikací při přenosu.

Na základě získaných vědomostí byl následně navržen systém pro sběr a analýzu dat z linuxových systémů v produkčním prostředí bez síťového spojení se systémem, který by data shromažďoval a analyzoval. K analýze a vizualizaci dat byla díky svým schopnostem a univerzalitě jako nejvhodnější zvolena sada nástrojů ELK Stack v kombinaci s Apache Kafka a Elastic Beats. Využití těchto nástrojů umožnilo navázat sběr, import, export i analýzu dat do systému, který dokáže sbírat data ze zařízení, která nejsou sítí spojena se systémem k shromažďování a vizualizaci dat.

Když jsem obdržel toto zadání, předpokládal jsem, že se z velké části bude jednat o unikátní projekt, který v praxi vyřeší existující problém a pomůže tak mnoha firmám. Při zjišťování informací o existujících řešeních a jejich analýze jsem došel k závěru, že již existuje mnoho kvalitního software, který se tímto problémem zabývá. Proto jsem přehodnotil způsob jakým lze naplnit cíl a úkol mé bakalářské práce. Záměrem práce je nyní sestavit vhodné řešení za využití existujícího software a doimplementovat chybějící funkcionalitu pro splnění cíle systému, který vytvářím.

Dle vytvořeného návrhu a s informacemi získanými během průzkumu exisujících řešení, byl implementován systém, jehož cílem je vyřešit problém který návrh systému definoval. Kombinací technologií ELK Stacku, Apache Kafky, Elastic Beatů a nově vytvořeného nástroje LogExport vznikl systém schopný sbírat data na zařízeních u klientů a následně tato data exportovat a přenášet na zařízení ve firemní infrastruktuře k analýza a vizualizaci.

Nástroj LogExport byl implementován tak, aby umožnil komukoliv jednoduše exportovat a importovat data z jednoho systém na druhý. Nástroj vytváří inkrementální exporty dat ze systému, na kterém běží do podepsaného a šifrovaného souboru pro zajištění ochrany dat při přenosu.

Na závěr se práce věnuje analýze dat, která byla získána během testování řešení na reálných datech. I přes limitace prostředí, na kterém byla data sesbírána lze v datech nalézt zajímavé informace.

Výsledná práce se nakonec nezastavila na okraji zadání a prozkoumala i v dnešní době méně běžný prostor přenosu dat mezi zařízeními bez síťového spojení, věnovala se zabezpečení přenášených dat, ale i práci s logy, jejich sběrem a analýzou i vizualizací za cílem zvýšit bezpečnost a dostupnost systémů se zaměřením na utilizaci hardware.

Literatura

[1] *Apache Kafka – Platforma na předávání událostí v prostředí distribuovaných systémů* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://kafka.apache.org/intro.

[2] *Apache Kafka Dokumentace – Platforma na předávání událostí v prostředí distribuovaných systémů* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://kafka.apache.org/documentation/.

[3] *Apache Kafka na Wikipedii – Platforma na předávání událostí v prostředí distribuovaných systémů* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Apache_Kafka.

[4] *Asymetrická kryptografie – Základy použití a principů asymetrické kryptografie* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Public-key_cryptography.

[5] *Auditbeat – Nástroj ke sběru auditních událostí ze systému* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://www.elastic.co/beats/auditbeat.

[6] *Azure Log Analytics agent – Nástroj ke sběru logů z virtuálních strojů v cloudu* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://docs.microsoft.com/en-us/azure/azure-monitor/agents/log-analytics-agent.

[7] Bishop, M. *Computer security: art and science*. 2. vyd. Addison-Wesley, 2019. ISBN 978-0-321-71233-2.

[8] *Elastic Beats – Sada nástrojů k odesílaní různých dat ze zdrojových zařízení* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://www.elastic.co/beats.

[9] *Elasticsearch – Nástroj ke zpracování a vyhledávání ve velkém množství dat* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://www.elastic.co/elasticsearch/.

[10] *Elasticsearch na Wikipedii – Nástroj ke zpracování a vyhledávání ve velkém množství dat* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Elasticsearch.

[11] *Electronic signature – Základy použití a principů elektronického podpisu* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Electronic_signature.

[12] *Filebeat – Nástroj ke sběru textových logů* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://www.elastic.co/beats/filebeat.

[13] *Information security – Základy zabezpečení a ochrany informací* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Information_security.

[14] *Journalbeat – Nástroj ke sběru logů ze systémového žurnálu* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://www.elastic.co/guide/en/beats/journalbeat/current/journalbeat-overview.html.

[15] *Kafkacat GitHub repozitář* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://github.com/edenhill/kafkacat.

[16] *Kibana – Nástroj k vizualizaci dat* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://www.elastic.co/kibana.

[17] *Kibana na Wikipedii – Nástroj k vizualizaci dat* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://en.wikipedia.org/wiki/Kibana.

[18] KREPS, J. *I ♡ logs : event data, stream processing, and data integration*. 1. vyd. Sebastopol, CA: O'Reilly Media, 2015. ISBN 978-1-491-90938-6.

[19] KSPARENBERG. *Top 10 Log Sources You Should Monitor* [online]. 2018 [cit. 2021-05-10]. Dostupné z: https://www.dnsstuff.com/top-10-log-sources-you-should-monitor.

[20] *Logrotate – Služba k usnadnění administrace logových souborů* [online]. 2021 [cit. 2021-01-17]. Dostupné z: https://github.com/logrotate/logrotate.

[21] *Logstash – Nástroj ke sběru dat z více zdrojů* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://www.elastic.co/logstash.

[22] *Metricbeat – Nástroj ke sběru logů utilizace prostředků systému* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://www.elastic.co/beats/metricbeat.

[23] MOFFATT, R. *Copying data between Kafka clusters with Kafkacat* [online]. 2019 [cit. 2021-05-10]. Dostupné z: https://rmoff.net/2019/09/29/copying-data-between-kafka-clusters-with-kafkacat/.

[24] *Monitorix – Nástroj ke sběru a vizualizaci stavu sítě* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://www.monitorix.org/.

[25] *Monitorix GitHub repozitář* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://github.com/mikaku/Monitorix.

[26] *Monitorix na Wikipedii – Nástroj ke sběru a vizualizaci stavu sítě* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Monitorix.

[27] *Netdata – Nástroj ke sběru a vizualizaci historie utilizace prostředků systému* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://www.netdata.cloud/product/.

[28] *Netdata GitHub repozitář* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://github.com/netdata/netdata.

[29] *Rsyslog – Implementace protokolu syslog ke sběru a filtrování zpráv* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://wiki.archlinux.org/title/Rsyslog.

[30] STALLINGS, W. *Network security essentials: applications and standards*. 4. vyd. Prentice Hall, Pearson Education, 2011. ISBN 978-0-13-610805-4.

[31] *Symetrická kryptografie – Základy použití a principů symetrické kryptografie* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Symmetric-key_algorithm.

[32] *Syslog – Protokol k posílání zpráv mezi programy* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Syslog.

[33] *Syslog – Implementace protokolu syslog k přeposílání a filtrování zpráv* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://wiki.archlinux.org/title/Syslog-ng.

[34] *Systemd – Sada základnách stavebních bloků operačního systému Linux* [online]. 2021 [cit. 2021-01-17]. Dostupné z: https://systemd.io/.

[35] *Systemd-netlogd – Nástroj k přeposílání událostí z Journald* [online]. 2021 [cit. 2021-01-18]. Dostupné z: https://github.com/systemd/systemd-netlogd.

[36] *Systemd/Journal* [online]. 2021 [cit. 2021-05-10]. Dostupné z: https://wiki.archlinux.org/title/Systemd/Journal.

Příloha A

Obsah přiloženého paměťového média

- **installers**

  Tato složka obsahuje několik instalačních skriptů, na které se text této práce odkazuje. Jednotlivé skripty slouží k instalaci různých částí systému, jehož návrhem a implementací se tato práce zabývá.

- **logexport**

  Složka obsahující kód Python balíčku, který implementuje backend nástroje LogExport.

- **log-export-frontend**

  V této složce jsou zdrojové kódy webové aplikace nástroje LogExport, napsané pomocí knihovny React.

- **utils**

  Tato složka obsahuje pár skriptů, které vznikly v rámci implementace systému navrženého v této práci. Jednotlivé skripty slouží k zjednodušení ladění nástroje LogExport při provozu mimo prostředí pro které byl navržen.

- **xkuchy02-text**

  Zdrojový kód textu tohoto dokumentu v LaTeXu.

- **LICENSE**

  Licence pod kterou je kód této práce vydán.

- **README.md**

  Průvodní manuál k používání obsahu paměťového média.

- **xkuchy02-print.pdf**

  Text bakalářské práce vygenerovaný ve verzi pro tisk.

- **xkuchy02-wis.pdf**

  Text bakalářské práce vygenerovaný ve verzi pro odevzdání online.