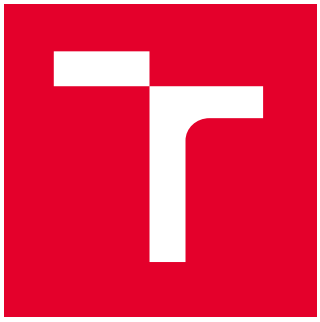


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

INTEGRACE POKROČILÝCH METOD UMĚLÉ INTELIGENCE S BEZPEČNOSTNÍMI SYSTÉMY PROVÁDĚJÍCÍMI MANAGEMENT LOGOVÝCH ZÁZNAMŮ

INTEGRATION OF ADVANCED ARTIFICIAL INTELLIGENCE METHODS WITH LOG MANAGEMENT SECURITY SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jiří Sedláček**
AUTHOR

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

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jiří Sedláček ***ID:*** 203714

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Integrace pokročilých metod umělé inteligence s bezpečnostními systémy provádějícími management logových záznamů

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce proveďte rešerši technik hlubokého učení umožňujících integraci s moderními monitorovacími platformami provádějícími agregaci a indexaci logů. Dle provedené analýzy, implementujte vybrané scénáře a integrujte vytvořenou neuronovou síť s vybranou platformou. V rámci teoretické části nastudujte problematiku bezpečnostního monitoringu se zaměřením na řešení typu SIEM (RSA Netwitness, QRadar) a na open-source řešení pro vyhledávání logů (Elasticsearch). Proveďte srovnání monitorovacích platforem z hlediska jejich funkcionality a způsobů propojení s externími systémy řízenými umělou inteligencí. Zpracujte rešerši možných vylepšení SIEM systémů pomocí technik hlubokého učení a definujte možnosti jejich aplikování se zaměřením na zefektivnění práce uživatele platformy. V rámci praktické části implementujte minimálně dva scénáře vylepšení systémů bezpečnostního monitoringu. Vytvořte vhodné datové sady umožňující realizovat vybrané scénáře. Proveďte návrh a implementaci API rozhraní umožňujícího integraci s vybranou monitorovací platformou. Realizujte testování modelu pro použití na zvolené problémy, výsledky testování vhodně prezentujte.

DOPORUČENÁ LITERATURA:

[1] GONZALEZ-GRANADILLOG, Gustavo, Susana GONZALEZ-ZARZOSA a Rodrigo DIAZ. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. Sensors 21.14 (2021).

[2] CHOLLET, Francois. Deep learning with Python. Shelter Island, New York: Manning Publications Co., [2018]. ISBN 1617294438.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 24.5.2022

Vedoucí práce: Ing. Yehor Safonov

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

ABSTRAKT

Kybernetická bezpečnost je velice důležitým aspektem našeho každodenního života. Se stále více se rozpínajícím kybernetickým prostorem a jeho rostoucím vlivem na náš reálný svět je o to více důležitější právě otázka kybernetické bezpečnosti. V rámci teoretické části diplomové práce jsou popsány základní aspekty bezpečnostního monitoringu. Také je stručně popsán proces sbírání logů událostí a jejich správa. Důležitým prostředkem bezpečnostního monitoringu je management bezpečnostních informací a událostí. Jsou zde probrány jeho výhody, nevýhody a možná vylepšení pomocí umělé inteligence. V teoretické části je rovněž zmíněna funkce orchestrace zabezpečení, automatizace a odezvy. Také jsou zde popsány techniky strojového učení, jako jsou neuronové sítě a hluboké učení. Tato část je rovněž zaměřena na kybernetická operační centra z hlediska zvýšení efektivity lidské „manuální" práce. Byla také provedena rešerše možných technik strojového učení pro tento případ použití, jelikož nedostatek lidských zdrojů je v rámci kybernetických operačních center kritickým problémem. Praktická část diplomové práce zahrnuje vytyčení cíle (klasifikace sekvencí textu), díky kterému by se dala značně ulehčit práce ve smyslu ručního rozdělování logů událostí na kategorie podle jejich zdroje. Pro tento stanovený úkol byla z různých zdrojů logů shromážděna data souvisejí s bezpečnostním monitoringem. V praktické části jsou také podrobně popsány metody pro zpracování těchto dat. Následně byl vybrán vhodný model neuronové sítě a proveden jeho technický popis. Na závěr je popsáno finální zpracování dat a proces trénování, validace a testování modelu. Pro tento proces byly zpracovány tři scénáře, které jsou následně podrobně popsány ve výsledcích měření.

KLÍČOVÁ SLOVA

BERT, Bezpečnostní monitoring, Hluboké učení, Kybernetická bezpečnost, Kybernetické operační centrum, Log události, Management bezpečnostních informací a událostí, Neuronová síť, Umělá inteligence, Zpracování přirozeného jazyka.

ABSTRACT

Cyber security is a very important aspect of everyone's daily life. With the ever-expanding cyberspace and its growing influence on the real world, the issue of cyber security is all the more important. The theoretical part of the thesis describes the basic aspects of security monitoring. Also, the process of collecting event logs and their management is briefly described. An important means of security monitoring is the management of security information and events. Its advantages, disadvantages and possible improvements with artificial intelligence are discussed. Security orchestration, automation and response functions are also mentioned in the theoretical part. Machine learning techniques such as neural networks and deep learning are also mentioned. This section also focuses on cyber operations centres in terms of improving the efficiency of human "manual" labour. A survey of possible machine learning techniques for this use case has been conducted, as the lack of human resources is a critical issue within security operations centres. The practical part of the thesis involves setting out a goal (text sequence classification) that could make the work considerably easier in terms of manually categorizing event logs according to their source. For this set task, security monitoring related data was collected from different log sources. In the practical part, the methods for processing this data are also described in detail. Subsequently, a suitable neural network model was selected and its technical description was performed. Finally, the final data processing and the process of training, validating and testing the model are described. Three scenarios were developed for this process, which are then described in detail in the measurement results.

KEYWORDS

Artificial intelligence, BERT, Cybersecurity, Deep learning, Event log, Natural language processing, Neural network, Security information and event management, Security monitoring, Security operations center.

SEDLÁČEK, Jiří. *Integrace pokročilých metod umělé inteligence s bezpečnostními systémy provádějícími management logových záznamů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 87 s. Diplomová práce. Vedoucí práce: Ing. Yehor Safonov

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jiří Sedláček
VUT ID autora:	203714
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Integrace pokročilých metod umělé inteligence s bezpečnostními systémy provádějícími management logových záznamů

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

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval Ing. Yehoru Safonovovi za obrovskou ochotu, trpělivost, pedagogickou a odbornou pomoc a cenné rady při zpracování mé diplomové práce.

Obsah

Úvod **11**

1 Problematika monitoringu v oblasti kybernetické bepečnosti **14**
1.1 Účel bezpečnostního monitoringu 14
1.2 Analýza datových toků v síti 16
1.3 Správa logů událostí 18
1.4 Management bezpečnostních informací a událostí 22
1.5 Nedostatky systémů SIEM 27
1.6 Orchestrace zabezpečení, automatizace a odezvy 28

2 Bezpečnostní monitoring s využitím umělé inteligence **30**
2.1 Úvod do strojového učení 30
2.2 Neuronové sítě a hluboké účení 33
2.2.1 RNN (*Recurrent Neural Networks*) 39
2.2.2 CNN (*Convolutional Neural Networks*) 41
2.2.3 TNN (*Transformer Neural Networks*) 41
2.3 Zpracování přirozeného jazyka 42
2.4 Vylepšení bezpečnostního monitoringu pomocí AI 43

3 Metodologie kategorizace logových záznamů **51**
3.1 Charakteristika dat 52
3.2 Zpracování datové sady 54
3.3 Volba modelu neuronové sítě 57
3.4 Technický popis modelu 59
3.5 Konečné zpracování dat a proces učení 61

4 Výsledky měření **68**

Závěr **74**

Literatura **77**

Seznam symbolů, veličin a zkratek **85**

A Obsah elektronických příloh **87**

Seznam obrázků

1.1 Porovnání interních a externích řešení SOC 16
1.2 Proces logování [8] 19
2.1 Dělení umělé inteligence [28] 30
2.2 Strojové učení a jeho dělení. 31
2.3 Struktura mělké neuronové sítě 35
2.4 Struktura hluboké neuronové sítě 36
2.5 Proces trénování neuronových sítí 37
2.6 Příklad matice záměn 38
2.7 Strojové učení a jeho dělení s ohledem na možná vylepšení v oblasti kybernetické bezpečnosti. 50
3.1 Vytyčení jednotlivých kategorií a podkategorií LES 53
3.2 Proces zpracování datové sady [28] 56
3.3 Výsledné datové sady 57
3.4 Proces předtrénování modelu Google BERT [73] 60
3.5 Výpis trénovací datové sady po doplnění speciálních tokenů `[CLS]` a `[SEP]` 61
3.6 Tokenizace logů událostí jednotlivých datových sad 62
3.7 Výpis prvního záznamu z trénovací datové sady po tokenizaci 62
3.8 Metoda pro detekci a oříznutí sekvencí dat o délce větší než `MAX_LEN` 63
3.9 Konverze sekvencí dat na tenzory pomocí PyTorch 64
3.10 Výpis tenzorů pro trénování neuronové sítě 65
3.11 Vytvoření finálních datových sad z tenzorů 66
3.12 Konečné kroky před učením modelu 67
4.1 Graf znázorňující průběh `Loss` funkce v poměru ke zpracovaným `batches` prvního scénáře 69
4.2 Graf znázorňující průběh `Loss` funkce v poměru ke zpracovaným `batches` druhého scénáře 70
4.3 Předpovězené hodnoty modelem (osa X) a reálné hodnoty (osa Y) pro druhý scénář 71
4.4 Graf znázorňující průběh `Loss` funkce v poměru ke zpracovaným `batches` třetího scénáře 72
4.5 Předpovězené hodnoty modelem (osa X) a reálné hodnoty (osa Y) pro třetí scénář 72

Seznam tabulek

2.1 Datová sada HDFS logů použitá pro model DeepLog 45
2.2 Detail datové sady Lastline, kategorizovaný podle úrovně rizika . . . 47
2.3 Porovnání DeepCASE s modely DeepLog a Tiresias 47
2.4 Preciznost modelů Tiresias v závislosti na datu trénování 49
3.1 Rozdělení souborů s logy událostí na jednotlivé kategorie 54
3.2 Vytvoření scénářů pro model BERT 67
4.1 Shrnutí učení modelu BERT v jednotlivých scénářích 73
4.2 Shrnutí testování modelu BERT v jednotlivých scénářích 73

Úvod

V dnešní době lze konstatovat, že každým dalším dnem celý svět využívá stále více a více prostředků informačních a komunikačních technologií. Ať už se jedná o chytrá mobilní zařízení, či počítače, je pro ně společná jedna věc. Využívají internet a jsou součástí kybernetického prostoru. Kybernetický prostor v tomto smyslu znamená nepřetržité nebezpečí. Lze se v rámci něj setkat s podvodníkem, který se např. snaží ze své oběti vylákat citlivé informace (jako jsou čísla bankovních účtů, číslo občanského průkazu, přihlašovací údaje do internetového bankovnictví a mnoho dalších). V tomto případě se jedná o tzv. sociální inženýrství a neexistuje 100 % účinná metoda, jak se proti němu bránit. Dalo by se konstatovat, že prakticky ihned po narození se člověk stává součástí kybernetického prostoru, protože již v této fázi jeho života o něm vzniká digitální záznam. Existuje mylná domněnka, že pokud se někdo vyvaruje užívání internetu, není součástí kybernetického prostoru. Být součástí kybernetického prostoru s sebou přináší spoustu rizik. Například lze tvrdit, že jakmile se informace stane součástí kybernetického prostoru, může se někdo se špatnými úmysly pokusit o narušení její důvěrnosti, integrity a dostupnosti. Z tohoto důvodu je důležité uvědomit si, s jakými informacemi a jak se v rámci kybernetického prostoru nakládá. Proto je důležité mít zajištěná technická opatření (např. antivirus) a jejich efektivitu podpořit organizačními opatřeními (např. vyvarování se nezabezpečenému připojení). Nejúčinnější obranou je v tomto případě dobrá digitální gramotnost, tedy orientace v kybernetické bezpečnosti a správné návyky a patřičná úroveň bezpečnostního povědomí konkrétního uživatele telekomunikačních technologií (např. nesdělovat nikomu své osobní údaje/přihlašovací údaje, nevyužívat služeb *Free WiFi* pro práci s citlivými údaji atd.). Vždy bude existovat někdo, kdo se nějakým způsobem snaží o narušení bezpečnosti, ať už s motivací vlastního obohacení, či jen pro poškození reputace oběti. S nástupem nových trendů, jako je internet věcí IoT (*Internet of Things*) nebo Průmysl 4.0 (*Industry 4.0*) a tedy postupnou digitalizací všech možných oblastí, je velice těžké bránit se čím dál tím více sofistikovaným útokům. Obecně lze konstatovat, že se v oblasti kybernetické bezpečnosti útočník snaží narušit jeden či více atributů, CIA (*Confidentiality, Integrity, Availability*) triády, dat nebo služeb v rámci kybernetického prostoru. Na rozdíl od sociálního inženýrství, se v tomto případě lze mnohem účiněji chránit a buď incidentu předejít, zpomalit jej, anebo alespoň zredukovat ztráty s pomocí bezpečnostního monitoringu. Bezpečnostní monitoring je v dnešní době velice užitečným nástrojem ať už pro odhalování všech různých anomálií v monitorované síťové infrastruktuře, či v rámci zvládání kybernetických bezpečnostních událostí či incidentů. Včasné odhalení konkrétního incidentu je v tomto případě kritickým aspektem pro jeho zvládání. Velice účinným nástrojem bezpečnostního monitoringu je management bezpečnost-

ních informací a událostí SIEM (*Security Information and Event Management*). SIEM funguje tak, že shromažďuje data protokolů a událostí generovaná aplikacemi, bezpečnostními zařízeními a systémy konkrétní organizace a sdružuje je do jediné centralizované platformy. SIEM shromažďuje data z antivirových prostředků, firewallu a dalších různých zdrojů a třídí tato data do kategorií. Když SIEM identifikuje hrozbu, vygeneruje upozornění a definuje úroveň hrozby na základě předem stanovených pravidel. Ač je SIEM v oblasti kybernetické bezpečnosti velice účinným nástrojem, nese spolu s sebou celou řadu nevýhod. Lze konstatovat, že jeho nejpodstatnější nevýhodou je závislost na lidech. Ačkoliv je při jeho implementaci zajištěná jistá míra automatizace, stále to pro plně automatické fungování nestačí. Proto je žádoucí nalézt metodu, pomocí které by bylo možné tuto míru automatizace zlepšit.

Cílem této diplomové práce je nastudování problematiky systémů SIEM v oblasti bezpečnostního monitoringu a vytyčení silných a slabých stránek. Následně provést analýzu a aplikaci metod využití umělé inteligence ve prospěch zdokonalení těchto systémů ve smyslu ulehčení práce bezpečnostním pracovníkům v rámci kybernetických operačních center. V případě technologií bezpečnostního monitoringu se sice jedná o kritický prostředek k zajištění CIA, tyto technologie jsou ovšem do jisté míry silně závislé na lidech. Tento nedostatek lze však poměrně jednoduše obejít, a to nashromážděním potřebných dat pro určitou úlohu a naučení modelu neuronové sítě.

Diplomová práce je rozdělena na čtyři části. První dvě kapitoly jsou zaměřeny na popis teorie systémů SIEM v oblasti bezpečnostního monitoringu a popis oblasti bezpečnostního monitoringu obecně a následné nastudování metod umělé inteligence a vytyčení možných způsobů, pomocí kterých by se dalo systémy SIEM zefektivnit. Jsou zde popsány silné a slabé stránky sysémů SIEM a také, pro porovnání, popsány podobné systémy SOAR (*Security Orchestration, Automation and Response*). Poslední dvě části jsou zaměřeny na definici problému, který lze vyřešit pomocí umělé inteligence. Byla provedena rešerše na možné aplikace umělé inteligence v oblasti bezpečnostního monitoringu a z této rešerše jsou vyvozena možná vylepšení stávajících řešení. Dále je specifikována úloha řešitelná pomocí zpracování přirozeného jazyka NLP (*Natural Language Processing*), konkrétně klasifikace sekvencí textu. Tato metoda je velice účinná pro zpracování velkého množství textových dat, která nejsou ve strojovém formátu. Existuje mnoho *state-of-the-art* modelů neuronových sítí, které velice účinně zvládají tento problém (porozumnění lidskému jazyku) řeší. Dále je v tomto případě výhodou, že lze použít modely tzv. přeneseného učení (*Transfer learning*), které jsou již na určité úlohy předpřipravené a stačí je jen přeučit (*fine tuning*) na určitý zvolený problém. Dále je popsána volba modelu neuronové sítě a jeho vlastnosti. Pro tyto účely byl zvolen model `Google BERT`, který je velice dobrým kompromisem mezi *state-of-the-art* výsledky a zároveň nízkou dobou

učení. Velice důležitou částí je také charakteristika dat potřebných k trénování, validaci a testování zvoleného modelu. Je proveden výběr vhodných dat pro tento úkol, shromáždění dat a jejich charakteristika. Po nasbírání všech nutných podkladů je provedeno předzpracování dat pomocí skriptu v jazyku Python (vývojové prostředí *PyCharm*), který data upravil do podoby vhodné k následnému zpracování před samotným učením modelu `BERT`. Tohle konečné zpracování je detailně popsáno a je již provedeno v prostředí *Colab Notebook*, které slouží jako účinné virtuální prostředí pro potřeby strojového učení. V poslední části diplomové práce jsou probrány výsledky měření a také některé problémy v rámci práce s modelem neuronové sítě.

1 Problematika monitoringu v oblasti kybernetické bepečnosti

Kybernetická bezpečnost, odvětví informační bezpečnosti[1], je široce užívaný termín, jehož definice mohou být odlišné, často subjektivní a neinformativní. Pojem kybernetická bezpečnost lze chápat jako souhrn právních, organizačních, technických a vzdělávacích prostředků směřujících k zajištění ochrany kybernetického prostoru[2]. Jedná se o soustavnou činnost, jejímž cílem je zajištění důvěrnosti, integrity a dostupnosti[3] informací a dat v rámci kybernetického prostoru. Na druhé straně ovšem vždy existuje někdo se špatnými úmysly, kdo se bude snažit tuto bezpečnost narušit, ať už z důvodu obohacení, dosažení konkurenční výhody, poškození reputace zasaženého subjektu, anebo s jinou motivací k protiprávnímu jednání. Může se jednat o jednotlivce nebo organizovanou skupinu útočníků, kteří usilují o odhalení, modifikaci, či zničení[4] informací a dat.

Dále je nutné zmínit pojem monitoring, což znamená kontrolovat či ověřovat, v tomto případě bezpečnost a míru rizik, a je tedy nedílnou součástí kybernetické bezpečnosti. Může se jednat o monitoring ad hoc, tedy pro jediný, konkrétní případ (pouze, pokud je to nutné). Dalším případem je nepřetržitý monitoring, při kterém je kladen důraz na všeobecný přehled o bezpečnosti a je prováděn nepřetržitě. Monitoring nespočívá jen v zajišťování kontroly v reálném čase, ale může se soustředit i na analýzu incidentů z minulosti a tím přispět k lepším bezpečnostním opatřením v případě následujících útoků.

1.1 Účel bezpečnostního monitoringu

S neustálým technologickým pokrokem a s nástupem digitalizace se dramaticky stupňuje míra kybernetických útoků. Tradiční způsob ochrany typu antivirus už dlouho nestačí – tohle již neplatí jen pro velké korporátní společnosti. Útoky mohou cílit na malé podniky, ale dokonce i na samotné koncové uživatele v rámci jejich domácností. Míra sofistikovanosti útoků se stále vyvíjí směrem kupředu a nadále rostoucí

[1] Informační bezpečnost se zabývá problematikou ochrany informací v jakékoliv podobě během jejich vzniku, zpracování, ukládání, přenosu a likvidace [1].

[2] Dle zákona č. 181/2014 Sb., o kybernetické bezpečnosti, se kybernetickým prostorem rozumí digitální prostředí umožňující vznik, zpracování a výměnu informací, tvořené informačními systémy, a službami a sítěmi elektronických komunikací.

[3] Jedná se o tzv. CIA triádu – confidentiality (důvěrnost), integrity (integrita), availability (dostupnost).

[4] Tzv. DAD triáda – disclosure (odhalení), alternation (modifikace), destruction/denial (zničení/odepření). Jde o protějšek CIA triády, kde lze říci, že obránce usiluje o CIA a útočník se snaží o DAD.

ztráty převyšují miliardy dolarů ročně. Realita je taková, že s rostoucí závislostí na připojení k internetu v kombinaci s rapidní expanzí malware[5] je stále těžší zabránit incidentům, či aspoň snížit jejich dopad. Nejčastějšími důvody pro úspěch takových útoků jsou:

- nízká míra znalosti bezpečnostních zásad zaměstnanců/koncových uživatelů;
- nedostatečná obezřetnost (např. bezmyšlenkovité odsouhlasení čehokoliv bez znalosti následků);
- nedodržování základních bezpečnostních zásad (slabá/žádná logická segmentace sítě, absence záloh, výchozí/slabá hesla administrátorských účtů atd.);
- nedostatečné finanční prostředky pro nasazení bezpečnostního monitoringu.

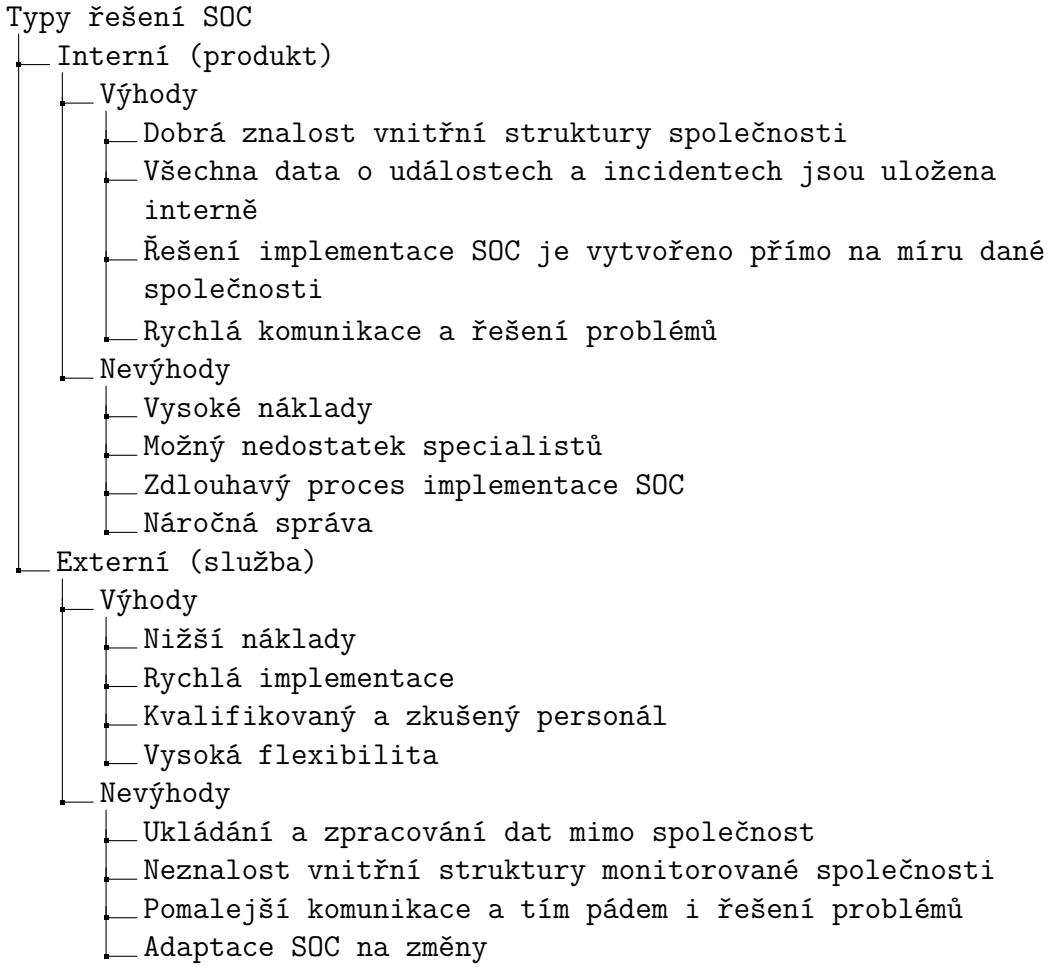
Co se týče vývoje antivirových programů, jedná se o velice komplexní proces. Jejich tvůrci musí nejprve zjistit, jak konkrétní malware funguje, než mohou zapracovat instrukce pro jeho spolehlivou detekci. To může trvat pouze pár hodin, ale i několik dnů, až týdnů. Autoři malware jsou si tohoto faktu samozřejmě vědomi. Důvodem neustálého vývoje nového malware je jeho omezená efektivita, jež se odvíjí od rychlosti adaptace antivirových programů. Zpočátku se dodavatelům antivirových programů s vývojem nových kybernetických hrozeb dařilo držet krok, ale s postupem času se tento úkol stává prakticky nemožným, protože útočníci jsou v tomto ohledu vždy o krok napřed [2].

Bezpečnostní monitoring zahrnuje shromažďování a analýzu informací s cílem odhalit neobvyklé chování nebo neoprávněnou aktivitu v síti. Pro minimalizaci škod je kritické, aby se na incident přišlo v době, kdy jsou ještě způsobené škody buď minimální nebo žádné a problém lze poměrně jednoduše řešit[6]. V drtivé většině případů, pokud je incident detekován tím způsobem, že už vyjdou najevo jím způsobené škody, je jeho dopad výrazně horší a pro zasaženou organizaci to může znamenat fatální následky.

Spolehlivým řešením monitoringu jsou kybernetická operační centra SOC (*Security Operation Center*). SOC lze definovat jako centralizovanou bezpečnostní organizaci, která pomáhá společnostem s identifikací a nápravou kybernetických bezpečnostních incidentů. Cílem SOC je zlepšit bezpečnost organizace, co se týče detekce a reakce na hrozby a útoky dříve, než budou mít dopad na kontinuitu podnikání. SOC může být cílové organizaci poskytnut jako produkt nebo formou služby. Oba přístupy mají své výhody a nevýhody [4] (viz obr. 1.1).

[5]Z anglického *malicious software*, což je možné přeložit jako škodlivý počítačový program.

[6]MTTD (*Mean Time to Detect*) a MTTR (*Mean Time to Respond*) jsou dvě velice důležité metriky co se týče kybernetických bezpečnostních incidentů. MTTD je průměrný čas, který je třeba k detekci incidentu a MTTR průměrná doba potřebná k reakci na tyto incidenty a jejich řešení [3].



Obr. 1.1: Porovnání interních a externích řešení SOC

V rámci bezpečnostního monitoringu přijímají pracovníci (operátoři, analytici atd.) SOC hlášení o kybernetických bezpečnostních incidentech (tzv. *alerty*) a očekává se od nich, že tyto incidenty, nahlášené bezpečnostními nástroji nebo osobami, budou analyzovat a řešit. Pro tyto případy musí být vytvořen tzv. *tiket*, který bude sledován až do jeho vyřešení, aby bylo zajištěno, že incident bude řádně prošetřen a zvládnut. K tomu je třeba správných nástrojů, dostatečných pravomocí a zajištění integrace s procesy reakce na incidenty a správy případů [5].

1.2 Analýza datových toků v síti

V průběhu let byly navrhovány a vyvíjeny přístupy analýzy datových toků v síti, přičemž každý z nich sloužil jinému účelu. Obecně je lze rozdělit do dvou kategorií: aktivní a pasivní. Aktivní přístupy, například implementované nástroji jako *ping* a *traceroute*, generují provoz v síti za účelem provádění různých typů měření. Pasivní

přístupy sledují provoz v síti. Jedním z pasivních monitorovacích přístupů je zachycování paketů[7]. Nicméně ve vysokorychlostních sítích vyžaduje zachycování paketů drahý hardware a složitou infrastrukturu pro ukládání datových toků a jejich analýzu. Další variantou pasivní analýzy datových toků je *flow export*, ve kterém jsou pakety agregovány do toků a exportovány pro ukládání a analýzu [6]. Datový tok je definován jako sada IP (*Internet Protocol*) paketů procházejících pozorovacím bodem v síti za určitý časový interval, takže všechny pakety patřící do určitého toku mají sadu společných vlastností. Tyto společné vlastnosti mohou zahrnovat pole záhlaví paketů, jako jsou zdrojové a cílové IP adresy a čísla portů, obsah paketů a metadata. Počáteční práce na *flow export* sahají až do devadesátých let 20. století a staly se základem pro moderní protokoly, jako je NetFlow a IPFIX (*IP Flow Information eXport*) [6].

Vzhledem k tomu, že zařízení pro export datových toků jsou běžně nasazována na strategických bodech v síti, kde lze sledovat provoz z velkého množství zařízení, což umožňuje vytvářet analýzy zachycených dat, které jsou velice užitečné pro detekci neobvyklého chování. Při použití datových toků pro detekci hrozeb lze rozlišit dva základní typy použití [6]:

- datové toky mohou být použity čistě pro analýzu komunikace mezi dvěma zařízeními: *peer-to-peer*/*client-server* (tedy komunikace typu klient-klient nebo klient-server) pro účely tzv. forenzní analýzy, případně včetně souhrnů počtu zahrnutých paketů a bajtů, počtu spojení atd.;
- druhý typ analýzy datových toků se zaměřuje přímo na analýzu určitých typů hrozeb, což umožňuje detekovat tyto hrozby na základě chování sítě.

Díky využití centrálních pozorovacích bodů je analýza datových toků účinná pro detekci např. DDoS útoků, skenování sítě, šíření škodlivého kódu a komunikace botnetů. Společným rysem těchto útoků je, že ovlivňují metriky, které lze přímo odvodit ze záznamů o datovém toku, jako je objem provozu, počet aktivních datových toků v určitém časovém intervalu, podezřelá čísla portů či podezřelý cíl komunikace. Identifikace podezřelých cílových uzlů je obvykle prováděna na základě databáze známých, škodlivých IP adres (*black lists*), či na základě reputačních databází. V těchto databázích lze narazit na IP adresy, které byly identifikovány jako zdroje škodlivých aktivit (např. odesílání SPAM, hostování malware nebo podíl na infrastruktuře botnetu). Tato technika však neidentifikuje pouze klasické hrozby, ale může také pomoci i při odhalování pokročilých trvalých hrozeb APT [6] (*Advanced Persistent Threat*[8]).

[7]Tato metoda obecně poskytuje největší přehled o síťovém provozu, protože lze zachytit a dále analyzovat celé pakety.

[8]APT jsou útoky/organizované skupiny útočníků, které kombinují vysoký stupeň utajení, dlouhodobé plánování a množství vektorů útoku (tedy např. využití více zranitelností najednou a tím pádem dosažení mnohem efektivnějšího útoku). Obvykle se zaměřují na vládní a komerční subjekty.

Analýza datových toků je velice účinným nástrojem pro detekci anomálií, které mohou vést ke kybernetickému bezpečnostnímu incidentu, a je tedy nedílnou součástí bezpečnostního monitoringu. Podrobný popis analýzy datových toků a protokolů NetFlow a IPFIX je možné vyčíst z literatury [6].

1.3 Správa logů událostí

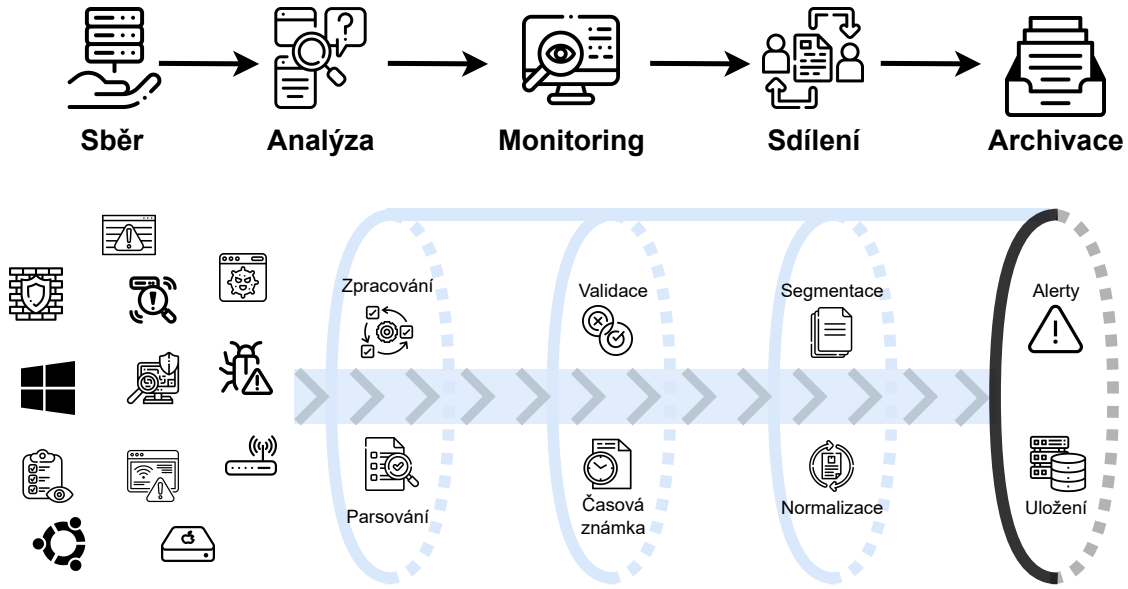
Log události je počítačem generovaný soubor, který zachycuje aktivitu v rámci operačního systému nebo softwarových aplikací. Může být také generován hardwarovým zařízením. Automaticky dokumentuje veškeré, předem nastavené, informace a také může být opatřen časovým razítkem, což pomáhá zejména bezpečnostním analytikům porozumět tomu, co se stalo a kdy se to stalo. Logy se skládají z jednotlivých záznamů, kde každý záznam obsahuje informace související s konkrétní událostí, která se stala v systému, nebo v síti. Původně byly logy používány především k řešení mimořádných problémů. Nyní mohou v organizacích poskytovat celou řadu funkcí, jako je například optimalizace výkonu sítě, zaznamenávání aktivit uživatelů a poskytování dat užitečných pro vyšetřování škodlivé činnosti. Mnoho logů obsahuje v rámci organizace záznamy související s kybernetickou bezpečností – zejména se může jednat o podezřelou aktivitu [7] (např. neúspěšné pokusy o autentizaci uživatele, velké přenosy dat, skenování otevřených portů atd.). Logy jsou kritické pro možnost detekce kybernetických bezpečnostních incidentů. Zejména pro zjištění zdroje hrozby, jaké systémy byly zasaženy a pro identifikaci toho, co se přesně stalo, aby bylo možné porozumět chování systému v daném okamžiku a podniknout potřebná opatření pro řešení incidentu.

Proces logování

Shromažďování a archivace logů je nezbytnou praxí pro každou organizaci, která chce udržovat výkon a bezpečnost své sítě. Proces logování (viz obr. 1.2) se v některých případech může lišit v závislosti na preferencích konkrétní organizace a může vypadat následovně [8]:

1. **Sběr** – základním úkonem ve správě logů událostí je jejich sběr z různých zdrojů, který v sobě zahrnuje nějaký druh normalizace (úprava na stejný formát) a parsování (strukturování dat pro lepší přehlednost). Proces shromažďování dat může zahrnovat také indexování a kompresi. Indexování je proces rozdělení bloků protokolu na segmenty a jejich následné uspořádání do datové mapy za účelem pozdějšího vyhledávání důležitých informací. Komprese pomáhá snížit nároky na výpočetní výkon a tím pádem může urychlit vyhledávání ve fázi analýzy;

2. **Analýza** – po shromáždění jsou logy analyzovány za účelem identifikace problémů s výkonem, chyb aplikací a bezpečnostních hrozeb. Typickým případem může být obdržení upozornění bezpečnostním analytikem, který poté přejde k vyhledávání souvisejících logů, za účelem zjištění hlavní příčiny události, která upozornění vyvolala;
3. **Monitorování** – sledování chyb a narušení v reálném čase. Téměř všechny součásti softwaru a infrastruktury generují logy, tudíž představují účinný prostředek, jak zjistit, co všechno se v systému děje v době incidentu, a je možné najít příčinu tohoto incidentu. Monitorování lze provádět pomocí dashboardů, které poskytují stručný přehled o aktuální situaci. Lze jej také realizovat nastavením výstrah, které upozorňují bezpečnostní analytiky na konkrétní události (např. pomocí e-mailu či SMS zprávy). Proces monitorování je možné automatizovat přizpůsobením alertů funkcím systému nebo implementací pravidel, která mohou upozornit na anomálie nebo bezpečnostní hrozby;
4. **Sdílení** – nedílnou součást procesu správy logů událostí je právě sdílení zjištěných poznatků, aby se informace o možné hrozbě či incidentu dostala ke správnému člověku v rámci zasažené organizace;
5. **Archivace** – posledním krokem je archivace dat tak, aby je bylo možné později vyhledat k případné zpětné analýze již proběhlých incidentů. Tato funkce je kritickou v rámci automatizace procesu bezpečnostního monitoringu, jelikož hrozby a incidenty mají v určitých případech tendenci opakovat se.



Obr. 1.2: Proces logování [8]

Struktura logu

Hlavním problémem s logy je to, že se obvykle jedná o nestrukturovaná textová data, což ztěžuje zjištění jakékoli užitečné informace. Formát logu umožňuje, aby byly protokoly strojově čitelné a snadno analyzovatelné. Schopnost převést nezpracovaná data do něčeho, co je okamžitě srozumitelné a snadno čitelné, je jednou z nezbytných funkcí správy logů [9].

Syslog

Syslog je jedním z nejznámějších a také nejstarších formátů logů, který byl vyvinut v 80. letech 20. století. Umožňuje oddělení softwaru pro generování zpráv, systému pro jejich ukládání, a softwaru pro analýzu zpráv. Každá zpráva je označena speciálním kódem, který označuje typ systému generující zprávu a zároveň je zprávě přiřazena úroveň závažnosti. Syslog je možné využít pro správu systému, bezpečnostní audit a obecné informační a analytické zprávy. Syslog využívá komunikaci typu klient-server, kde syslog server naslouchá a zaznamenává zprávy přicházející od klientů [10].

JSON

JSON (*JavaScript Object Notation*) je formát pro výměnu dat, který je jednoduchý pro zápis a čtení pro lidi i stroje. Může být analyzován téměř všemi programovacími jazyky, dokonce i těmi, které nemají vestavěnou funkci JSON. Jedná se o univerzální formát díky kódování *Unicode* [9].

Windows event log

Windows event log zachycuje a ukládá operační systém Windows. Tyto záznamy obsahují informace o operačním systému, aplikaci a bezpečnostní upozornění. Správci systému tyto události obvykle používají k diagnostice potenciálního problému a k prevenci možných budoucích problémů. Struktura logu je následující [9]:

- datum a čas události;
- uživatelské jméno v rámci operačního systému;
- název počítače;
- identifikační číslo;
- zdroj události (program, který událost způsobil);
- typ události (informace, varování, chyba, úspěch či selhání bezpečnostní kontroly).

CEF

CEF (*Common Event Format*) je formát logů, který vyvinula společnost ArcSight. Jedná se o textový, snadno čitelný formát. CEF byl vytvořen jako běžný standard logů událostí, lze ho tedy využít ke snadnému sdílení informace o zabezpečení pocházející z různých síťových zařízení, aplikací a nástrojů. Log ve formátu CEF může vypadat následovně [11]:

```
Oct 12 04:16:11 localhost CEF:0|nxlog.org|nxlog|2.7.1243|Executable
Code was Detected|Advanced exploit detected|100|src=192.168.255.110
spt=46117 dst=172.25.212.204 dpt=80
```

Význam jednotlivých parametrů uvedeného logu [12]:

- **ReceiptTime** – `Oct 12 04:16:11` – datum a čas;
- **Host** – `localhost` – název uživatele;
- **CEF:Version** – `CEF:0` – verze CEF;
- **Device Vendor** – `nxlog.org` – dodavatel zařízení;
- **Device Product** – `nxlog` – typ odesílajícího zařízení;
- **Device Version** – `2.7.1243` – verze zařízení;
- **Device Event Class ID** – `Executable Code was Detected` – typ události;
- **Name** – `Advanced exploit detected` – popis události;
- **Severity** – `100` – závažnost události;
- **Extension** – `src=192.168.255.110 spt=46117 dst=172.25.212.204 dpt=80` – doplňující informace (src – IP adresa zdrojového zařízení, spt – zdrojový port, dst – IP adresa cílového zařízení, dpt – cílový port).

CLF

CLF (*Common Log Format*) je pevný (nepřizpůsobitelný) formát používaný webovými servery při generování logů serveru. Každý řádek v tomto formátu protokolu je uložen pomocí následující standardizované syntaxe [13]:

```
125.125.125.125 - dsmith [10/Oct/1999:21:15:05 +0500]
"GET /index.html HTTP/1.0"200 1043
```

Význam jednotlivých parametrů uvedeného logu [13]:

- **Host** – `125.125.125.125` – IP adresa nebo doménové jméno zdroje;
- **Ident** – `"-"` – identifikace klienta (pokud není hodnota k dispozici, pole je nahrazeno znakem `"-"`);
- **UserID** – `dsmith` – jméno uživatele (pokud není hodnota k dispozici, pole je nahrazeno znakem `"-"`);

- **Date:time timezone** – `[10/Oct/1999:21:15:05 +0500]` – datum, čas a časová zóna;
- **Request** – `"GET /index.html HTTP/1.0"` – HTTP požadavek obsahující metodu HTTP, požadovaný zdroj a verzi HTTP protokolu;
- **Statuscode** – `200` – stavový kód HTTP;
- **Bytes** – `1043` – množství přenesených dat.

1.4 Management bezpečnostních informací a událostí

Míra bezpečnostních rizik za posledních pár let enormě vzrostla, protože útočníci používají sofistikovanější metody a je tedy čím dál tím těžší odhalit kybernetické bezpečnostní incidenty včas. Tyto incidenty jsou v drtivé většině případů zapříčiněny lidskou chybou[9] a velice často se jedná o cílené sociální inženýrství, které může vést k odcizení citlivých informací nebo k infekci celé sítě zasažené společnosti škodlivým programem typu ransomware. Pro včasnou detekci a možnost rychlého řešení incidentů již v jejich počátku, dokud ještě nebyly napáchány velké škody, je velice účinným řešením systém SIEM (*Security Information and Event Management*). Funkce SIEM spočívá v shromažďování, agregování, ukládání a korelování logů událostí zachycených různými způsoby – tzv. LES (*Log Event Source*), neboli zdroje logů událostí. Jako LES mohou figurovat následující nástroje [15]:

- **IDS (*Intrusion Detection System*)** – slouží k detekci a hlášení kybernetických bezpečnostních incidentů. IDS mohou fungovat na bázi [16]:
  - Detekce signatur (*Signature-based*), která probíhá na základě již známých metod útoků např. pomocí databází hrozeb (*Threat intelligence database*);
  - Detekce anomálií (*Anomaly-based*), kde je monitorováno chování uživatelů, či přenosů dat na síti a pomocí pravidel jsou definovány odchylky od normálního chování nebo stavu sítě;
  - Analýza stavových protokolů (*Stateful protocol analysis*) umožňující detekci neobvyklé sekvence požadavků u určitého protokolu: např. FTP (*File Transfer Protocol*) je stavový protokol, protože se jeho chování odvíjí od předešlých, již provedených, požadavků.

  Metody IDS jsou limitovány na detekci a není pomocí nich tedy možné problém přímo řešit;
- **Antivirus** – počítačový program, který skenuje příchozí soubory nebo zdrojový kód. Ve většině případů, stejně jako IDS, fungují antivirové programy na bázi detekce signatur. Porovnávají tedy příchozí tok dat s databází známých

[9] Dle studie společnosti IBM je na vině v až 95 % úspěšných útoků lidská chyba [14]

hrozeb. Jediný rozdíl je v tom, že pokud detekují hrozbu, mají schopnost se s ní vypořádat bez nutnosti jakékoliv další akce. Nevýhoda samozřejmě spočívá v neschopnosti detekce neznámých hrozeb;

- **Firewall** – umožňuje pomocí předdefinovaných či dynamických pravidel blokovat nevhodný příchozí i odchozí síťový provoz;
- **Skener zranitelností** – zranitelnost je neúmyslná chyba[10], která byla způsobena při vývoji software či hardware. Zneužití takové chyby se nazývá *exploit* a může útočníkovi poskytnout značnou výhodu při jeho snaze o odcizení dat, získání kontroly nad zařízením, eskalaci privilegií atd. Již odhalené zranitelnosti jsou dokumentovány v databázi CVE (*Common Vulnerabilities and Exposures*) a mohou poskytnout vývojářům kritické informace o chybách v jejich produktu, které lze následně opravit prostřednictvím aktualizace;
- **Honeypot** – může se jednat o úmyslně podstrčený informační systém, který je náchylný k útokům proto, aby na sebe přitáhl pozornost útočníků (kupříkladu tím, že v sobě má zakomponovány lehce zneužitelné zranitelnosti). Pokud je na honeypot realizován útok a je úspěšný, lze analyzovat chování útočníka či škodlivého programu a tím zdokonalit zabezpečení reálné infrastruktury.

Bez systému SIEM by pro pracovníky SOC bylo prakticky nemožné analyzovat všechna možná bezpečnostní oznámení z různých LES[11] a hlavně rozlišit, zda se jedná o skutečně kritické události, nebo naopak o falešný poplach. Pro pochopení, jak vše funguje, bude následovat popis životního cyklu SIEM [17]:

1. **Detekce hrozeb** – neustálé zpracovávání všech událostí z různých LES a hledání možných hrozeb;
2. **Agregace, filtrace, normalizace a korelace** – tyto procesy musí proběhnout po identifikaci každé možné hrozby. Všechna relevantní surová data (logy událostí) jsou tedy zachycena a následně označena, převedena na stejný formát. SIEM může později automaticky klasifikovat, o jak kritické bezpečnostní riziko se jedná;
3. **Prioritizace** – jedná se o konfigurovatelnou funkci, která umožňuje definovat závažnost události pro informaci, na co se zaměřit nejdříve. Lze rozlišovat 3 základní kategorie závažnosti:
    - Anomálie – může se jednat pouze o falešný poplach nechtěně způsobený legitimním uživatelem, ale také o podezřelé chování útočníka, je tedy nutné takové události přezkoumat;
    - Bezpečnostní riziko – událost, která sice nemusí znamenat okamžitý problém, ale k předejití většího problému je nutné ji co nejdříve řešit;

[10] V případě nutnosti, např. umožnění přístupu k systému v krajní nouzi, mohou vývojáři do jejich produktu zakomponovat i úmyslnou chybu (tzv. *Backdoor*).

[11] Některé systémy SIEM mají schopnost číst a interpretovat události až z několika stovek LES.

- Bezpečnostní incident – bezprostřední bezpečnostní hrozba nebo kybernetický útok, což vyžaduje okamžitý zásah bezpečnostních týmů.

4. **Eskalace** – vytvoření varování, které může být zasláno konkrétní zodpovědné osobě [12], anebo díky určité konfiguraci, může SIEM reagovat automaticky (např. zablokování uživatelského účtu při více neúspěšných pokusech o přihlášení);
5. **Analýza** – z každé události se ukládají surová data a SIEM může z těchto dat generovat zprávy s různou mírou podrobnosti, které dokumentují životní cyklus události. Bezpečnostní týmy mohou tyto zprávy opatřit poznámkami obsahujícími analýzu konkrétní události a také doporučení v případě podobné události –> budoucí reakce na incident může být tedy efektivnější a rychlejší;
6. **Archivace** – surová data označená při výskytu bezpečnostní události jsou uchovávána v záznamech po určitou dobu a hlášení generované systémem SIEM vytvářejí kompletní auditní stopu, která může organizaci poskytnout značnou výhodu při analýze incidentů z minulosti.

Ačkoli systémy SIEM poskytují výrazně jednodušší cestu k řešení bezpečnostních hrozeb, největší výhodou, kterou poskytují, je úspora času.

ArcSight ESM

ArcSight ESM (*Enterprise Security Manager*) je vhodným nástrojem pro zlepšení bezpečnostního monitoringu v organizaci, kde pro tento úkol nestačí klasické nástroje. Jako každý jiný SIEM je ArcSight ESM užíván pro správu zabezpečení na všech koncových zařízeních. Poskytuje řadu funkcí, které pomáhají rychle kontrolovat a snadno řešit bezpečnostní události a incidenty na všech koncových zařízeních. ArcSight ESM slouží především k integraci všech nástrojů pro správu bezpečnosti koncových zařízení, ať už se jedná o IPS, IDS, Firewall, Antivirus atd., a pomáhá omezit nadbytečná a falešná upozornění, která nemusí být z hlediska bezpečnosti užitečná, a tím pomáhá efektivním způsobem rychle kontrolovat velké množství zařízení [18].

Je také užitečný pro kompletní kontrolu aktivity odehrávající se v monitorované síti, vytváření vlastních pravidel a filtrů a vytváření přehledů aktivních kanálů, které pomáhají udržet ostražitost v případě, že se na některém zařízení vyskytne nějaká potenciálně škodlivá událost.

[12] Např. pokud dojde k detekci viru na Linux serveru, je upozornění posláno Linux specialistovi, který může problém neprodleně vyřešit.

RSA NetWitness Evolved SIEM

SIEM od společnosti RSA NetWitness umožňuje bezpečnostním týmům detekovat hrozbu a porozumět celému rozsahu kompromitace, protože analyzuje data a chování napříč celou organizací: protokoly, pakety a koncové body, stejně jako NetFlow[13] aktivity generované lidmi a procesy. Data jsou následně transformována na užitečnější informace prostřednictvím operací v reálném čase. RSA NetWitness SIEM využívá jednotnou taxonomii napříč transformovanými daty, aby se urychlila detekce známých i neznámých hrozeb. neznámých hrozeb. RSA NetWitness SIEM nabízí výkonné funkce založené na strojovém učení, analýze chování uživatelů a entit (UEBA) a pokročilých informací o hrozbách (*Threat Intel*) [19].

RSA NetWitness SIEM poskytuje bohaté možnosti pro činnosti detekce hrozeb a reakce na ně a také flexibilní modely nasazení (cloud SIEM, virtualizovaný SIEM nebo fyzické zařízení), které podporují moderní IT infrastruktury. Tato komplexní a flexibilní platforma umožňuje RSA NetWitness výrazně optimalizovat procesy detekce hrozeb a reakce na ně a umožňuje bezpečnostním analytikům mnohem efektivněji chránit své organizace před pokročilými kybernetickými hrozbami. Mezi klíčové funkce systému RSA NetWitness SIEM tedy patří [19]:

- jediné řešení, které kombinuje analýzu detekce hrozeb a reakci na ně s monitorováním protokolů a událostí, telemetrií koncových bodů, vyšetřováním a funkcemi pro analýzu hrozeb ve všech datech;
- okamžité informace o datech z nových a neznámých zdrojů bez nutnosti vlastních parserů dat, či nadbytečného programování;
- integrovaný kontext hrozeb, díky kterému mohou organizace určité hrozby prioritizovat na základě potenciálního dopadu na kontinuitu byznysu;
- automatická analýza chování s využitím nástroje *Advanced Analytics Engine*, který vyhledává potenciálně škodlivé problémy v protokolech a NetFlow a také koreluje data napříč síťovými pakety a koncovými body – což jsou hlavní vektory útoku pro dnešní pokročilé hrozby;
- pokročilé prostředí pro třídění alertů a incidentů, včetně rozhraní navrženého speciálně pro bezpečnostní vyšetřování;
- možnost nativně a vizuálně rekonstruovat síťový útok nebo exfiltraci[14] dat.

IBM QRadar SIEM

IBM QRadar SIEM je vysoce škálovatelné podnikové řešení, které sjednocuje zdrojová data událostí protokolů z tisíců zařízení rozmístěných v síti, ukládá každou ak-

[13]Protokol od společnosti Cisco vyvinutý k monitorování síťového provozu na základě IP toků.

[14]Vynesení informací organizace zaměstnanci, ale i útočníky, přičemž se nemusí vždy jednat o špatný úmysl.

tivitu do své databáze a následně provádí okamžitou korelaci a použití analytických nástrojů k rozlišení skutečných hrozeb od falešných poplachů. QRadar SIEM poskytuje dohled nad celou IT infrastrukturou a pomáhá organizacím odhalovat a napravovat hrozby, které jiná bezpečnostní řešení často přehlížejí. Mezi tyto hrozby může patřit např. nevhodné používání aplikací, podvody a další, včetně hrozeb, které se mohou ztratit v "šumu"milionů dalších událostí [20].

Mezi hlavní vlastnosti IBM QRadar může patří [20]:

- přehled o celé infrastruktuře IT v reálném čase pro detekci hrozeb a stanovení priorit;
- redukce a upřednostnění alertů, aby se vyšetřování bezpečnostních analytiků zaměřilo na použitelný seznam incidentů;
- efektivnější správa hrozeb a zároveň vytváření podrobné zprávy;
- možnost lokálního i cloudového prostředí;
- podrobné zprávy o přístupu k datům a aktivitě uživatelů;
- stovky vestavěných případů využití UCs (*Use Cases*), algoritmy detekce anomálií, pravidla a korelace v reálném čase;
- řešení určitých útoků jsou automaticky prioritizována na základě kritičnosti zasažených prostředků.

Elastic SIEM

Bezplatná a otevřená aplikace Elastic SIEM poskytuje bezpečnostním týmům přehled, vyhledávání hrozeb, automatickou detekci a pracovní postupy SOC. Elastic SIEM je součástí výchozí distribuce platformy pro protokolování, softwaru *Elastic Stack*. Dodává se s hotovými detekčními pravidly sladěnými s MITRE ATT&CK[15], která odhalují hrozby často přehlížené jinými nástroji. Tato pravidla, vytvořená, udržovaná a aktualizovaná bezpečnostními experty společnosti Elastic automaticky detekují a řeší nejnovější aktivity v oblasti hrozeb. Skóre závažnosti a rizika spojená se signály generovanými detekčními pravidly umožňují analytikům rychle třídit problémy a zaměřit svou pozornost na prioritní události [21].

Elastic SIEM je postaven na rychlosti a škálovatelnosti *Elasticsearch*[16], jako základní vyhledávací platformě, a jeho hlavními vlastnostmi jsou [21]:

- přehledová stránka, která zobrazuje stav SOC a stav zabezpečení;
- dashboardy pro vyhledávání hrozeb a celkové povědomí o situaci;

[15]Znalostní báze a model chování kybernetických útočníků, který odráží různé fáze životního cyklu útoku a platformy, na které se útočníci zaměřují.

[16]Fulltextový vyhledávač vycházející z *Apache Lucene*, který je šířen jako svobodný software (*free software*) a otevřený software (*open source*) pod licencí Apache.

- integrace s aplikacemi *Elastic Maps*, *Elastic Lens* a zbytkem systému *Kibana*[17];
- engine umožňující automatizovanou detekci;
- časové osy se šablonami vyšetřování pro analytiky.

Splunk Enterprise Security

Splunk ES (*Enterprise Security*) je řešení SIEM, které bezpečnostním týmům umožňuje rychle odhalovat interní a externí útoky a reagovat na ně, zjednodušit správu hrozeb a zároveň minimalizovat rizika a chránit organizaci. Splunk ES umožňuje bezpečnostním týmům využívat veškerá data k získání přehledu a bezpečnostních informací v rámci celé organizace. Bez ohledu na model nasazení – on-premise, ve veřejném nebo soukromém cloudu, SaaS nebo jejich kombinaci lze Splunk ES používat pro nepřetržité monitorování, reakci na incidenty, provozování SOC nebo pro poskytování informací vedoucím pracovníkům o rizicích organizace. Splunk ES lze nasadit jako software společně se *Splunk Enterprise* nebo jako cloudovou službu společně se *Splunk Cloud* [22].

1.5 Nedostatky systémů SIEM

Systémy SIEM jsou v dnešní době velice užitečné, dokonce prakticky nepostradatelné v každé větší společnosti. Bez nich by bezpečnostní týmy nemohly zvládat ochranu před bezpečnostními hrozbami, protože i přes mnoho LES by nebylo možné se vyznat ve všech událostech, které nastaly, natož rozpoznat, na jaké z nich je potřeba se zaměřit a jaké z nich je třeba neprodleně řešit. Pro představu, průzkum společnosti Cisco z roku 2019 [23] uvádí, že 41 % z 3540 zkoumaných organizací obdrží více než 10 000 alertů denně. Z těchto alertů bylo kvůli omezené kapacitě bezpečnostních operátorů prošetřeno pouze 50,7 % a pouze 24,1 % prošetřených alertů bylo považováno za skutečnou hrozbu. Přestože jsou dnešní systémy SIEM velice výkonné a užitečné, stále mají mnoho nedostatků [24]:

- **Závislost na lidech** způsobená omezenou možností automatické reakce na kybernetické bezpečnostní incidenty. Systémy SIEM se zaměřují na poskytování komplexní interpretace hrozeb. V mnoha případech však reakce na hrozby stále vyžadují, aby analýzu provedl člověk a učinil rozhodnutí týkající se definování vhodných protiopatření a jejich nasazení. Jedná se o pomalý a nákladný proces, který vyžaduje vysokou úroveň odborných znalostí a v jeho průběhu

[17] Otevřený software pro vizualizaci dat ve webovém prohlížeči – součást trojice **Elastic Stack**, dříve ELK (*Elasticsearch, Logstash, Kibana*) Stack.

je možné dopustit se mnoha chyb. Proto se současný výzkum v oblasti technologií SIEM zaměřuje na schopnost automatizovat proces výběru a nasazení protiopatření;

- **Korelační pravidla** je nutné psát ručně a není prakticky možné ošetřit veškeré varianty možných bezpečnostních incidentů. Navíc je proces psaní pravidel složitý a zdlouhavý. Jen velmi málo řešení SIEM má vestavěný pokročilý korelační mechanismus, který dokáže provádět analýzu korelací z minulosti;
- **Nedostatek dat** potřebných ke zpracování a odhalení všech bezpečnostních incidentů, a to i přesto, že současné systémy SIEM pracují s velkým množstvím dat. Důvodem je to, že z hlediska nákladů není efektivní zachytit a zpracovat všechna potřebná data. V důsledku absence podrobnější kontroly není možné zjistit spolehlivou detekci všech potřebných událostí;
- **Nesoulad se současnými nařízeními** jako je například GDPR (*General Protection Data Regulation*). V budoucí generaci systémů SIEM je tedy kritické, aby splňovaly požadavky na ochranu osobních údajů a zároveň analytikům poskytovaly dostatek informací pro řádnou identifikaci bezpečnostních incidentů;
- **Problém s využitím archivovaných dat** spočívající v neschopnosti automatického využití takových dat systémem SIEM. Navíc způsob, jakým se s archivovanými daty nakládá nebo kam se ukládají či přenášejí, závisí na určité konfiguraci operátory a obvykle se provádí ručně, protože existují různé možnosti, kam archivovaná data ukládat a na jak dlouho.

1.6 Orchestrace zabezpečení, automatizace a odezvy

Jak již bylo zmíněno v podkapitole 1.5, systémy SIEM mají velké množství nedostatků, z nichž tím nejkritičtějším je pravděpodobně značná závislost na lidské práci (zejména práce týkající se analýzy velkého množství dat). Tato závislost na lidech má za následek mnoho problémů, ať už po stránce finanční, či ve smyslu efektivity. Práce bezpečnostních analytiků v rámci SOC je sice do značné míry efektivní, ale také pomalá pro potřeby zpracování takového množství dat, což má za následek ignorování mnoha výstrah a incidentů [25]. SOAR (*Security Orchestration, Automation and Response*) je soubor kompatibilních softwarových programů, které umožňují shromažďovat data o bezpečnostních hrozbách a do jisté míry reagovat na bezpečnostní události bez nutnosti lidských zdrojů. Cílem používání platformy SOAR je zvýšit efektivitu operací fyzické a digitální bezpečnosti [26].

Důležitou součástí těchto systémů jsou tzv. *SOAR playbooky*, které je možné definovat jako soubor pravidel/akcí. Úkolem těchto playbooků je pomoci bezpečnostním

týmům urychlit a zefektivnit časově náročné procesy. Playbooky jsou vybaveny možnostmi integrace bezpečnostních nástrojů a vytvářením pracovních postupů, tím pádem umožňují bezpečnostním týmům automatizovat opakující se úkoly a tedy uvolnit bezpečnostní analytiky pro plnění důležitějších úkolů. Jelikož mnoho incidentů má tendenci se opakovat, jsou *SOAR playbooky* velice efektivním nástrojem pro automatické řešení incidentů, které již v minulosti proběhly [25, 27].

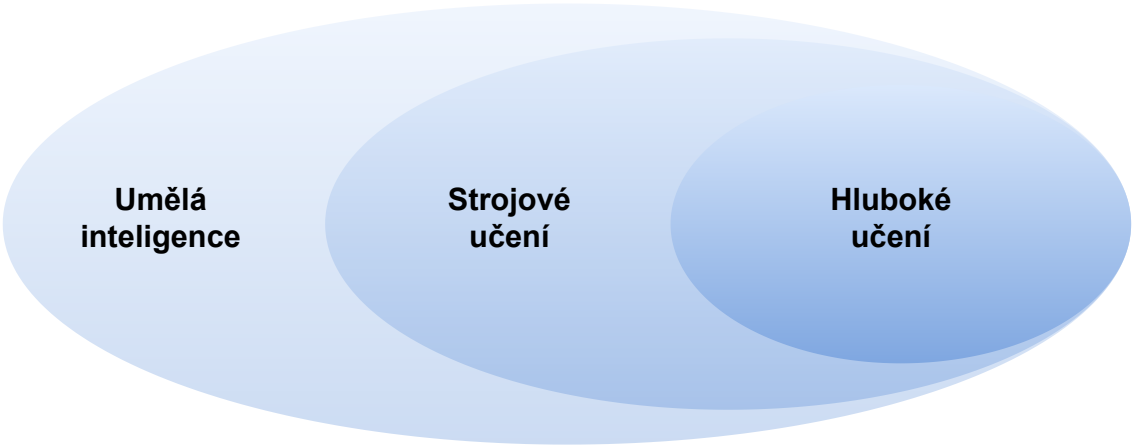
SOAR v sobě, jak název napovídá, spojuje tři věci [25, 26]:

- **Orchestrace** zahrnuje propojení a integraci různých interních a externích nástrojů, jako např. skenery zranitelností, produkty na ochranu koncových bodů EDR (*Endpoint Detection and Response*), analýzy chování koncových uživatelů UEBA (*User and Entity Behaviour Analytics*), firewally, systémy detekce a prevence (IDS/IPS), systémy SIEM, reputační databáze a černé listiny. Kombinace všech shromážděných dat z těchto zdrojů umožňuje vysokou efektivitu SOAR. Pro zpracování takového množství dat je ovšem nutná automatizace, díky které jsou podniknuty důležité kroky k analýze zjištěných hrozeb a také jejich rozeznání od falešných poplachů;
- **Automatizace** je založena na datech a upozorněních shromážděných z orchestrace, přijímá a analyzuje data a vytváří automatizované procesy, které nahrazují manuální procesy. Úlohy, které dříve prováděli bezpečnostní analytici, jako je skenování zranitelností, analýza protokolů, kontrola tiketů a možnosti zkoumání jednotlivých hrozeb/incidentů, mohou být pomocí SOAR automatizovány. Pomocí umělé inteligence a strojového učení k přizpůsobení poznatků od bezpečnostních analytiků může automatizace vytvářet doporučení a také může pomoci automatizovat budoucí odezvy na hrozby, či z velké části bezpečnostním analytikům s touto procedurou pomoci;
- **Odezva** nabízí analytikům jednotný pohled na plánování, řízení, monitorování a hlášení akcí prováděných po zjištění hrozby. Zahrnuje také činnosti po již zakončené odezvě na incident, jako je generování reportů a sdílení informací o hrozbách.

Ačkoli platformy SOAR a SIEM sdružují data z více zdrojů, jedná se o rozdílné techniky přístupu k bezpečnostnímu monitoringu. Systémy SIEM shromažďují data, identifikují anomálie, hodnotí hrozby a generují alerty. Systémy SOAR tyto úkoly také zvládají, ale mají další možnosti a zahrnují vyšší míru automatizace [25]. Systémy SOAR pracují s větším množstvím interních a externích aplikací, a to jak z oblasti bezpečnosti, tak i z oblasti aplikací, které z bezpečností přímo nesouvisí. Mnoho současných řešení SOAR využívá ke svému rozšíření systémů SIEM. V budoucnu se očekává, že dodavatelé SIEM přidají do svých služeb funkce SOAR [26].

2 Bezpečnostní monitoring s využitím umělé inteligence

Tato kapitola se zaměřuje na rozebrání základních technik umělé inteligence a strojového učení s následným rozborem možností zdokonalení funkčnosti systémů SIEM pomocí těchto technik. Umělá inteligence neboli AI (*Artificial Intelligence*) a strojové učení ML (*Machine Learning*) umožňují strojům učit se na základě zkušeností, přizpůsobovat se novým vstupům a řešit problémy podobným způsobem jako lidé. Může se jednat o cokoliv od počítačů hrajících šachy až po samořídící auta (obr. 2.1 znázorňuje jednotlivé oblasti umělé inteligence). Umělá inteligence se do značné míry opírá o hluboké učení (*Deep Learning*) a problém zpracování přirozeného jazyka – NLP (*Natural Language Processing*). Pomocí těchto technologií lze počítače, přesněji řečeno počítačové programy, vycvičit k plnění konkrétních úkolů tím, že zpracovávají velké množství dat a rozpoznávají v nich souvislosti.



Obr. 2.1: Dělení umělé inteligence [28]

2.1 Úvod do strojového učení

Strojové učení ML (*Machine Learning*) je metoda analýzy dat, která automatizuje vytváření složitých analytických modelů. Jedná se o odvětví umělé inteligence založené na myšlence, že systémy se mohou učit z dat konkrétní problematiky, identifikovat vzory (souvislosti) a rozhodovat s minimálním zásahem člověka. Strojové učení vzniklo na základě rozpoznávání vzorů a za předpokladu, že se počítače mohou učit, aniž by byly naprogramovány k provádění konkrétních úkolů. V rámci strojového učení je velice důležitý jeho iterativní aspekt, protože jak jsou modely vystavovány novým datům, měly by být schopny se samostatně přizpůsobovat [29].

V rámci strojového učení existují tři základní přístupy: učení s učitelem, učení bez učitele a zpětnovazebné učení[1]. Dále bude následovat rozbor těchto základních přístupů [29, 30, 31] (viz také obr. 2.2):

Obr. 2.2: Strojové učení a jeho dělení.

- **Učení s učitelem** (*Supervised learning*) – tyto algoritmy se trénují na základě označených příkladů, například na vstupu, kde je znám požadovaný výstup. Učící se algoritmus obdrží sadu vstupů spolu s odpovídajícími správnými výstupy a trénuje se porovnáváním skutečného vstupu s výstupy, aby našel chyby. Poté se model odpovídajícím způsobem upraví pro generování relevantnějších výstupů. Prostřednictvím metod, jako je **klasifikace** a **regrese**, používá učení s učitelem vzory k předpovídání hodnot značek na dalších neoznačených datech. Učení s dohledem se běžně používá v aplikacích, kde historická data předpovídají pravděpodobné budoucí události;
  - **Klasifikace** (*Classification*) – algoritmus k přesnému zařazení testovacích dat do určitých kategorií. Například rozeznání falešných poplachů od potenciálních hrozeb. Lineární klasifikátory, metoda podpůrných vektorů,

[1]Existuje také přístup, který je kombinací učení s učitelem a bez učitele – tzv. *semisupervised learning*.

rozhodovací stromy a náhodný les jsou běžnými typy klasifikačních algoritmů;
  - **Regrese** (*Regression*) – algoritmus k pochopení vztahu mezi závislými a nezávislými proměnnými. Regresní modely jsou užitečné pro předpovídání číselných hodnot na základě různých dat, jako jsou například data z již proběhlých kybernetických bezpečnostních incidentů k odhalení aktivit svědčících o potenciální infiltraci. Mezi typické regresní algoritmy patří lineární regrese, logistická regrese a polynomiální regrese.
- **Učení bez učitele** (*Unsupervised learning*) – používá se na datech, která nejsou nijak předem označená. Systému není sdělen správný výstup. Cílem je prozkoumat data a najít v nich nějakou strukturu. Učení bez učitele může například identifikovat segmenty logů s podobnými atributy v rámci probíhajícího incidentu. Mezi časté techniky užití učení bez učitele patří samoorganizující sítě (také *Kohonenova neuronová síť*), mapování nejbližších sousedů[2], a singulární rozklad;
  - **Shluková analýza** (*Clustering*) – technika dolování dat pro seskupování neoznačených dat na základě jejich podobností nebo rozdílů. Například algoritmy shlukování *K-means* přiřazují podobné datové body do skupin, kde hodnota K představuje velikost seskupení a granularitu. Tato technika je užitečná pro určení podobností u různých logů událostí (*korelace*), kategorizace logů událostí podle různých LES atd.;
  - **Redukce dimenzionality** (*Dimensionality reduction*) – technika učení, která se používá v případě, kdy je počet rysů (nebo dimenzí) v daném souboru dat příliš vysoký. Snižuje počet datových vstupů na zvládnutelnou velikost a zároveň zachovává integritu dat. Často se tato technika používá ve fázi předzpracování dat, například odstraňování šumu z vizuálních dat pro zlepšení kvality obrazu.
- **Zpětnovazebné učení** (*Reinforcement learning*) – často se používá v robotice, hrách a navigaci. Algoritmus pomocí pokusů a omylů zjišťuje, které akce jsou nejvhodnější pro jeho úspěšnost (toto je realizováno pomocí systému "odměn"a "trestů"). Tento typ učení má tři základní složky: agenta (učící se nebo rozhodující se subjekt), prostředí (vše, s čím agent pracuje) a akce (to, co agent může dělat). Cílem je, aby agent volil takové akce, které maximalizují očekávanou odměnu za daný čas. Agent dosáhne cíle mnohem rychleji, bude-li plnit očekávaný cíl programu. Známými algoritmy zpětnovazebného učení jsou např. **SARSA** (*state-action-reward-state-action*) a **Q-learning**.

[2]Tato technika se užívá k určení vztahů dvou rozdílných datových sad k vytyčení jejich podobností. Z angl. NNS (*Nearest Neighbour Search*).

2.2 Neuronové sítě a hluboké účení

Umělé neuronové sítě ANNs (*Artificial Neural Networks*) jsou prostředkem strojového učení, při kterém se počítač učí provádět určitý úkol na základě analýzy trénovacích příkladů. Příklady jsou obvykle předem ručně označeny. Například model pro rozpoznávání objektů by mohl dostat tisíce obrázků různých předmětů[3] spadajících do několika tříd (*Classes*). Názorným příkladem takového souboru dat může být datová sada *CIFAR-10* [32], která obsahuje 60 000 obrázků (z toho 50 000 je trénovacích a 10000 testovacích) deseti různých tříd o velikosti 32x32 pixelů – konkrétně se jedná o obrázky letadel, osobních automobilů, ptáků, koček, jelenů, psů, žab, koní, lodí a nákladních automobilů. Po dokončení fáze učení bude neuronová síť schopna s určitou pravděpodobností rozpoznat stejný typ předmětů ze tříd, na kterých se trénovala[4].

Pro dosažení nejvyšší efektivity strojového učení se datové sady zpravidla dělí na tři části [33]:

- **Trénovací datová sada** – tento typ dat slouží k vytvoření algoritmu strojového učení. Algoritmu se dodají potřebná data, která odpovídají očekávanému výstupu. Model opakovaně data vyhodnocuje, aby se dozvěděl více o jejich charakteristikách, a poté se upraví tak (proces učení), aby jednotlivé skupiny dat rozlišoval podle jejich identifikace[5];
- **Validační datová sada** – tato skupina dat během trénování vnáší do modelu nová data, která předtím nevyhodnocoval. Validační data poskytují první testování na neznámých datech a umožňují konkrétnímu modelu vyhodnotit, s jakou úspěšností řadí daná data do jednotlivých skupin. Validační data se ne vždy používají, ale mohou poskytnout užitečné informace pro optimalizaci modelu;
- **Testovací datová sada** – jedná se o další skupinu dat, se kterými v předchozích krocích model nepracoval, aby se dalo zjistit, že model dokáže vytvářet přesné předpovědi. V této fázi již nedochází k optimalizaci modelu, ale pouze k vyhodnocení jeho přesnosti. Na základě zjištěných informací lze vytvořit tzv. *matici záměn*, která určuje chybovost daného modelu.

V rámci neuronových sítí a hlubokého učení je důležité porozumět několika pojmům [34]:

- **Epocha** (*epoch*) – počet epoch určuje, kolikrát algoritmus zpracuje celou trénovací datovou sadu;
- **Vzorek** (*sample*) – jeden objekt v rámci datové sady;

[3] Soubor objektů, pomocí kterých se konkrétní neuronová síť učí, lze definovat jako *Dataset* (datová sada).

[4] Neuronová síť trénovaná pomocí datové sady *CIFAR-10* tedy u každého z 10 000 testovacích obrázků určí, do jaké z deseti tříd každý obrázek spadá s chybovostí zhruba 15 % [32].

[5] Zpravidla jsou skupiny dat rozděleny číselnými identifikátory, počínaje číslem 0.

- **Dávka** (*batch*) – počet vzorků, které mají být algoritmem zpracovány pro aktualizaci parametrů modelu;
- **Váhy** (*weight*) – proměnné parametry modelu, které řídí signál mezi dvěma neurony;
- **Rychlost učení** (*learning rate*) – parametr, který modelu předává měřítko o tom, jak markantně by se měly aktualizovat váhy modelu;
- **Ztrátová funkce** (*loss function*) – funkce, která slouží k výpočtu efektivity (ztrátovosti) modelu, tedy porovnává skutečné výsledky s těmi předpovídanými;
- **Optimalizátor** (*optimizer*) – funkce nebo algoritmus, který upravuje atributy neuronové sítě, jako jsou váhy (případně i rychlost učení). Pomáhá tak snížit celkovou ztrátovost modelu a zlepšit jeho přesnost. Nastavení správných vah pro model je náročný úkol, protože model hlubokého učení se obvykle skládá z milionů parametrů.

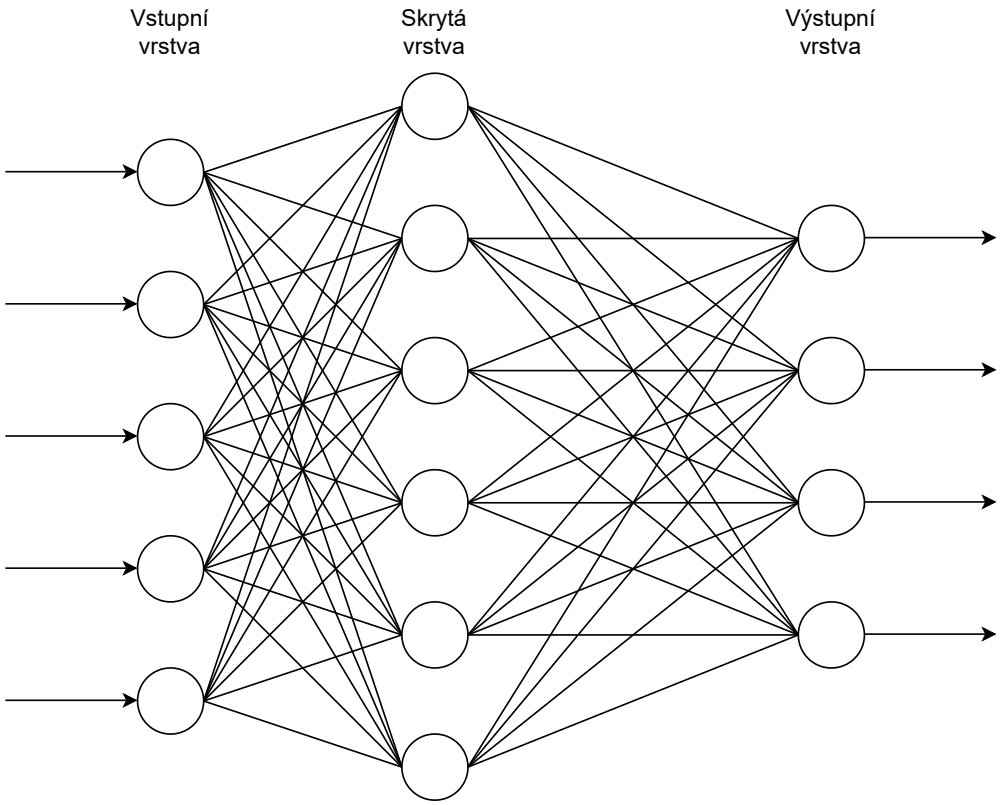
Neuronová síť je modelována podle lidského mozku a skládá se z tisíců nebo dokonce milionů jednoduchých výpočetních uzlů (neuronů), které jsou hustě propojeny. Většina neuronových sítí je uspořádána do vrstev neuronů a data jimi procházejí pouze jedním směrem. Jeden neuron může být propojen s několika neurony ve vrstvě pod ním, od kterých přijímá data, a s několika neurony ve vrstvě nad ním, kam jsou data posílána. Neuronová síť se skládá ze tří základních vrstev [35]:

- **Vstupní vrstva** – tato vrstva přijme data a předá je zbytku sítě;
- **Skrytá vrstva** – skrytá vrstva je v neuronové síti buď jedna, nebo je jich více. Právě skryté vrstvy jsou zodpovědné za výkon a složitost neuronových sítí. Vykonávají více funkcí najednou, například transformaci dat, automatické vytváření funkcí atd.;
- **Výstupní vrstva** – výstupní vrstva uchovává výsledek nebo výstup problému.

Příklad struktury neuronové sítě s jednou skrytou vrstvou (tzv. mělká neuronová síť) je vykreslen na obr. 2.3. Každému ze svých příchozích spojení neuron přiřadí hodnotu *weight* (váha). Neuron přijímá přes každé ze svých spojení jinou datovou položku (vstupní data) a násobí ji přiřazenou váhou. Výsledkem bude jediné číslo. Pokud je toto číslo nižší než *treshold* (prahová hodnota), neuron žádná data do další vrstvy nepředává. Pokud číslo překročí prahovou hodnotu, dojde k aktivaci neuronu, což bude mít za následek poslání čísla podél všech svých odchozích spojení. Při trénování neuronové sítě jsou všechny její váhy a prahové hodnoty zpočátku nastaveny na náhodné hodnoty. Během trénování se váhy a prahové hodnoty průběžně upravují, dokud trénovací data se stejným označením konzistentně nedávají podobné výstupy [36].

Deep learning (hluboké učení[6]) je specifickou podskupinou strojového učení, od

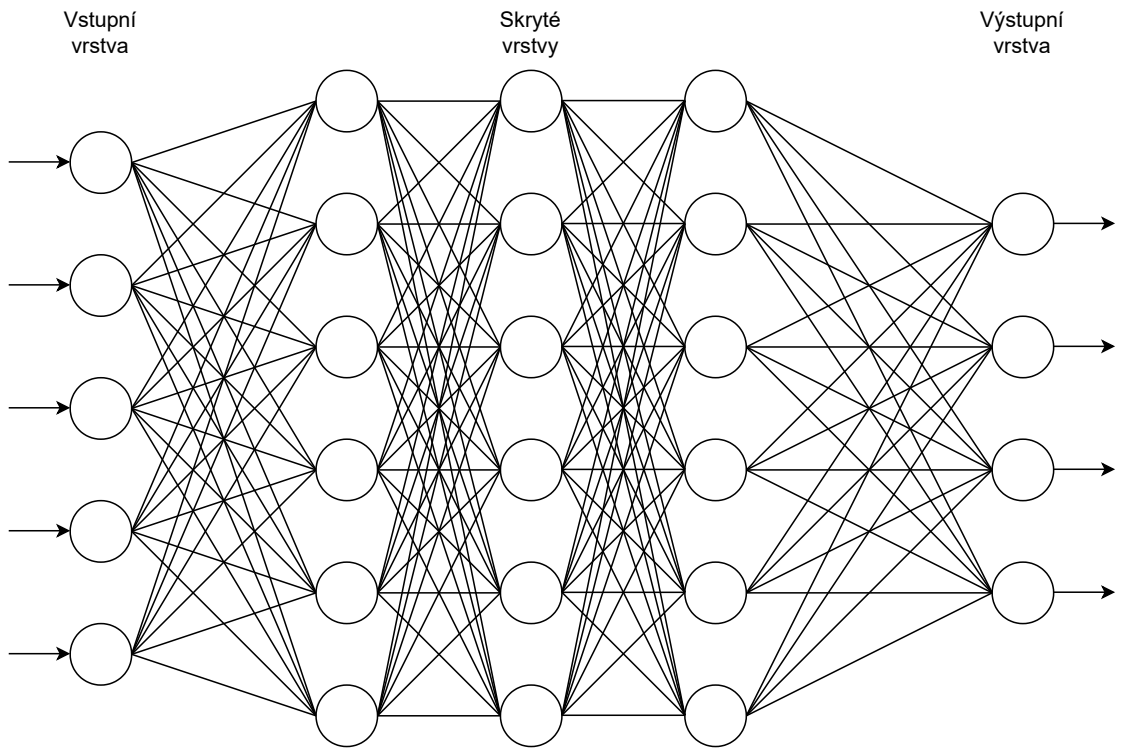
[6]Jedná se o nový přístup k učení se reprezentací z dat, který klade důraz na učení se následujících



Obr. 2.3: Struktura mělké neuronové sítě

kterého se liší typem zpracovávaných dat a metodami učení [37]. Algoritmy strojového učení využívají strukturovaná a označená data k tvoření předpovědí – to znamená, že ze vstupních dat jsou pro model definovány konkrétní rysy, které jsou uspořádány do tabulek. To nutně neznamená, že nestrukturovaná data nemohou být také použita; znamená to jen, že pokud jsou nestrukturovaná data použita, zpravidla prochází určitým procesem, aby byla uspořádána do strukturovaného formátu[38]. Hluboké učení eliminuje některé procesy předzpracování dat. Tyto algoritmy mohou přijímat a zpracovávat nestrukturovaná data, jako je text a obrazy, a automatizuje extrakci příznaků, čímž odstraňuje část závislosti na nutnosti lidského zásahu. Běžně se tedy užívá k analýze a interpretaci velkého množství dat. Hluboké učení si lze představit jako způsob automatizace prediktivní analýzy. Zatímco tradiční algoritmy strojového učení jsou lineární, algoritmy hlubokého učení jsou poskládány v hierarchii rostoucí složitosti a abstrakce [39]. V rámci hlubokého učení lze brát v potaz neuronové sítě s vícero skrytými vrstvami (tzv. hluboké neuronové sítě) – viz obr. 2.4.

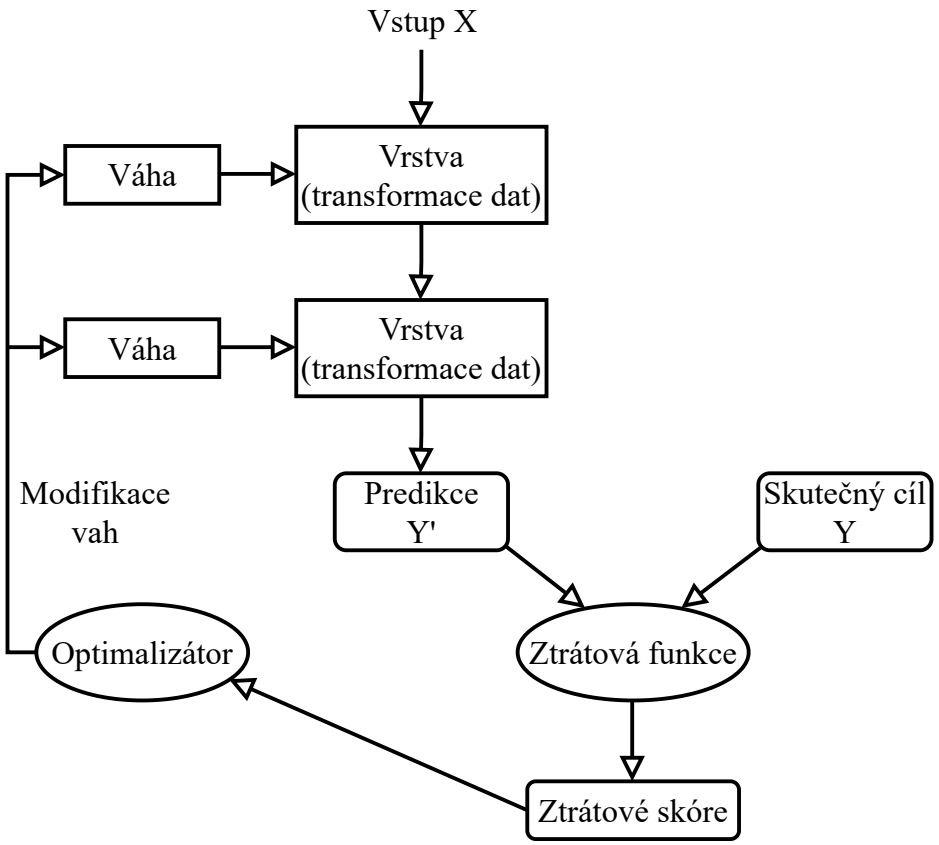
vrstev stále smysluplnějších reprezentací. Hloubka v tomto případě referuje k desítkám až stovkám vrstev reprezentací, zatímco strojové učení se zaměřuje na učení pouze jedné nebo dvou vrstev reprezentací dat [40] – tzv. *shallow learning* (mělké učení)



Obr. 2.4: Struktura hluboké neuronové sítě

Počítačové programy využívající hluboké učení procházejí procesem učení, kde jsou mapovány vstupy na cíle pomocí hluboké posloupnosti jednoduchých transformací dat (vrstev) a tyto transformace dat jsou naučeny na konkrétní množině vyřešených příkladů. Specifikace toho, co vrstva provádí se vstupními daty, je uložena ve vahách vrstvy, které slouží k parametrizaci transformace vrstvy. V tomto kontextu má učení význam zjišťování množiny hodnot pro váhy všech vrstev v konkrétní síti pro mapování příkladů vstupů na přidružené cíle [40]. Z důvodu značného množství využitých dat (a také faktu, že změna hodnoty jednoho parametru ovnivní chování všech ostatních) je důležité měřit míru toho, na kolik se výstup liší od očekávání, což má za úkol tzv. *loss function* (ztrátová funkce[7]). Ztrátová funkce porovná výstup sítě a skutečný cíl a vypočítá *loss score* (ztrátové skóre). Velice důležitým rysem hlubokého učení je použití ztrátového skóre ve smyslu zpětné vazby, pomocí čehož se posléze pomocí tzv. *optimalizátoru* upraví hodnoty vah ke zdokonalení výstupu. V počátečním stavu jsou váhy nastavené náhodně, takže dochází k náhodným transformacím a ztrátové skóre proto bude nejvyšší. V rámci mnohokrát opakujícího se trénovacího cyklu (*training loop*) jsou váhy upravovány a ztrátové skóre se snižuje. Tento proces trénování lze sledovat na obr. 2.5. Síť s minimální ztrátou se označuje jako *trained network* (natrénovaná síť)[40].

[7]Také nazývaná jako *objective function* (cílová funkce).

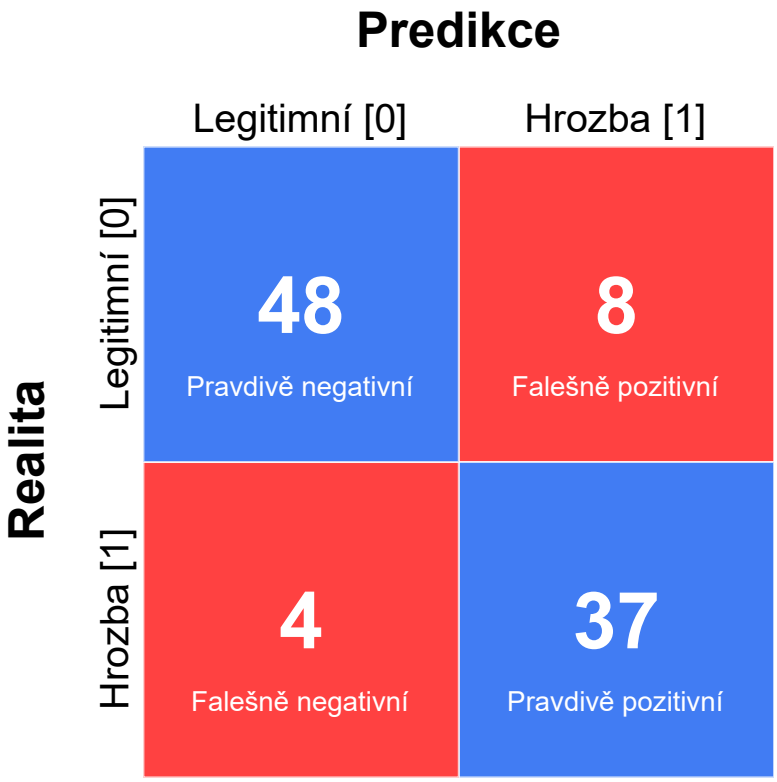


Obr. 2.5: Proces trénování neuronových sítí

V rámci trénování je také důležité vyhodnocení modelu pro kontrolu toho, na kolik model správně vykonává svoji funkci podle čtyř *klasifikačních metrik* [41]. Podle těchto základních klasifikačních metrik lze sestrojit tzv. *matici záměn*, která je užitečná k vizualizaci úspěšnosti konkrétního modelu (pro příklad viz obr. 2.6). Tyto klasifikační metriky jsou pro vyhodnocení modelu velice důležité a dají se podle nich vypočítat další metriky (viz dále), které poskytují perspektivu úspěšnosti modelu. Následuje popis čtyř základních klasifikačních metrik z pohledu kybernetické bezpečnosti [41, 42]:

- **Správný výsledek**:
  - Pravdivě pozitivní **TP** (*True Positive*) – vyhodnocení události jako škodlivé, když je opravdu škodlivá a jedná se tedy o hrozbu;
  - Pravdivě negativní **TN** (*True Negative*) – vyhodnocení události jako neškodné, když je opravdu neškodná a jedná se tedy o výsledek legitimního provozu.
- **Špatný výsledek**
  - Falešně pozitivní **FP** (*False Positive*) – vyhodnocení události jako škod-

livé, ale ve skutečnosti se jedná o legitimní provoz;
– Falešně negativní **FN** (*False Negative*) – vyhodnocení události jako neškodné, ale ve skutečnosti se jedná o hrozbu.



Obr. 2.6: Příklad matice záměn

Pro podrobnější analýzu výkonnosti modelu existují další, často užívané metriky, které detailněji popisují, jak je model účinný [28, 41, 43]:

- **Přesnost** (*Accuracy*) – podíl správných předpovědí pro testovací data. Lze ji vypočítat vydělením počtu správných předpovědí počtem všech předpovědí;

$$přesnost = \frac{správné\ předpovědi}{všechny\ předpovědi} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Preciznost** (*Precision*) – podíl správně odhalených hrozeb vzhledem k legitimnímu provozu nesprávně označenému jako hrozba, tedy kolik označených hrozeb je relevantních;

$$preciznost = \frac{TP}{TP + FP}$$

- **Senzitivita** (*Recall*) – podíl správně odhalených hrozeb vzhledem hrozbám nesprávně vyhodnoceným jako legitimní provoz, tedy kolik relevantních hrozeb je označených;

$$senzitivita = \frac{TP}{TP + FN}$$

- **Specificita** (*Specificity*) – podíl provozu správně označeného jako legitimní vzhledem k legitimnímu provozu nesprávně označenému jako hrozba, tedy jaké množství legitimního provozu je správně označeno;

$$specificita = \frac{TN}{TN + FP}$$

- **F-míra** (*F-score*) – metrika užitečná k vypočítání úspěšnosti modelu, pokud se hodnoty preciznosti a senzitivity výrazně liší, jedná se o jejich harmonický průměr.

$$F\text{-}míra = 2 * \frac{preciznost * senzitivita}{preciznost + senzitivita}$$

2.2.1 RNN (*Recurrent Neural Networks*)

Rekurentní neuronové sítě řeší problém bezstavových neuronových sítí. Tento problém spočívá v tom, že chybí persistence informace –> paměť sítě se při každé nové iteraci resetuje[8]. RNN proto využívají smyček, kdy je informace z předchozího stavu předána i do dalšího stavu.

LSTM (*Long Short-Term Memory*)

Speciální druh architektury RNN, který se dokáže učit dlouhodobé závislosti (*long-term dependencies*). Všechny RNN mají strukturu v podobě řetězce opakujících se modulů neuronové sítě, ale ve standardních RNN má každý modul velmi jednoduchou strukturu. LSTM mají stejnou strukturu s tím rozdílem, že opakující se modul má složitější strukturu (např. 4 vzájemně na sebe působící vrstvy neuronové sítě namísto pouze jedné vrstvy) [44]. Základním konceptem LSTM je stav buňky (*cell*) a její různé brány (*gate*). Stav buňky přenáší relativní informace po celém řetězci sekvencí (jedná se tedy o paměť sítě). Stav buňky může teoreticky přenášet příslušné informace po celou dobu zpracování sekvence. Tudíž i informace z dřívějších časových kroků se mohou dostat do pozdějších časových kroků, což snižuje účinky krátkodobé paměti. Jak stav buňky pokračuje ve své cestě, informace se do stavu buňky přidávají nebo odebírají prostřednictvím bran. Brány rozhodují o tom, jaké informace mohou být ve stavu buňky obsaženy a zároveň se mohou během trénování naučit důležitost informací – tato operace probíhá pomocí tzv. aktivace *sigmoid*, která spočívá v určení hodnoty mezi 0 a 1, což určuje, které informace budou zapomenuty (hodnota 0) a které zachovány (hodnota 1). K regulaci sítě je také užito funkce *tanh*, která určí hodnoty mezi -1 a 1. V rámci LSTM figurují 3 druhy bran [45]:

[8]Bezstavové neuronové sítě se používají pro případy, kdy není důležitý předchozí stav sítě – např. při analýze zvuku v reálném čase. Tato metoda má výhodu menší složitosti sítě a tím pádem vyšší výpočetní rychlost.

- **Forget gate** zajišťuje pomocí funkce sigmoid důležitost informací – hodnota funkce sigmoid bližší k 1 znamená ponechání informace. Hodnota blížící se k 0 značí zapomenutí informace;
- **Input gate** slouží k aktualizaci stavu buňky. Nejprve je předán předchozí skrytý stav (*hidden state*) a aktuální vstup do funkce sigmoid. Skrytý stav a aktuální vstup je také předán do funkce tanh. Nakonec je vynásoben výstup tanh s výstupem sigmoid, kde je rozhodnuto, které informace z výstupu tanh budou zachovány;
- **Output gate** rozhoduje o příštím skrytém stavu obsahujícím informace z předchozího vstupu. Nejprve je předán předchozí skrytý stav a aktuální vstup do funkce sigmoid, poté je předán nově upravený stav buňky funkci tanh, výstup tanh vynásoben výstupem funkce sigmoid a nakonec je rozhodnuto, jakou informaci má skrytý stav nést. Výstupem je tedy skrytý stav. Nový stav buňky i nový skrytý stav se potom přenášejí do dalšího časového kroku.

ULMFiT (*Universal Language Model Fine-tuning*)

ULMFiT je architektura, založená na RNN, a metoda přeneseného učení, kterou lze aplikovat na všechny úlohy NLP a dosáhnout žádaných výsledků. Zahrnuje třívrstvou architekturu AWD-LSTM (*ASGD Weight-Dropped Long Short-Term Memory*) a proces trénování se skládá ze tří kroků [52]:

1. **LM pre-training** – předtrénování obecného jazykového modelu pro zachycení obecných rysů jazyka na datové sadě *Wikitext-103*, která obsahuje přes 28000 článků z Wikipedie a 103 milionů slov;
2. **LM fine-tuning** – doladění jazykového modelu na datech cílové úlohy;
3. **Classifier fine-tuning** – doladění klasifikátoru na cílové úloze.

Protože různé vrstvy tohoto modelu zachycují různé typy informací, jsou tyto vrstvy v různé míře dolaďovány pomocí 'Discr' (*Discriminative fine-tuning*), což je strategie ladění spočívající v různých rychlostech učení pro každou vrstvu. Trénování se provádí pomocí STLR (*Slantered Triangular Learning Rates*). STLR spočívá v plánování rychlosti učení, kde je nejprve rychlost učení lineárně navyšována a poté lineárně snižována.

Doladění cílového klasifikátoru je v ULMFiT dosaženo pomocí postupného rozmrazování (*gradual unfreezing*). Namísto dolaďování všech vrstev najednou, u kterého hrozí zapomínání, ULMFiT postupně rozmrazuje model počínaje poslední vrstvou (tj. nejblíže výstupu), protože ta obsahuje nejvíce specifických znalostí. Nejprve se rozmrazí poslední vrstva a všechny rozmrazené vrstvy se dolaďují po dobu jedné epochy. Poté se rozmrazí a doladí další skupina zmrazených vrstev a to se opakuje, dokud nejsou všechny vrstvy doladěny [52].

2.2.2 CNN (*Convolutional Neural Networks*)

Konvoluční neuronové sítě slouží k rozeznání jednoho obrázku od druhého pomocí přidělování důležitosti určitým aspektům v rámci tohoto obrázku. CNN dokáže zachytit prostorové a časové závislosti pomocí příslušných filtrů. Úkolem konvoluční sítě je redukovat obrázky do podoby, která je snadněji zpracovatelná, aniž by se ztratily vlastnosti rozhodující pro získání dobré míry úspěšné předpovědi. Zatímco v primitivních metodách se filtry vytvářejí ručně, CNN mají schopnost se naučit aplikovat tyto filtry samostatně [46].

Pro rozeznávání obrázků je klasická neuronová síť neefektivní[9]. Konvoluční neuronové sítě využívají toho, že vstup tvoří obrázky, a mohou tedy svoji architekturu efektivně omezit – konkrétně na rozdíl od běžné neuronové sítě mají vrstvy konvoluční sítě neurony uspořádané ve 3 rozměrech: **šířka, výška, hloubka** (kde výška a šířka je počet pixelů a hloubka značí barevné spektrum).

2.2.3 TNN (*Transformer Neural Networks*)

Architektura Transformer je v poslední době jádrem téměř všech významných vývojových trendů v oblasti NLP. V roce 2017, kdy byla tato architektura představena společností Google, se pro jazykové úlohy, jako je strojový překlad a systémy pro zodpovídání otázek (*question answering*), používaly rekurentní neuronové sítě. Tato architektura překonala jak RNN, tak i CNN a navíc se snížily také výpočetní prostředky potřebné k trénování modelů [47]. Velice důležitou funkcí architektury Transformer je zlepšování porozumění přirozenému jazyku stroji pomocí tzv. *self-attention* mechanismu, který přímo modeluje vzahy mezi všemi slovy ve větě bez ohledu na jejich pozici. Této funkce lze samozřejmě využít i v oblasti kybernetické bezpečnosti (např. trénování architektury Transformer pro predikci škodlivých URL [48]). Architektura Transformer má ovšem i nedostatky. Dokáže pracovat pouze s textovými řetězci fixní délky, které musí být před vstupem do systému rozděleny na určitý počet segmentů. Toto rozdělení textu způsobuje fragmentaci kontextu (např. pokud je věta rozdělena v její polovině, je ztracen kontext), protože není respektována věta nebo jiná sémantická hranice [49]. Fragmentace kontextu tedy omezuje dlouhodobé učení se závislostem. Ve fázi vyhodnocování je segment posunut doprava pouze o jednu pozici. Nový segment musí být zpracován zcela od začátku. Tento způsob vyhodnocování je navíc poměrně náročný na výpočetní výkon.

Tato fragmentace kontextu je lépe řešena v architektuře **Transformer-XL**, kde se během trénovací fáze skrytý stav, vypočítaný pro předchozí stav, používá jako dodatečný kontext pro aktuální segment. Tento mechanismus rekurence odstraňuje

[9]Pokud by měla klasická neuronová síť zpracovávat obrázek o rozměrech 200x200 pixelů, musel by plně propojený neuron v první skryté vrstvě mít 120000 vah.

omezení plynoucí z použití kontextu s pevnou délkou. Během vyhodnocovací fáze lze reprezentace z předchozích segmentů použít opakovaně, místo aby se počítaly od začátku (jako je tomu v případě původního modelu Transformer) [49]. To samozřejmě mnohonásobně zvyšuje rychlost výpočtu; konkrétně až 1800 krát [47].

Google BERT

Jako příklad modelu využívajícího architekturu Transformer lze uvést **Google BERT** (*Bidirectional Encoder Representations*), který zohledňuje kontext z obou stran slova. Všechny předchozí modely braly v potaz vždy jen jednu stranu slova – buď levou, nebo pravou. Tato obousměrnost pomáhá modelu mnohem lépe pochopit, ve kterém kontextu bylo slovo použito. Kromě toho je BERT navržen pro víceúlohové učení a tím pádem může provádět různé úlohy NLP současně. V době svého vydání dosahoval BERT nejlepších výsledků v 11 úlohách NLP. S jeho pomocí lze natrénovat vlastní model bez jakékoliv přípravy (pro úlohy klasifikace, rozpoznávání entit, odpovědi na otázky atd.) už během pár hodin [47].

2.3 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka neboli NLP (*Natural Language Processing*), je odvětví umělé inteligence, které pomáhá počítačům porozumět lidskému jazyku, interpretovat ho a manipulovat s ním. Jedná se o velice rychle se rozvíjející technologii. Hlavní motivací pro rozvoj NLP je problém porozumění počítačů lidmi – zejména kvůli složitosti strojového kódu nebo strojového jazyka, které jsou pro většinu lidí do značné míry nesrozumitelnými. Zpracování přirozeného jazyka zahrnuje mnoho různých technik interpretace lidského jazyka, od statistických metod a metod strojového učení až po metody založené na pravidlech, včetně algoritmických přístupů [50]. Jedná se o velice složitou a variabilní oblast, protože textová a hlasová data se velmi liší, stejně jako jejich praktické využití. K základním úlohám NLP patří [50]:

- **Kategorizace obsahu** – lingvistický přehled včetně vyhledávání, indexování a detekce duplicit;
- **Rozpoznání významu textu** – zachycení významu a tématu;
- **Získání kontextu (parsování)** – výtah/shrnutí důležitých informací z rozsáhlých textů;
- **Analýza emocí** – identifikace nálady nebo subjektivních názorů tvůrce textu;
- **Komunikace s lidmi** – převod textu na řeč a naopak;
- **Strojový překlad** – automatický překlad řeči nebo textu z jednoho jazyka do druhého.

Zpracování přirozeného jazyka se zabývá především analýzou textu. Tato analýza spočívá v počítání, seskupení a kategorizaci slov, aby z velkých objemů textu byly získány struktura a význam. Díky tomu lze získané proměnné z původního textu vizualizovat, filtrovat, nebo použít jako vstupy do prediktivních modelů či jiných statistických metod [50].

NLP má podoblast zvanou porozumění přirozenému jazyku NLU (*Natural Language Understanding*), která začíná být stále populárnější díky svému potenciálu v kognitivních aplikacích a aplikacích umělé inteligence. Navíc se orientuje nad rámec strukturálního porozumění jazyku. Pomocí NLU lze interpretovat záměr, řešit kontext a nejednoznačnost slov. NLU disponuje i schopností generovat dobře formulovaný lidský jazyk. Algoritmy NLU řeší velice složitý problém sémantické interpretace – tedy pochopení zamýšleného významu mluveného nebo psaného jazyka se všemi detaily, které jsou poměrně jednoduché pro většinu lidí, ale složité pro stroje [50]. Je to právě vývoj NLU, které by mohlo přispět ke spolehlivé automatizaci všech procesů v rámci bezpečnostního monitoringu a zároveň při řešení kybernetických bezpečnostních incidentů.

K rychlému nárůstu rozšíření NLP došlo především díky konceptu přeneseného učení (*transfer learning*), které je umožněno prostřednictvím předem naučených modelů. Přenesené učení je v podstatě schopnost trénovat model na jedné datové sadě a poté tento model přizpůsobit k provádění různých funkcí NLP na jiné datové sadě [47]. Tento fakt usnadňuje práci a čas, protože není nutné vytvářet nový model od začátku. Největší výzvou v oblasti hlubokého učení jsou vysoké požadavky (z hlediska objemu) na data pro trénování modelů. Je obtížné najít datové sady tak rozsáhlých rozměrů a jejich příprava je příliš nákladná (také kvůli velkému výpočetnímu výkonu, který je nutný), a tím pádem pro většinu organizací není možné je vytvořit [51]. Lze ovšem použít předtrénované *state-of-the-art*[10] modely hlubokého učení a upravit je tak, aby vyhovovaly konkrétním požadavkům. Tato metoda není tak náročná na zdroje jako trénování modelu hlubokého učení od nuly a přináší dobré výsledky i na malém množství trénovacích dat. Dále budou uvedeny příklady architektur, které využívají přenesené učení.

2.4 Vylepšení bezpečnostního monitoringu pomocí AI

Tato kapitola se věnuje možným vylepšením procesu bezpečnostního monitoringu, která již byla zpracována v rámci výzkumu a jsou stručně popsána níže. Hlavním

[10]Tento výraz označuje nejvyšší úroveň obecného vývoje, např. zařízení, techniky nebo vědního oboru, které bylo dosaženo v určitém čase.

aspektem těchto vylepšení je zefektivnění práce bezpečnostních operátorů SOC, přičemž je kladen důraz na automatizaci některých procesů a nahrazení ruční práce s cílem zvýšení efektivity bezpečnostního týmu.

Jak již bylo popsáno v kapitole 1.4, řešení SIEM poskytuje organizaci možnost monitorování hrozeb, korelaci událostí, hlášení incidentů a reakci na incidenty pomocí různých zdrojů logů událostí a následně upozorňuje bezpečnostní tým na potenciálně škodlivé aktivity, které potřebují pozornost, anebo neprodlený zásah. Kvůli nedostatkům systémů SIEM a nutnosti jejich správy (např. psaní korelačních pravidel) je pro jejich efektivitu kritické zlepšení, které by operátorům usnadnilo detekci kybernetických bezpečnostních incidentů a reakci na ně.

Využití umělé inteligence má v kybernetické bezpečnosti značný význam. Dále bude následovat výpis možných zdokonalení pro stávající metody bezpečnostního monitoringu pomocí umělé inteligence s ohledem na možné využití technik strojového učení [53, 54]:

- **Shromažďování**, **zpracování** a **analýza** velkého množství dat, aniž by byl zpomalen reakční potenciál systému (*redukce dimenzionality*);
- **Optimalizace modulu UEBA** (*User and Entity Behavioral Analytics*), který odhaluje nepravidelné vzorce chování uživatelů. Tyto vzorce zahrnují časové změny v pravidelném rozvrhu přístupu uživatelů do systému (anomálie) nebo připojení z různých geografických lokací a je díky nim možné lépe rozlišovat mezi standardním a podezřelým chováním uživatelů (*shluková analýza, regrese*);
- **Snížení počtu falešných poplachů**, což by bezpečnostnímu týmu umožnilo soustředit pozornost na události s vyšší prioritou (*klasifikace*);
- **Funkce předpovídat průběh** budoucích incidentů na základě těch předešlých (*regrese*);
- **Automatické generování** grafů či statistik (*shluková analýza*);
- **Určení fáze** probíhajícího útoku (*regrese*);
- **Zodpovězení důležitých otázek** ohledně stávajících, či minulých incidentů (*question answering*);
- **Parsování logů** a jejich **korelace** bez nutnosti ručního psaní pravidel (*shluková analýza, klasifikace*);
- **Analýza toku logů** z jednoho zdroje a navrhování dalších užitečných informací ke sledování – nové use cases (*klasifikace, regrese*);
- **Automatická rozhodnutí** v reálném čase vedoucí k mitigaci hrozby (*zpětnovazebné učení*).

Rešerše existujících vylepšení

V rámci analýzy možných vylepšení bezpečnostního monitoringu pomocí umělé inteligence byla provedena rešerše existujících řešení, která se soustředí na zvýšení efektivity současných systémů. Hlavním přínosem těchto vylepšení je automatizace opakujících se procesů, které bezpečnostní týmy musejí provádět ručně. Tato řešení zahrnují detekci potenciálně škodlivých událostí (anomálií) – **DeepLog**, hledání korelací v rámci sekvence událostí a následně určení potenciálně škodlivé události v rozsahu této sekvence – **DeepCASE**, a v poslední řadě určení pravděpodobnosti možné následující události po určité sekvenci událostí – **Tiresias**.

DeepLog

DeepLog, model hluboké neuronové sítě využívající LSTM, který zpracovává logy událostí jako sekvenci přirozeného jazyka. To modelu DeepLog umožňuje automaticky se učit vzory logů z jejich běžného zpracovávání a detekovat anomálie, pokud se vlastnosti logů odchylují od dat logů při běžném provozu. DeepLog se také v průběhu času přizpůsobuje novým vzorům logů [55].

Model DeepLog využívá ke svému trénování a testování datovou sadu složenou z HDFS (*Hadoop Distributed File System*) logů. Tato datová sada (viz tab. 2.1) je vytvořena pomocí zpracování původních dat z logů a je rozdělena na tři části (kde pouze 1 % z celkové datové sady slouží k trénování modelu) [56]:

- **hdfs_train**;
- **hdfs_test_normal**;
- **hdfs_test_abnormal**.

Pro finální práci s modelem jsou ovšem ponechány pouze dvě datové sady **hdfs_train** a **hdfs_test** (tato byla složena z předchozích dvou částí obsahujících normální a abnormální logy, přičemž data v ní byla promíchána).

Tab. 2.1: Datová sada HDFS logů použitá pro model DeepLog

Povaha logů	Počet logů k trénování	Počet logů k testování
Normální	4 855	553 366
Abnormální	1 638	15 200

Pro efektivní zpracování logů je nutné jejich předzpracování, aby se model učil ze strukturovaných dat. V tomto konkrétním případě je předzpracování dat provedeno pomocí extrakce tzv. *log key*, tedy příslušného klíče (unikátní ID) každého logu, přičemž je vytvořena finální datová sada obsahující sekvence čísel, které byly vygenerovány z původních logů ve výsledném poměru 1:1. Zpracování dat probíhá

v reálném čase (tzv. *streaming mode*) a je provedeno pomocí parseru **Spell** (*Streaming Parser for Event Logs using an LCS; longest common subsequence*), který je navržen pro zpracování velkého množství dat z logů sbíraných v reálném čase [57].

Model využívá knihovnu PyTorch a další (více viz [58]) a jeho trénování funguje na bázi učení s učitelem. I přes užití velice malé části z celkové datové sady k trénování modelu, jsou v popsaném řešení [55] prezentovány *state-of-the-art* výsledky – téměř 100 % úspěšnost detekce abnormálních logů z testovací datové sady, která tvoří více než 99 % z celé datové sady.

DeepCASE

Stejně jako v případě DeepLog se jedná o řešení využívající hluboké učení. Tento návrh slouží k analýze kontextu bezpečnostních událostí, aby bylo jednodušší určit, jaké události vyžadují podrobnější zkoumání. Díky tomu je snížen počet událostí, které je třeba zkontrolovat. Navíc je v řešení popsáno, jak může kontext poskytnout informace o tom, proč jsou některé události klasifikovány jako škodlivé. V oficiálním článku (viz literatura [59]) je popsáno, že tento přístup automaticky filtruje 86,72 % událostí a tím pomáhá snížit manuální zátěž bezpečnostních operátorů SOC až o 90,53 %, přičemž předpokládá riziko potenciálních hrozeb v méně než 0,001 % případů.

Tento přístup intuitivně hledá korelace v sekvencích událostí generovaných pro konkrétní zařízení. Přesněji řečeno, hledají se korelace mezi událostmi v kontextu události e_i a samotnou událostí e_i, podle kterých se posléze určuje, zda e_i vznikla škodlivou aktivitou. Po detekování škodlivé aktivity určité e_i jsou podobné sekvence událostí shlukovány a předloženy bezpečnostnímu operátorovi, který určí, zda tato kombinace událostí představuje hrozbu, či ne. DeepCASE se pak toto rozhodnutí naučí a automaticky ho použije na podobné sekvence událostí nalezené v budoucnu (podobně jako systém SOAR, popsaný v podkapitole 1.6). Tento přístup automaticky zpracovává známé korelace, takže se bezpečnostní operátoři mohou zaměřit na nové hrozby [59].

Hlavní předností tohoto modelu je užití datové sady **Lastline**. Tato datová sada je velice obsáhlá, protože zahrnuje logy událostí sesbírané v rozmezí pěti měsíců z **20 různých organizací**, kde vcelku bylo pomocí **395 monitorovacích nástrojů** sledováno téměř **388 tisíc zařízení**. Tohle sledování mělo za výsledek **∼10,5 milionů bezpečnostních logů**, přičemž tyto obsahovaly **291 různých typů** bezpečnostních událostí (detailní popis datové sady lze vidět v tab. 2.2).

Na datovou sadu Lastline se vztahuje dohoda o mlčenlivosti, tudíž nejsou dostupné její podrobnější detaily [60]. Na modelu DeepCASE byly provedeny testy s oběma datovými sadami (HDFS i Lastline), kde 20 % z nich bylo určeno k tréno-

Tab. 2.2: Detail datové sady Lastline, kategorizovaný podle úrovně rizika

Úroveň rizika	INFO	LOW	MEDIUM	HIGH	ATTACK
Počet událostí	7 741 084	2 383 306	184 907	46 401	45 089

vání modelu a 80 % k jeho testování, a výsledky byly porovnány s měřeními ostatních řešení – DeepLog a Tiresias; viz tab. 2.3.

Tab. 2.3: Porovnání DeepCASE s modely DeepLog a Tiresias

	Model	Preciznost	Senzitivita	F-míra	Přesnost	Čas trénování
HDFS	DeepLog	0,897	0,893	0,894	0,893	1,0 s
	Tiresias	0,897	0,876	0,880	0,876	15,0 s
	DeepCASE	0,904	0,906	0,904	0,906	1,3 s
Lastline	DeepLog	0,897	0,904	0,898	0,904	6,8 s
	Tiresias	0,955	0,962	0,957	0,962	291,5 s
	DeepCASE	0,979	0,981	0,979	0,981	13,8 s

Stejně jako v případě DeepLog, DeepCASE také využívá framework PyTorch, jehož kód je dostupný z [60] a instrukce k instalaci jsou dostupné z [61]. Dle výsledků, prezentovaných v oficiálním dokumentu, jde vidět, že DeepCASE je užitečným nástrojem pro uvolnění manuální zátěže operátorů SOC, která je v dnešní době jedním z nejvýraznějších problémů v oblasti kybernetické bezpečnosti.

Tiresias

Systém Tiresias využívá rekurentní neuronové sítě (RNN) k předpovídání budoucích událostí na základě předchozích pozorování. Tento přístup je účinný při předpovídání příští události, která může nastat s **přesností až 0,93**, což bylo prezentováno v oficiálním dokumentu [62]. V dokumentu je také prokázáno, že modely naučené pomocí systému Tiresias jsou s postupem času poměrně stabilní a poskytují mechanismus, který dokáže identifikovat náhlé poklesy přesnosti a v reakci na to spustit přetrénování systému. Je také poukázáno na klíčový aspekt RNN v rámci predikce budoucích událostí, což je dlouhodobá paměť, a z toho je vydedukováno, že jednodušší metody na tento úkol nestačí.

Tento model byl testován na datové sadě **3,4 miliard bezpečnostních událostí** shromážděných z komerčních IPS, pomocí kterých bylo monitorováno 740 tisíc zařízení v rozmezí 27 dní. Datová sada, ani její část není v rámci řešení poskytnuta. Nevýhodou systému Tiresias je nutnost předem označit škodlivá data, aby bylo

možné vytvořit na základě těchto označení signatury. Pokud tedy dojde k útoku pomocí nové *zero-day* zranitelnosti, není možné takový útok odhalit, dokud na takové zranitelnosti nejsou vytvořeny nové signatury. Tuhle nevýhodu lze ovšem do jisté míry obejít a řešení automatizovat s pomocí databází hrozeb.

Ke správné funkci tohoto modelu jsou popsány základní čtyři kroky [62]:

1. **Sběr dat a jejich zpracování** (*Data collection and preprocessing*) – data jsou sbírána z koncových zařízení, potom uspořádaná jako sekvence bezpečnostních událostí a přijímána na vstup. Tyto sekvence dat, rozdělené podle zdrojového zařízení, jsou následně seřazeny podle časových značek. Cílem tohoto kroku je vytvoření trénovací (D_T) a validační (D_V) datové sady pro sestavení predikčního modelu Tiresias pro další fázi, přičemž platí, že $D_T \cap D_V = \emptyset$;
2. **Trénování a validace modelu** (*Model training and validation*) – pro tuto fázi byl zvolen konkrétní model rekurentních neuronových sítí LSTM. V prvním kroku, tedy pro trénování se model učí předpovídat událost e_{w+1} pro každou sekvenci událostí z trénovací datové sady $\{e_1, ..., e_w\}$. Cílem tohoto kroku je maximalizace pravděpodobnosti predikce správné události e_{w+1}. V kroku validace jsou tyhle predikce přezkoumány a případně upraveny váhy modelu. Je důležité podotknout, že (D_T) a (D_V) pocházejí z různých zařízení, což pro model představuje náročnější práci, ale nejspíše i jeho lepší výsledky při aplikaci v neznámém prostředí;
3. **Předpověď bezpečnostních událostí** (*Security event prediction*) – jakmile je model natrénován, jsou na vstup poslány sekvence událostí $\{e_0, ..., e_i\}$ z reálného prostředí a jsou určeny možné následující události e_{i+1}, každá s modelem přidělenou pravděpodobností, přičemž je vybrána ta nejpravděpodobnější. Následně je ověřeno, zda byla tato předpověď pravdivá a pokud ne, jsou v souladu s tím upraveny parametry modelu;
4. **Monitorování spolehlivosti predikce** (*Prediction performance monitoring*) – cílem této poslední fáze je zajistit co nejvyšší přesnost modelu v rámci jeho vyhodnocování (tzn. hodnoty preciznosti, senzitivity a F-míry) v delším časovém období. Je stanovena jistá mezní hodnota, která jakmile je překročena a model již nevykazuje optimální přesnost, je automaticky vyhodnocena nutnost jeho přeučení.

Tiresias využívá frameworku TensorFlow, což umožňuje jeho využití nejen na klasická koncová zařízení (tzn. stolní počítač atd.), ale i např. na mobilní zařízení. V testovacím scénáři pro vyhodnocení modelu Tiresias je provedeno trénování modelu v rozmezí jednoho týdne (1. - 7. 11.). Tímto trénováním bylo vytvořeno 6 modelů, které následně byly testovány po dobu více než tří měsíců, aby se zjistila jejich preciznost (viz tab. 2.4). Také byly vytvořeny 3 další modely (1. - 3. 1.) pro kontrolu, jak velký vliv to bude mít na přesnou předpověď další události po určité sekvenci.

Tab. 2.4: Preciznost modelů Tiresias v závislosti na datu trénování

Datum trénování modelu	Preciznost modelu v závislosti na datu testování							
	8. 11.	23. 11.	8. 12.	23. 12.	8. 1.	23. 1.	8. 2.	23. 2.
1. 11.	0,815	0,785	0,832	0,899	0,899	0,921	0,930	0,921
2. 11.	0,821	0,800	0,835	0,895	0,896	0,921	0,931	0,918
3. 11.	0,822	0,782	0,835	0,898	0,899	0,923	0,930	0,922
4. 11.	0,820	0,793	0,834	0,901	0,898	0,922	0,929	0,921
5. 11.	0,817	0,790	0,833	0,900	0,898	0,921	0,929	0,920
1. - 7. 11.	0,836	0,788	0,829	0,895	0,892	0,917	0,925	0,915
1. 1.	-	-	-	-	0,905	0,927	0,931	0,926
2. 1.	-	-	-	-	0,908	0,926	0,930	0,924
3. 1.	-	-	-	-	0,914	0,933	0,935	0,929

Výsledky testování ukazují preciznost, která neklesá pod **0,78**, přičemž průměr všech hodnot je **0,886**. Pro tyto hodnoty je kritické období testování modelu, což může být z důvodu využívání nových, neznámých zranitelností (*snižování preciznosti*), respektive vydávání a aplikování nových záplat na tyto zranitelnosti (*zvyšování preciznosti*). Pro tak složitý problém dosahuje Tiresias vysoké přesnosti a rovněž vykazuje stabilní výsledky i v případě, že je model trénován měsíce před použitím na testovací datovou sadu.

Shrnutí

V oblasti bezpečnostního monitoringu existuje velké množství možných vylepšení stávajících řešení pomocí umělé inteligence. Tato řešení jsou ovšem ve značné části případů špatně popsána. Dále bude následovat stručný výpis zjištěných nedostatků[11]:

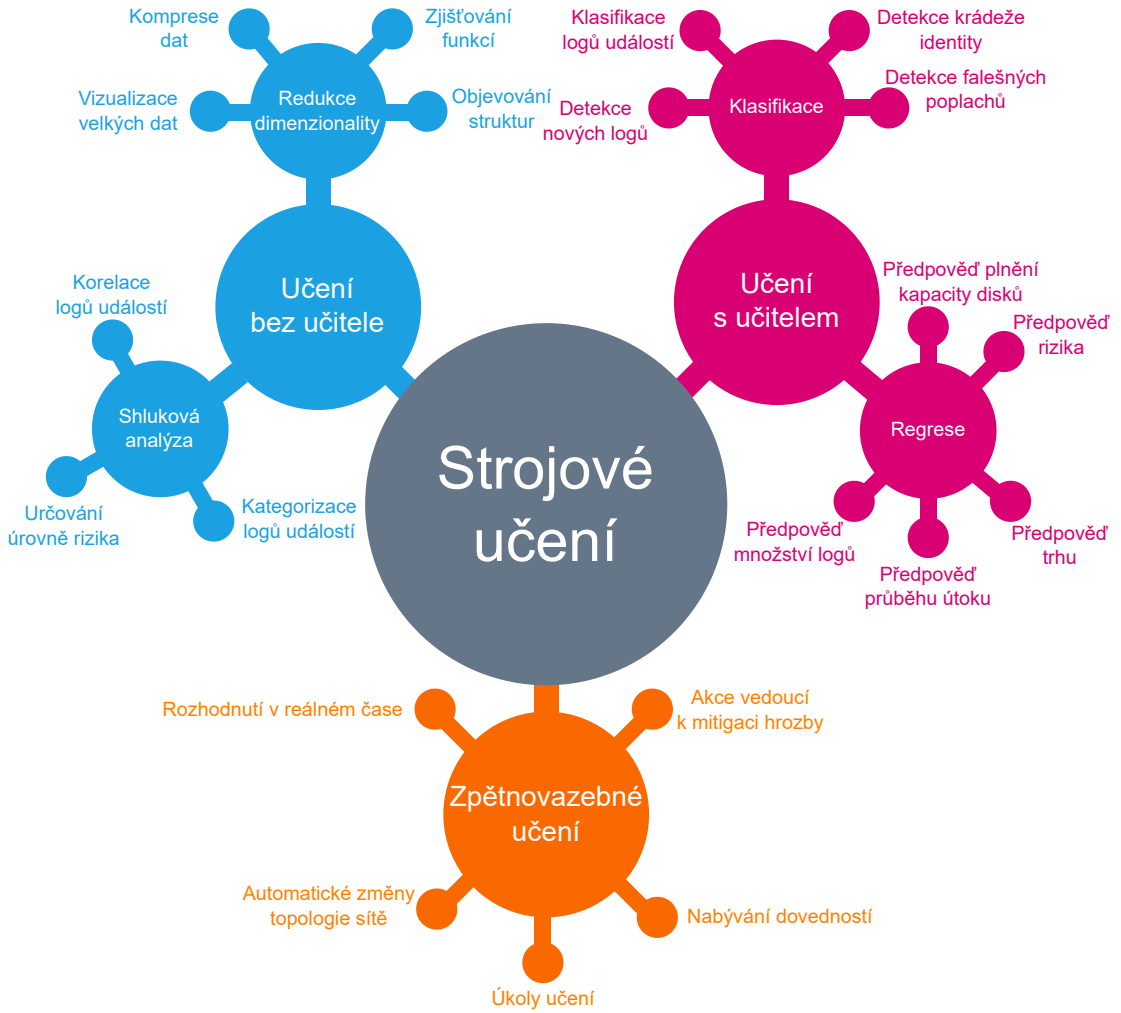
- Chybějící technický popis modelu využité neuronové sítě;
- Absence, či jen malá zmínka využité datové sady;
- Nejasná specifikace možných benefitů řešení;
- Nedostatečné zpracování výsledků;
- Opomenutí porovnání efektivity řešení s ostatními;
- Žádný návrh pro možný budoucí výzkum v dané oblasti.

Deeplog, DeepCASE a Tiresias jsou modely zpracované do podrobna a jsou podložené dalšími podobnými výzkumy, které se týkají dané problematiky. Jediným nedostatkem je zde model Tiresias, ke kterému nebyl připojen zdrojový kód užitý

[11]Tyto nedostatky byly zjištěny v rámci zpracování rešerše při snaze najít co nejefektivnější návrhy na vylepšení bezpečnostního monitoringu v oblasti umělé inteligence. Řešení obsahující zmíněné nedostatky nebyly v rámci této rešerše brány v potaz.

k trénování a validaci modelu neuronové sítě, stejně tak, jako i podrobnější specifikace modelu. I přesto je Tiresias nejvíce ambiciózním řešením, na které by se dalo navázat, protože jeho funkce by mohla být v oblasti bezpečnostního monitoringu užitečná.

S pomocí provedené rešerše byla zpracována možná vylepšení bezpečnostního monitoringu pomocí metod strojového učení (viz obr. 2.7)



Obr. 2.7: Strojové učení a jeho dělení s ohledem na možná vylepšení v oblasti kybernetické bezpečnosti.

3 Metodologie kategorizace logových záznamů

V této části budou rozebrány nutné kroky, které byly učiněny k dosažení vytyčených cílů diplomové práce. Z hlediska zvýšení efektivity, v oblasti bezpečnostního monitoringu s využitím systémů SIEM, byla zvolena technika strojového učení, a to zpracování přirozeného jazyka (viz kapitola 2.3) s účelem klasifikace sekvencí textu. Tato technika je užitečná k analýze, porozumění, organizování a třídění textových dat. Jedná se tedy o účinný nástroj pro zdokonalení procesu zpracování dat surových logů událostí, ze kterých je na první pohled obtížné vyčíst všechny důležité informace, či určit jejich zdroj. Představa implementace této techniky je taková, že by při jejím správném fungování nebylo nutné ručně definovat zdroje jednotlivých zdrojů logů událostí LES – tato kategorizace by tedy probíhala automaticky. Zároveň, díky *fine-tuningu* zvoleného modelu neuronové sítě na tuto konkrétní problematiku, lze kategorizaci logů událostí považovat za dílčí krok pro využití umělé inteligence v rámci budoucího řešení komplexnějšího problému v oblasti bezpečnostního monitoringu.

Tato část diplomové práce je rozdělena na 5 kapitol:

1. **Charakteristika dat** – popis všech potřebných dat a způsob jejich shromáždění s uvedením zdrojů, ze kterých bylo čerpáno. Vytyčení základních kategorií dat a podrobnosti o jejich zdroji (LES);
2. **Zpracování dat** – detailní popis získaných dat a jejich zpracování pomocí skriptu napsaném v jazyku Python. Tato část zahrnuje rozdělení dat na tři datové sady – *trénovací*, *validační* a *testovací*. Datové sady jsou ve formátu *.csv*, což umožňuje pozdější, konečné zpracování dat před jejich předáním na vstup modelu neuronové sítě;
3. **Volba modelu neuronové sítě** – hlavní poznatky o vhodném modelu neuronové sítě zvoleném pro úlohu klasifikace sekvencí textu a popis použitého *frameworku*;
4. **Technický popis modelu** – podrobnější popis fungování zvoleného modelu;
5. **Konečné zpracování dat a proces učení** – vytvoření sešitu *Colab notebook* a popis jednotlivých kroků, které byly provedeny pro uskutečnění zvolené úlohy. V této části jsou popsané tři různé scénáře pro podrobné vyhodnocení výsledků.

3.1 Charakteristika dat

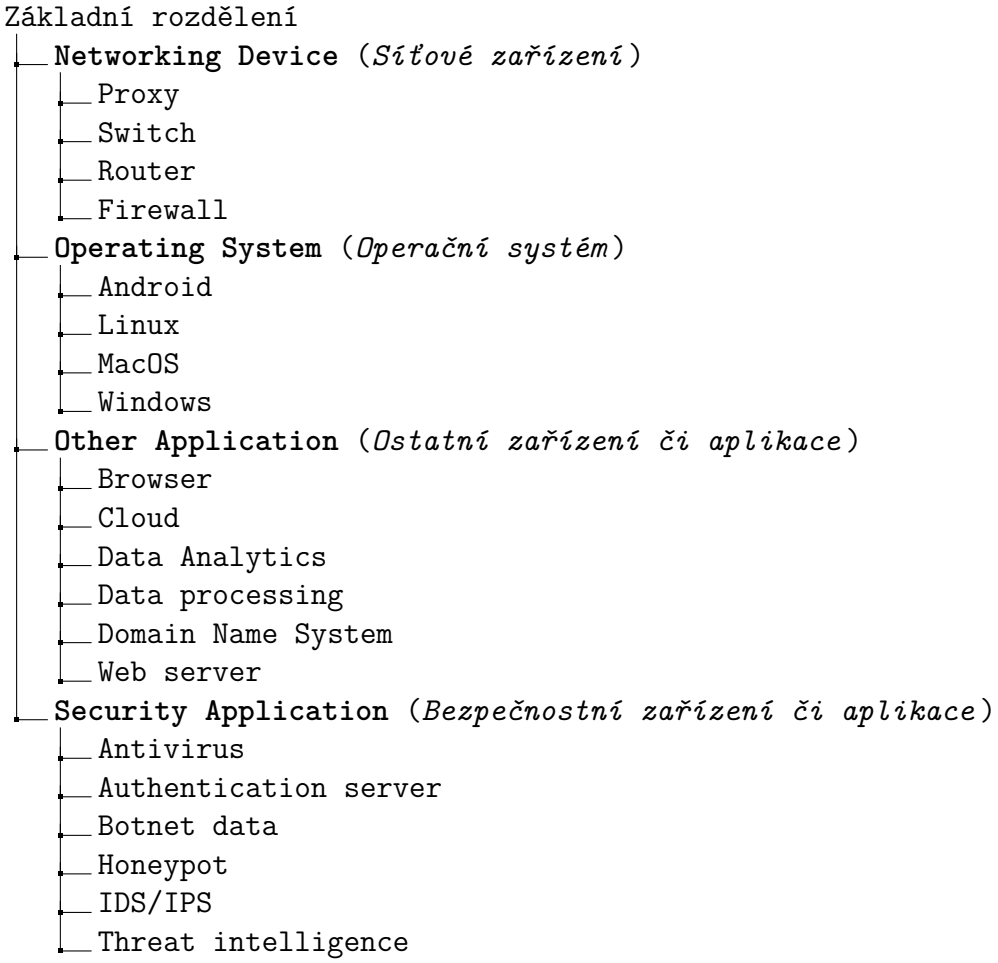
V první řadě bylo nutné vybrat vhodná data pro tvorbu datové sady, s pomocí které se bude konkrétní model neuronové sítě učit. V této podkapitole bude rozebrán způsob volby a proces shromáždění všech užitečných dat a jejich charakteristika. Strojové učení je významně závislé na datech. Je to nejdůležitější aspekt, který umožňuje korektní trénování modelů. Nutno dodat, že ani tolik nezáleží na množství dat, ale spíše na informacích obsažených v datech.

Cílem shromažďování potřebných dat bylo zahrnout co nejvíce možných kategorií zařízení plnících různé funkce, která vytvářejí logy událostí[1]. Těchto zařízení a tedy zdrojů logů událostí (*LES*) bylo vytyčeno `20`. Největší výzvou tohoto procesu bylo nalezení vyhovujících souborů (nejčastěji ve formátu *.txt* a *.log*) s optimální strukturou a obsahem. V rámci sběru dat byly prozkoumány desítky až stovky různých zdrojů na internetu, kde hlavní problém spočíval ve faktu, že prakticky žádná organizace nesdílí nasbírané logy událostí, protože tyto mohou obsahovat citlivé informace z jejich interní sítě, které by se daly zneužít pro případný útok na danou síť.

Nakonec bylo přistoupeno k řešení, které zahrnovalo kolekci dat ze zdrojů v rámci již provedených výzkumů s podmínkou, že jsou data zveřejněná a určená k volnému užití – např. pod licencí `Creative Commons 4.0` (zdroje viz literatura [63, 64, 65, 66, 67]). Nutno podotknout, že nějaká data byla také shromážděna z dostupných zdrojů (tj. např. laboratorní prostředí) vlastním úsilím. Nashromážděná data byla rozdělena do čtyř hlavních kategorií: **Síťové zařízení**, **Operační systém**, **Ostatní aplikace** a **Bezpečnostní aplikace**. S ohledem na relevanci zařízení nebo aplikace v rámci bezpečnostního monitoringu byly nashromážděny logy událostí z proxy serveru, přepínače, směrovače a firewallu v rámci síťových zařízení, dále operační systémy Android, Linux, MacOS a windows. Z kategorie ostatních zařízení byly vybrány logy událostí z webového prohlížeče, cloudového úložiště, aplikace na analýzu a zpracování velkého množství dat, DNS (*Domain Name System*) serveru a webového serveru. Poslední a prakticky i nejdůležitější částí byl výběr logů událostí z bezpečnostních aplikací. Shromážděny byly následující: antivirový program, autentizační server, údaje o aktivitě botnetu, honeypot, IDS/IPS a databáze hrozeb. Rozdělení na kategorie je možné sledovat na obr. 3.1.

Po shromáždění všech 20 souborů dat byly tyto rozřazeny do již zmíněných kategorií, přičemž na základě podkategorií byl každému souboru přidělen unikátní ifentifikátor:

[1] V případě dat z databází hrozeb (*Threat intellligence*) či *Honeypot* a *Botnet data*) se spíše jedná o výpisy nashromážděných dat, které ovšem nelze zanedbat, protože jsou nedílnou součástí bezpečnostního monitoringu.



Obr. 3.1: Vytyčení jednotlivých kategorií a podkategorií LES

- **Source Category** – čtyři základní zdrojové kategorie logů událostí;
- **LES Category** – kategorie konkrétní skupiny zdrojů logů událostí s unikátním identifikátorem *LES Category ID*.

Data, spolu s rozdělením na kategorie a podkategorie s unikátními identifikátory, jsou vyobrazena v tab. 3.1. V tabulce je také zmíněno, z jakého zdroje jednotlivá data pocházejí. Nedílnou součástí je předzpracování všech dat (viz dále – kapitola 3.2).

Tab. 3.1: Rozdělení souborů s logy událostí na jednotlivé kategorie

Source Category	LES Category	LES Category ID	Zdroj
Networking Device	Proxy	0	[62]
	Switch	1	vlastní
	Router	2	vlastní
	Firewall	3	vlastní
Operating System	Android	4	[62]
	Linux	5	[62]
	MacOS	6	[62]
	Windows	7	[62]
Other Application	Browser	8	vlastní
	Cloud	9	[62]
	Data analytics	10	[62]
	Data processing	11	[62]
	Domain Name System	12	[63]
	Web server	13	[62]
Security Application	Antivirus	14	vlastní
	Authentication server	15	[62]
	Botnet data	16	[64]
	Honeypot	17	[65]
	IDS/IPS	18	[63]
	Threat intelligence	19	[66]

3.2 Zpracování datové sady

Téměř každá úloha NLP vyžaduje, aby před trénováním modelu proběhlo předzpracování dat. Pokud by na vstupu neuronové sítě byly pouze soubory se surovými logy událostí ve formátech *.txt* a *.log*, nebylo by možné se dopracovat k optimálnímu výsledku. Modely hlubokého učení nemohou používat nezpracovaný text přímo. V závislosti na povaze úlohy mohou být metody předzpracování různé. Důležité je převést data do jednotné podoby (formátu *.csv*) a přidělit jim identifikační hodnoty, se kterými bude konkrétní model neuronové sítě schopen pracovat a klasifikovat pomocí nich konkrétní skupinu zdroje logů událostí[2] (*LES Category ID*). V konkrétním případě bylo předzpracování textu provedeno následujícími kroky:

1. **Integrace dat** – sjednocení datových formátů, v tomto konkrétním případě do podoby CSV (*Comma-Separated Values*). CSV je v podstatě jednoduchý

[2]Nutno podotknout, že před samotným procesem učení modelu musí data projít tzv. *tokenizací* a dále musí být převedena na tenzory, aby s nimi model mohl pracovat – viz dále kapitola 3.5.

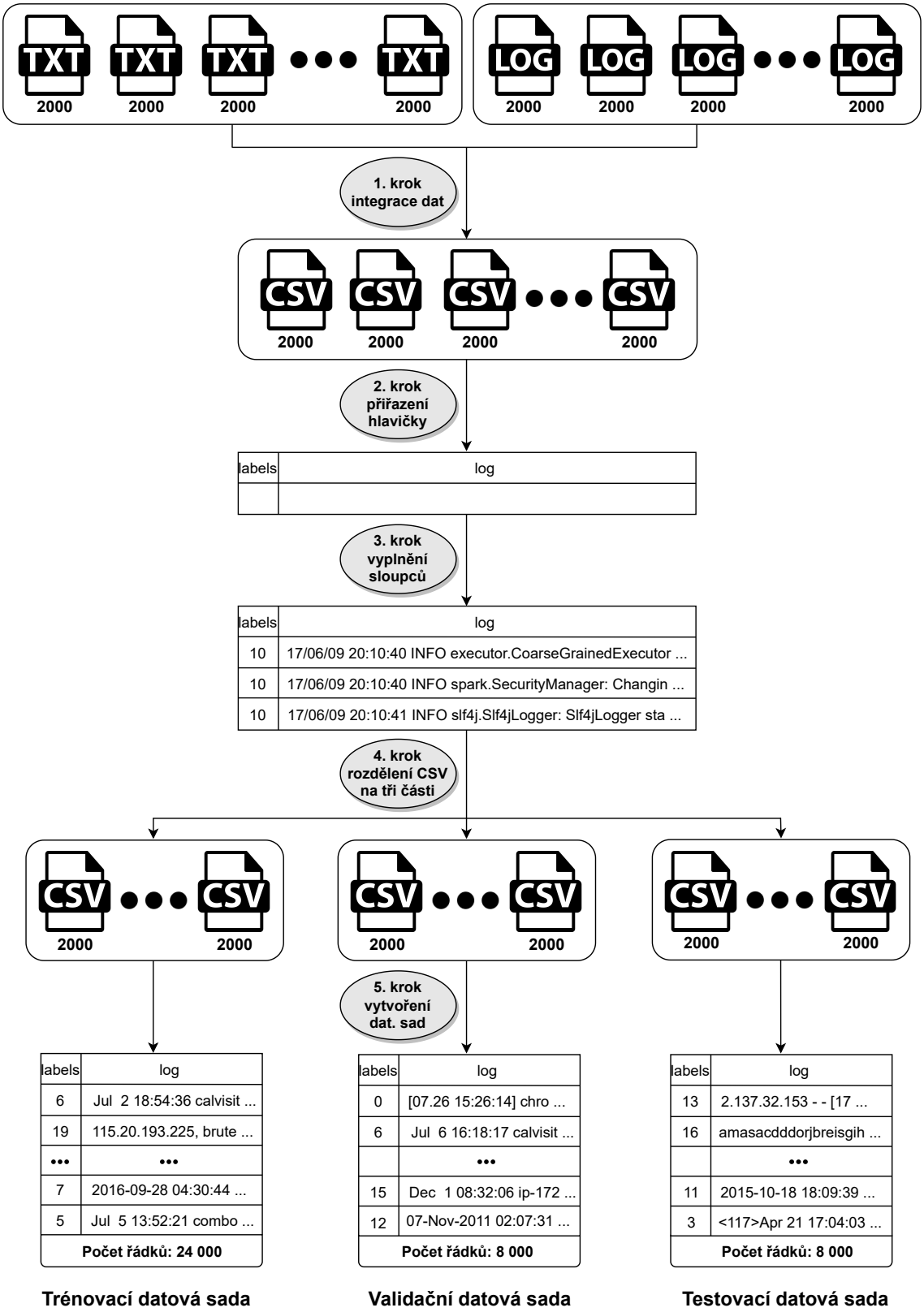
formát souboru, který se používá k ukládání tabulkových dat (čísel a textu), například tabulky v prostém textu. Na prvním řádku může být umístěna hlavička, která popisuje názvy jednotlivých sloupců. Formát hlavičky je stejný jako formát ostatních sloupců. V případě, že je v souboru definována hlavička, měl by být počet sloupců na každém řádku (*text vždy musí být rozdělen oddělovačem – nejčastěji pomocí "," a ";"*) stejný, jinak hlavička nedává smysl. V jazyce Python lze CSV data načítat různými způsoby (např. s pomocí knihovny *Pandas*), v čemž spočívá značná výhoda užívání tohoto formátu v oblasti strojového učení (zároveň se jedná o nejvíce užívaný formát k reprezentaci dat pro účel strojového učení);

2. **Transformace dat** – proces transformace dat spočívá ve změně formátu, struktury nebo hodnot dat. Je základním aspektem většiny úloh integrace dat. Nástroje a technologie používané pro transformaci dat se mohou značně lišit v závislosti na formátu, struktuře, složitosti a objemu transformovaných dat. V tomto konkrétním případě byla transformace dat provedena v pěti dílčích krocích[3]:
   (a) **Vytvoření** CSV souborů (jeden na každý zdrojový soubor), kde každý soubor disponuje 2000 záznamy;
   (b) **Přiřazení** hlavičky CSV souboru o dvou skupinách – *Log* a *lesID*;
   (c) **Vyplnění** dat sloupce s hlavičkou *Log* ze souborů obsahujících surové logy událostí a přiřazení odpovídajících hodnot do sloupců s hlavičkou *lesID* podle zdroje daného logu;
   (d) **Rozdělení** CSV souborů na tři části v poměru 60:20:20 (poměr počtu řádků na každý soubor je tedy 1200:400:400) pro vytvoření dílčích datových sad pro *trénování*, *validaci* a *testování* modelu neuronové sítě;
   (e) **Sloučení** všech souborů z dílčích datových sad do tří CSV souborů tvořících finální datovou sadu;

Kroky uskutečněné ke zpracování datové sady jsou znázorněny také na obr. 3.2. Na výstupu tedy byly vytvořeny tři CSV soubory, kde první obsahuje 12000 záznamů (*trénovací datová sada*), druhý 4000 záznamů (*validační datová sada*) a poslední také 4000 (*testovací datová sada*).

Pro dokončení procesu transformace dat bylo ještě nutné náhodně promíchat všechny řádky v datových sadách. Tímto tedy byly vytvořeny tři finální datové sady, jejichž výpis je možné vidět na obr. 3.3.

[3]Tyto kroky byly provedeny pomocí sady skriptů napsaných v jazyku Python, které jsou dostupné z URL: <`https://github.com/xsedla1f/zpracovaniDat`>



Obr. 3.2: Proces zpracování datové sady [28]

```
Trénovací datová sada:
#################################################################################################
       labels                                                                                 log
0           6  Jul  2 18:54:36 calvisitor-10-105-163-202 kernel[0]: ARPT: 661915.168735: wl0: MD...
1          19  115.20.193.225, brute force host, 2022-05-12
2           0  [10.30 20:47:14] chrome.exe - proxy.cse.cuhk.edu.hk:5070 open through proxy proxy...
3          12  07-Nov-2011 01:42:14.224 queries: info: client 10.1.60.203#61108: query: jabber.u...
4          12  07-Nov-2011 00:10:25.080 queries: info: client 10.1.100.5#58499: query: usma.blue...
...       ...                                                                                  ...
23995      13  122.166.142.108 - - [17/May/2015:17:05:26 +0000] "GET /presentations/logstash-pup...
23996       2  05/17/2022 22:33:07 [info][WiFi][ DISASSOC - MAC=be:7d:d7:6b:da:59, Reason=34; WG...
23997      11  2015-10-18 18:06:40,108 ERROR [RMCommunicator Allocator] org.apache.hadoop.mapred...
23998       7  2016-09-28 04:30:44, Info                  CBS    Session: 30546174_101172793 ini...
23999       5  Jul  5 13:52:21 combo ftpd[6592]: connection from 211.72.2.106 () at Tue Jul  5 1...

[24000 rows x 2 columns]
#################################################################################################

Validační datová sada:
#################################################################################################
      labels                                                                                  log
0          0  [07.26 15:26:14] chrome.exe *64 - api.github.com:443 open through proxy proxy.cse...
1          6  Jul  6 16:18:17 calvisitor-10-105-162-178 kernel[0]: ARPT: 740501.982555: wl0: MD...
2         10  17/06/09 20:11:08 INFO spark.CacheManager: Partition rdd_42_34 not found, computi...
3         12  07-Nov-2011 02:06:17.257 queries: info: client 127.0.0.1#60230: query: smtp.afit2...
4          4  03-17 16:15:48.106  2227  2227 I PhoneStatusBar: setSystemUiVisibility vis=c00007...
...      ...                                                                                  ...
7995       3  <117>Apr 21 17:03:41 172.25.164.254 CEF:0|Fortinet|Fortigate|v6.0.9|00013|traffic...
7996      11  2015-10-18 18:07:23,189 WARN [RMCommunicator Allocator] org.apache.hadoop.ipc.Cli...
7997      18  06/02-01:49:14.253731  [**] [1:2014237:2] ET CURRENT_EVENTS DRIVEBY Blackhole - P...
7998      15    Dec  1 08:32:06 ip-172-31-27-153 sshd[24925]: Invalid user guest from 67.205.20...
7999      12  07-Nov-2011 02:07:31.458 queries: info: client 127.0.0.1#64886: query: smtp.usma....

[8000 rows x 2 columns]
#################################################################################################

Testovací datová sada:
#################################################################################################
      labels                                                                                  log
0         13  2.137.32.153 - - [17/May/2015:23:05:28 +0000] "GET /style2.css HTTP/1.1" 200 4877...
1         16  amasacdddorjbreisgihduhicfeoiapu.shop CNAME . ; Botnet C2 - confidence level: 100...
2          6  Jul  8 04:16:21 calvisitor-10-105-161-176 QQ[10018]: FA||Url||taskID[2019353853] ...
3         16  peulnm16.top CNAME . ; Botnet C2 - confidence level: 100% (2022_01_03), see https...
4         16  tobday02.top CNAME . ; Botnet C2 - confidence level: 100% (2022_01_03), see https...
...      ...                                                                                  ...
7995      17  2021-01-25T03:17:14.369214Z\t115.74.213.139\tlogin attempt [ubnt/123456] failed\t...
7996      12  07-Nov-2011 17:17:45.468 queries: info: client 10.1.60.253#60379: query: ns1.usma...
7997      17  2021-01-25T05:27:37.465844Z\t99.195.114.207\tlogin attempt [admin/guest] failed\t...
7998      11  2015-10-18 18:09:39,729 WARN [LeaseRenewer:msrabi@msra-sa-41:9000] org.apache.had...
7999       3  <117>Apr 21 17:04:03 172.25.164.254 CEF:0|Fortinet|Fortigate|v6.0.9|00013|traffic...

[8000 rowsx 2 columns]
#################################################################################################
```

Obr. 3.3: Výsledné datové sady

3.3 Volba modelu neuronové sítě

Pro úlohu klasifikace textu byl zvolen model **Google BERT**. Tento model byl vytvořen v roce 2018 a navzdory tomu, že již vyšlo velké množství modelů, které jej v mnoha ohledech předčily, stále vykazuje velice dobré výsledky. Rok 2018 byl v oblasti NLP přelomový [68]. Modely přeneseného učení, jako např. ELMO společnosti

Allen AI, Open-GPT společnosti OpenAI a BERT společnosti Google, umožnily výzkumníkům v době jejich vzniku překonat hned několik benchmarků s minimálním dolaďováním na konkrétní úlohy a poskytly zbytku komunity NLP předtrénované modely. Tyto modely lze (s menším množstvím dat a kratším výpočetním časem) doladit a implementovat tak, aby poskytovaly *state-of-the-art* výsledky [68].

V případě BERT se jedná o předem naučený model, který je možné pomocí *fine-tuningu* adaptovat na konkrétní úlohu a využít jej k extrakci vysoce kvalitních jazykových rysů z textových dat s vlastními daty a vytvořit tak velice přesné předpovědi. Tohoto lze dosáhnout prostřednictvím dosazení nové vrstvy neuronů na konec modelu, díky které je možné jej přizpůsobit konkrétnímu problému. Předtrénované váhy modelu BERT již kódují mnoho informací o přirozeném jazyku. V důsledku toho zabere trénování předem naučeného modelu mnohem méně času, tedy lze si to představit, jako kdyby již spodní vrstvy této sítě prošly stovkami hodin trénování a potřebovaly pouze nepatrně doladit, přičemž jejich výstupy budou použity na korespondující klasifikační úlohu. Díky těmto predispozicím je u modelu BERT doporučeno trénovat pouze ve 2-4 epochách, což výrazně ušetří čas [68].

PyTorch

PyTorch je open source framework pro strojové učení založený na knihovně Torch, používaný zejména pro aplikace zpracování přirozeného jazyka (NLP) [69].

PyTorch je knihovna založená na tenzorech, optimalizovaná pro hluboké učení. Tato knihovna je založená na jazyce *Python* a *Torch* a se používá především pro aplikace využívající GPU a CPU. PyTorch je upřednostňován před ostatními frameworky hlubokého učení, jako jsou *TensorFlow* a *Keras*, protože využívá dynamické výpočetní grafy. PyTorch umožňuje spouštět a testovat části kódu v reálném čase. Uživatelé tak nemusí čekat na implementaci celého kódu, aby mohli zkontrolovat funkčnost [69].

Konkrétně užitá knihovna je dostupná pod názvem **PyTorch-Transformers** a obsahuje předtrénované modely architektury Transformers: BERT, GPT, GPT-2, Transformers-XL, XLNet, XLM, RoBERTa, DistilBERT [70]. Hlavními úlohami NLP, pro které je tato knihovna vhodná, jsou *SequenceClassification* a *QuestionAnswering*.

3.4 Technický popis modelu

Jak již bylo zmíněno v podkapitole 2.2.3, model BERT je založen na architektuře Transformers. V současné době jsou dostupné jeho dvě varianty (Base a Large), co se týče počtu parametrů, transformer bloků, skrytých vrstev a *attention heads* a několik subvariant (Multilingual a Chinese – Cased/Uncased). Dále následuje výpis základních variant modelu Google BERT [71]:

- **BERT Base (*Cased/Uncased*[4])** – 12 transformer bloků, 786 skrytých vrstev, 12 *self-attention heads*, 110 milionů parametrů;
- **BERT Large (*Cased/Uncased*)** – 24 transformer bloků, 1024 skrytých vrstev, 16 *self-attention heads*, 340 milionů parametrů;
- **BERT Base Multilingual (*Cased*)** – 12 transformer bloků, 786 skrytých vrstev, 12 *self-attention heads*, 110 milionů parametrů;
- **BERT Base Chinese** – 12 transformer bloků, 786 skrytých vrstev, 12 *self-attention heads*, 110 milionů parametrů.

Dále je důležité porozumět speciálním tokenům, které jsou používány pro učení a *fine-tuning* modelu BERT [72]:

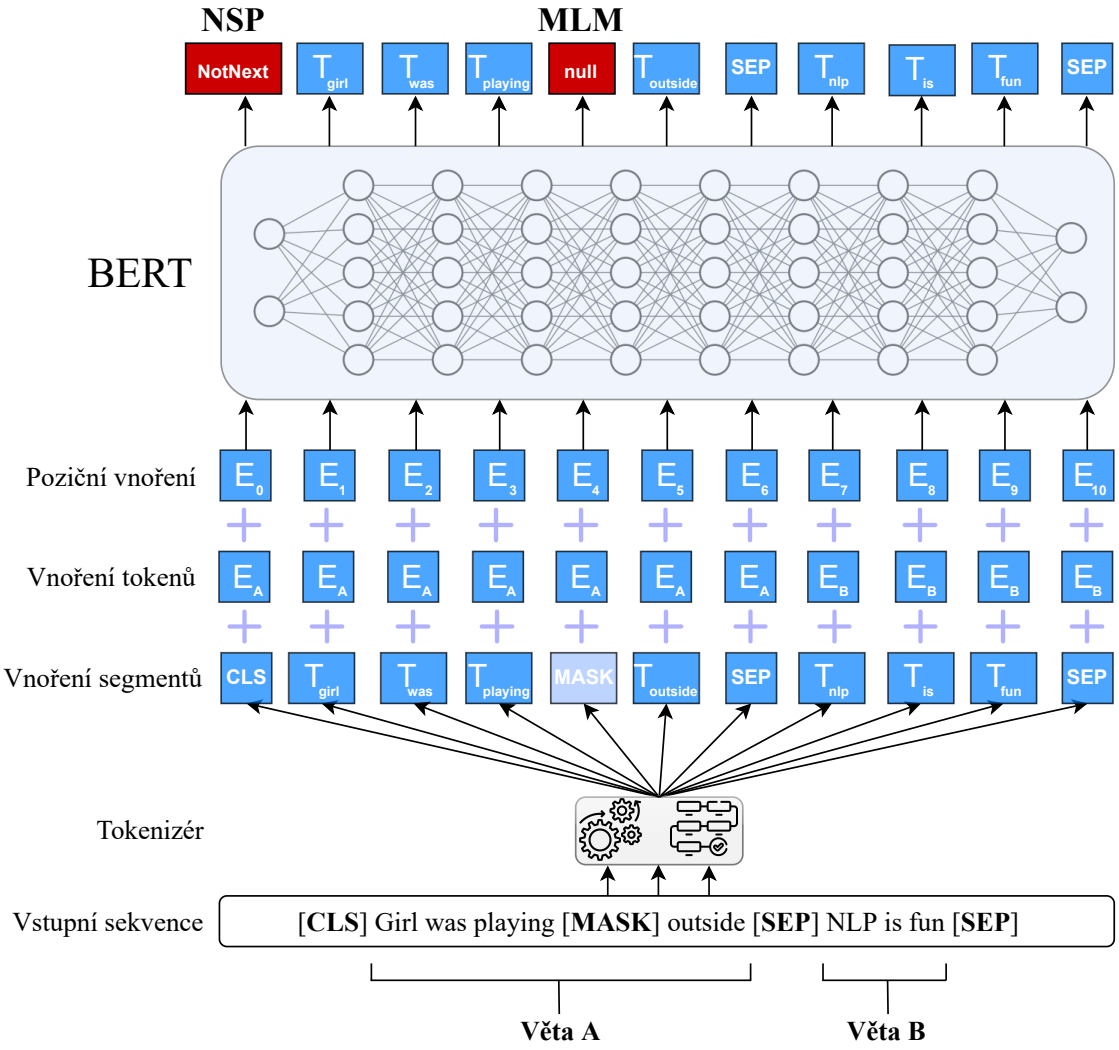
- **[CLS]** – tento speciální token je na začátku každé sekvence dat a slouží pro rozdělení sekvencí dat (kde každá sekvence může obsahovat n vět);
- **[SEP]** – token sloužící k rozdělení vět v rámci jedné sekvence dat;
- **[MASK]** – token užívaný k maskování slov (pouze pro předtrénování modelu).

Model BERT je předtrénován na dvou datových sadách bez učitele (*unsupervised learning*), zejména na dvou NLP úlohách – jazykové modelování pomocí masek MLM (*Masked Language Modeling*) a predikce další věty NSP (*Next Sentence Prediction*). Znázornění procesu předtrénování modelu BERT je možno vidět na obr. 3.4.

MLM

BERT maskuje zhruba 15 % slov ve větě (nahrazuje je speciálním tokenem [MASK]) a s využitím kontextu věty je následně předpovídá. MLM je zejména účinné při snaze naučit model hluboké reprezentace (*deep representations*), tzn. naučit jej více reprezentací, čímž bude v následných úlohách zlepšen výkon modelu. Kupříkladu reprezentace nižší vrstvy určitých modelů jsou užitečné pro syntaktické úlohy, zatímco reprezentace vyšší vrstvy pro sémantické úlohy [71]. Maskovaná slova ovšem nebyla při předběžném trénování modelu BERT vždy zaměněna za [MASK] tokeny, protože se při *fine-tuningu* modelu tyto tokeny nikdy neobjeví, takže bylo maskování slov provedeno následovně [71]:

[4]Rozdíl mezi těmito dvěma variantami spočívá v tom, že BERT cased přijímá k tokenizaci nezměněný vstupní test, kdežto BERT uncased přijímá pouze malá písmena – *lowercase*.



Obr. 3.4: Proces předtrénování modelu Google BERT [73]

- V 80 % případů byla slova nahrazena zmíněnými [MASK] tokeny;
- V 10 % případů byla slova nahrazena jinými, náhodnými slovy;
- V 10 % případů zůstala slova nezměněna.

NSP

Podstata MLM je tedy založena na pochopení vztahu mezi slovy. Model BERT byl také trénován na úloze předpovědi další věty, z čehož vyplývá, že se jedná o pochopení vztahu mezi jednotlivými větami, přičemž 50 % dat je označeno jako *isNext*, kde věta B ze vstupní sekvence je právě další větou věty A. Dalších 50 % dat je označeno jako *notNext*, kde věta B není další větou po větě A, ale libovolnou náhodnou větou ze souboru dat.

3.5 Konečné zpracování dat a proces učení

Pro potřeby učení byl využit model **BERT Base** (*Uncased*), který obsahuje 12 transformer bloků (vrstev), 768 skrytých vrstev, 12 *self-attention heads* a 110 milionů parametrů [71]. Celý proces učení byl realizován ve virtuálním prostředí `Google Colaboratory`. *Colaboratory by Google* (zkráceně Google Colab) je běhové prostředí založené na zápisníku Jupyter Notebook, které umožňuje spouštět kód v režimu cloudu a tedy bez nutnosti instalovat jakékoliv vývojové prostředí. To znamená, že je možné trénovat rozsáhlé modely hlubokého učení bez nutnosti disponovat pokročilejšími výpočetními prostředky, které jsou pro úlohy hlubokého učení nezbytné [74].

Nejprve bylo provedeno načtení připravených datových sad ve formátu *.csv* do proměnných pomocí knihovny Pandas – konkrétně:

- `Train_data_merged_shuffled.csv -> bert_train`;
- `Validation_data_merged_shuffled.csv -> bert_valid`;
- `Test_data_merged_shuffled.csv -> bert_test`.

Po jejich načtení do proměnných bylo uskutečněno doplnění speciálních tokenů `[CLS]` a `[SEP]` a převedení všech sekvencí dat na malá písmena – *lowercase* (výpis trénovací datové sady viz obr. 3.5). Přidělení speciálních tokenů bylo provedeno způsobem pro zpracování jednotlivých sekvencí textu (*single sequence*) [75].

```
       labels                                                log
0           1  [CLS] may 19 08:44:36 192.168.1.1 60: may 19 0...
1           9  [CLS] nova-api.log.1.2017-05-16_13:53:08 2017-...
2           2  [CLS] 05/17/2022 18:27:26 [info][wifi][ mac be...
3          10  [CLS] 17/06/09 20:10:54 info python.pythonrunn...
4           4  [CLS] 03-17 16:15:08.076  1702  2633 d powerma...
...       ...                                                ...
23995       1  [CLS] may 19 08:58:31 192.168.1.1 70: may 19 0...
23996      19  [CLS] 1.165.137.23, brute force host, 2022-05-...
23997      11  [CLS] 2015-10-18 18:02:47,497 info [rmcommunic...
23998      16  [CLS] gtersx3.cfd cname . ; botnet c2 - confid...
23999      16  [CLS] bokkuvirde.ddns.net cname . ; botnet c2 ...

[24000 rows x 2 columns]
```

Obr. 3.5: Výpis trénovací datové sady po doplnění speciálních tokenů `[CLS]` a `[SEP]`

Dalším krokem byla tokenizace dat, provedena pomocí tokenizéru `BertTokenizer` pro neuronovou síť `bert-base-uncased`. V této fázi proběhlo rozdělení každého záznamu z datové sady na jednotlivé tokeny (obr. 3.6). Výpis tokenizovaného záznamu z trénovací datové sady lze sledovat na obr. 3.7.

```
Tokenizace vstupů jednotlivých datových sad:

input_ids = [tokenizer.convert_tokens_to_ids(x) for x in tokenized_texts];
input_ids_valid = [tokenizer.convert_tokens_to_ids(x) for x in tokenized_texts_valid];
input_ids_test = [tokenizer.convert_tokens_to_ids(x) for x in tokenized_texts_test];
```

Obr. 3.6: Tokenizace logů událostí jednotlivých datových sad

```
Tokenizace prvního záznamu z trénovací datové sady:
['[CLS]', 'may', '19', '08', ':', '44', ':', '36', '192',
'.', '168', '.', '1', '.', '1', '60', ':', 'may', '19',
'08', ':', '43', ':', '34', '.', '07', '##9', ':', '%',
'link', '-', '3', '-', 'up', '##down', ':', 'interface',
'fast', '##eth', '##ern', '##et', '##0', '/', '5', ',',
'changed', 'state', 'to', 'up', '[SEP]']
Délka tokenizovaného záznamu:  50
```

Obr. 3.7: Výpis prvního záznamu z trénovací datové sady po tokenizaci

Po tokenizaci jednotlivých záznamů byly definovány proměnné určující důležité parametry pro proces učení modelu neuronové sítě:

- LEARNING_RATE = 4e-5 – rychlost učení;
- BATCH_SIZE = 8 – počet vzorů pro trénování použitých v jedné iteraci;
- NUMBER_OF_EPOCHS = 2 – počet epoch;
- DROPOUT = 0.1 – regularizační metoda.

Jelikož v jednotlivých záznamech logů událostí nelze s přesností určit začátek či konec věty a tudíž byl speciální token `[CLS]` vložen na začátek každého záznamu a token `[SEP]` na konec. Nutno podotknout, ze model Google BERT podporuje délku sekvencí tokenů do maximální délky 512 [76].

Při překročení této hodnoty by docházelo k chybným hlášením. Pro zajištění délky tokenů "`len(token) < 512`" byla data postupně prověřena pomocí metody `process_tokenized_text`, která provedla kontrolu délky jednotlivých záznamů a pokud délka přesahovala hodnotu `MAX_LEN = 512`, byly do takových datových záznamů vloženy tokeny pro separaci `[SEP]` (viz obr. 3.8).

```
def _process_tokenized_text(MAX_LEN, tokenized_texts, segments):
  for i in range(len(tokenized_texts)):
    token = tokenized_texts[i]

    if len(token) > MAX_LEN:
      token = token[:(MAX_LEN-1)]
      token.append("[SEP]")

    segment_id = [0] * MAX_LEN

    segments.append(segment_id.copy())
    tokenized_texts[i] = token.copy()
  return segments, tokenized_texts

segments = []
segments_valid = []
segments_test = []

segments, tokenized_texts = _process_tokenized_text(MAX_LEN, tokenized_texts, segments)
segments_valid, tokenized_texts_valid = _process_tokenized_text(MAX_LEN, tokenized_texts_valid, segments_valid)
segments_test, tokenized_texts_test = _process_tokenized_text(MAX_LEN, tokenized_texts_test, segments_test)
```

Obr. 3.8: Metoda pro detekci a oříznutí sekvencí dat o délce větší než `MAX_LEN`

Tímto je sekvence dat větší jak `MAX_LEN` oříznuta, protože veškeré tokeny, které jsou za touto pozicí, jsou zahozeny. To ovšem nezpůsobí žádné potíže, ani to neovlivní relevanci dat v procesu učení, validace a trénování, protože drtivá většina sekvencí dat má délku menší než 100.

Nakonec, před vstupem dat do neuronové sítě bylo nutné data převést do formátu, který bude neuronová síť schopna zpracovat. Převedení do tenzorů je realizováno pomocí knihovny PyTorch (viz obr. 3.9). Výpis jednotlivých tenzorů lze sledovat na obr. 3.10. Tenzory jsou matematické objekty, které zobecňují skaláry, vektory a matice do vyšších dimenzí. Stručně řečeno, jednorozměrný tenzor lze reprezentovat jako vektor. Dvourozměrný tenzor lze reprezentovat jako matici. I když je snadné zobecnit tenzory jako vícerozměrné matice od nuly do N rozměrů, je důležité si uvědomit, že tenzory jsou dynamické. To znamená, že se tenzory při interakci s jinými matematickými entitami transformují. Naproti tomu matice tuto vlastnost vždy nemají. Základním rozdílem mezi tenzory a maticemi je: každý tenzor 2. řádu lze reprezentovat jako matici, ale ne každá matice je ve skutečnosti tenzor 2. řádu [77].

```
Vytvoření tenzorů z jednotlivých paramtetrů datových sad:

train_inputs = torch.tensor(input_ids)
train_labels = torch.tensor(labels)
train_masks = torch.tensor(attention_masks)
train_segments = torch.tensor(segments)

validation_inputs = torch.tensor(input_ids_valid)
validation_labels = torch.tensor(labels_valid)
validation_masks = torch.tensor(attention_masks_valid)
validation_segments = torch.tensor(segments_valid)

test_inputs = torch.tensor(input_ids_test)
test_labels = torch.tensor(labels_test)
test_masks = torch.tensor(attention_masks_test)
test_segments = torch.tensor(segments_test)
```

Obr. 3.9: Konverze sekvencí dat na tenzory pomocí PyTorch

Nutno podotknout, že vytvořené tenzory jsou všechny stejné velikosti a tím pádem s nimi může model bez problému pracovat v jednotlivých fázích – trénování, validace a testování. Popis vytvořených tenzorů s jejich velikostmi:

- **inputs_tensor** – tenzor vytvořený z tokenizovaných vstupních dat:
  - `train_inputs` – 24 000 x 512;
  - `validation_inputs` – 8 000 x 512;
  - `test_inputs` – 8 000 x 512.
- **labels_tensor** – tenzor obsahující všechny identifikátory LES Category ID. Tento tenzor je jedinou výjimkou v tom, že se jeho druhý rozměr rovná 1, protože hodnota LES ID je uchována pouze v jednom tokenu:
  - `train_labels` – 24 000 x 1;
  - `validation_labels` – 8 000 x 1;
  - `test_labels` – 8 000 x 1.
- **masks_tensor** – tenzor, který uchovává informace o tom, jaký token obsahuje hodnotu ze vstupních dat (1) a jaký token je prázdný (0):
  - `train_masks` – 24 000 x 512;
  - `validation_masks` – 8 000 x 512;
  - `test_masks` – 8 000 x 512.

- **segments_tensor** – tenzor, který slouží k identifikaci věty A (0) a věty B (1) v páru sekvencí (*pair of sequences*) a případně prázdné tokeny po větě B (0). V tomto konkrétním případě je ovšem užita jediná sekvence (single sequence), tudíž jsou všechny tenzory nulové:
  - `train_segments` – 24 000 x 512;
  - `validation_segments` – 8 000 x 512;
  - `test_segments` – 8 000 x 512.

```
Tenzory vytvořené z trénovací datové sady

train_inputs:
  tensor([[  101,  2089,  2539,  ...,     0,     0,     0],
        [  101,  6846,  1011,  ...,     0,     0,     0],
        [  101,  5709,  1013,  ...,     0,     0,     0],
        ...,
        [  101,  2325,  1011,  ...,     0,     0,     0],
        [  101, 14181,  2545,  ...,     0,     0,     0],
        [  101,  8945, 19658,  ...,     0,     0,     0]])

train_labels:
  tensor([ 1,  9,  2,  ..., 11, 16, 16])

train_masks:
  tensor([[1., 1., 1.,  ..., 0., 0., 0.],
        [1., 1., 1.,  ..., 0., 0., 0.],
        [1., 1., 1.,  ..., 0., 0., 0.],
        ...,
        [1., 1., 1.,  ..., 0., 0., 0.],
        [1., 1., 1.,  ..., 0., 0., 0.],
        [1., 1., 1.,  ..., 0., 0., 0.]])

train_segments:
  tensor([[0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        ...,
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0]])
```

Obr. 3.10: Výpis tenzorů pro trénování neuronové sítě

Před trénováním, validací a testováním modelu je ještě nutné všechny tenzory sjednotit. V rámci PyTorch existuje knihovna přímo navržená pro tento úkol – torch.utils.data [78]. V rámci této knihovny byly užity 4 metody (viz obr. 3.11):

- **TensorDataset** – sloučení všech připravených tenzorů do jedné datové sady:
  - `train_data`;
  - `validation_data`;
  - `test_data`.
- **RandomSampler** – náhodné promíchání prvků datové sady TensorDataset (pouze pro trénovací datovou sadu):
  - `train_sampler`.
- **SequentialSampler** – řadí prvky datové sady postupně, vždy ve stejném pořadí:
  - `validation_sampler`;
  - `test_sampler`.
- **DataLoader** – automatické sdružování jednotlivých načtených prvků dat do dávek pomocí argumentů `batch_size`, sampler:
  - `train_dataloader`;
  - `validation_dataloader`;
  - `test_dataloader`.

```
Načtení datových sad z tenzorů:

train_data = TensorDataset(train_inputs, train_segments, train_masks, train_labels)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)

validation_data = TensorDataset(validation_inputs, validation_segments, validation_masks,
validation_labels)
validation_sampler = SequentialSampler(validation_data)
validation_dataloader = DataLoader(validation_data, sampler=validation_sampler,
batch_size=batch_size)

test_data = TensorDataset(test_inputs, test_segments, test_masks, test_labels)
test_sampler = SequentialSampler(test_data)
test_dataloader = DataLoader(test_data, sampler=test_sampler, batch_size=batch_size)
```

Obr. 3.11: Vytvoření finálních datových sad z tenzorů

Posledním krokem před započetím procesu trénování, validace a testování bylo vyprázdnění mezipaměti a načtení modelu pro klasifikaci sekvencí textu s definováním proměnné `num_labels` = 20, a načtení optimalizátoru `BertAdam` – viz obr. 3.12.

```
torch.cuda.empty_cache()
model = None

model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=20)
model.cuda()

optimizer = BertAdam(optimizer_grouped_parameters,
                     lr=LEARNING_RATE,
                     warmup=.1)
```

Obr. 3.12: Konečné kroky před učením modelu

Pro trénování, validaci a testování modelu byly provedeny desítky testovacích běhů pro nalezení nejlepšího výsledku. Po prověření všech těchto běhů byly vybrány tři různé scénáře s nejlepšími výsledky trénování, validace a testování modelu BERT. Popis těchto scénářů je možné vidět v tab. 3.2.

Tab. 3.2: Vytvoření scénářů pro model BERT

Scénář	Trénovací datová sada	Validační datová sada	Testovací datová sada	Poměr dat. sad	Rychlost učení
1	24 000	8 000	8 000	60:20:20	4e-5
2	8 000	8 000	24 000	20:20:60	2e-6
3	4 000	4 000	32 000	10:10:80	2e-6

K vytvoření zdrojového kódu v sešitu `Google Colab` bylo čerpáno z následujících zdrojů literatury: [28, 75, 79, 80, 81]

4 Výsledky měření

Jak již bylo řečeno v rámci kapitoly 3.5, byly vytvořeny tři různé scénáře trénování, validace a testování modelu neuronové sítě `Google BERT`. Tyto finální scénáře jsou výsledkem desítek předchozích testů, kde bylo experimentováno zejména s jinými hodnotami rychlosti učení a poměru jednotlivých datových sad. Z těchto testovacích scénářích byl vyvozen závěr, že nejideálnější rychlost učení je `2e-6` a nejúčinější poměr datových sad, co se týče kompromisu mezi rychlostí procesu a kvalitou výsledků, je `20:20:60` – trénovací datová sada:validační datová sada:testovací datová sada. Každé měření ovšem bylo svým způsobem unikátní, protože každé vytvoření nové datové sady (viz kapitola 3.2) bylo jiné z hlediska náhodného pořadí datových sad. Co se týče časové náročnosti testovacích scénářů, nikdy doba trénování, validace, testování a vyhodnocení modelu nepřesáhla 3 hodiny (hlavně z důvodu konstantní velikosti logů událostí celkem – popsáno v kapitole 3.1). Také je nutno podotknout, že se nejedná o binární klasifikaci označení sekvencí dat, nýbrž o klasifikaci obsahující 20 různých označení a nebylo tedy možné vygenerovat matici záměn jako takovou, ale jen její napodobeninu, kde jsou vyobrazeny špatné předpovědi.

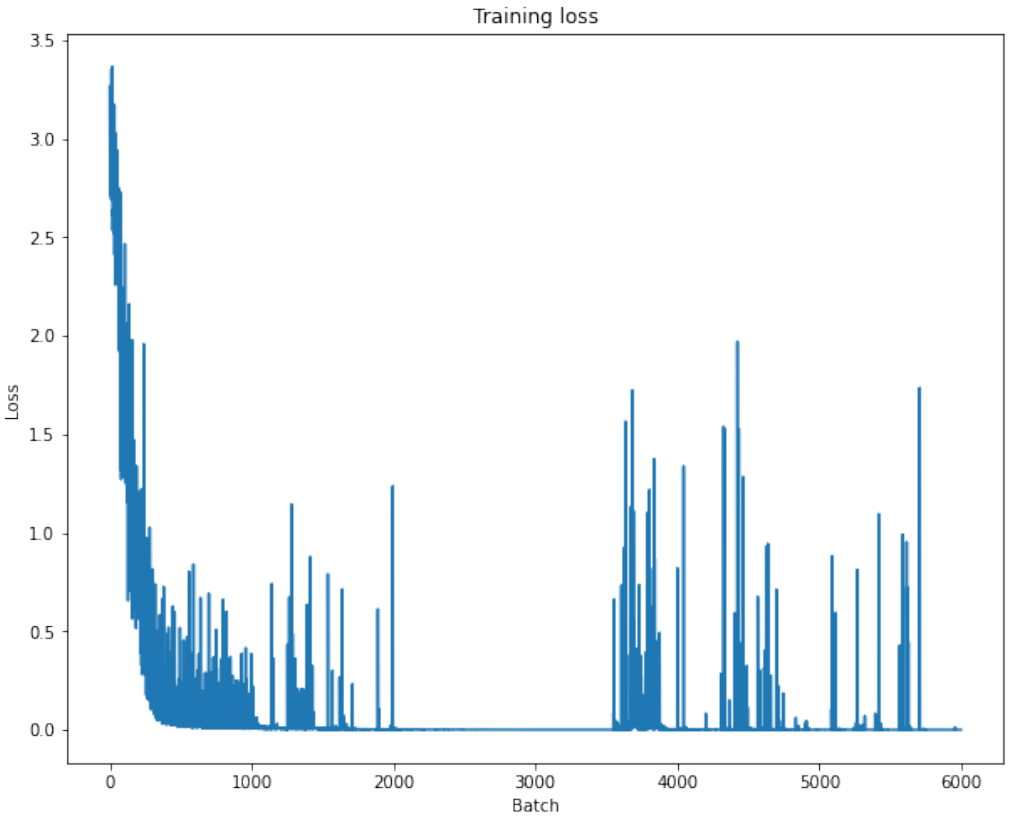
První scénář

V prvním scénáři bylo přistoupeno k vytvoření nejobjemnější trénovací datové sadě (60 %) a menší validační a testovací datové sadě (20 % a 20 %). Tento scénář byl časově nejvíc náročný z důvodu objemné trénovací datové sady. Výsledky byly následující:

- 1. epocha:
  - Čas epochy – 32 minut, 6 sekund;
  - Ztráta při trénování (*train loss*) – 0,1797;
  - Validační přesnost (*validation accuracy*) – 1,0000.
- 2. epocha:
  - Čas epochy – 31 minut, 56 sekund;
  - Ztráta při trénování (*train loss*) – 0,0250;
  - Validační přesnost (*validation accuracy*) – 1,0000.

V tomto případě byla v obou epochách validační přesnost 1.0000, což bylo způsobeno příliš velkou datovou sadou pro trénování. Na obr. 4.1 lze vidět graf průběhu ztrátové funkce při trénování modelu. Zpočátku graf vykazuje dobré výsledky, ale ke konci je možné sledovat náhlé výkyvy. Tohle chování může být způsobeno loss funkcí blížící se k nule v rozmezí dávek zhruba 2 000-3 500. Model stále aktualizuje své parametry a to paradoxně vede ke zhoršení výsledků. Stále se ale jedná o nejmenší hodnotu loss funkce v rámci všech tří scénářů. V testovací fázi modelu byly všechny

predikce správné (přesnost = 1.0000) a tím bylo dosáhnuto nejlepšího výsledku ze zmíněných scénářů.



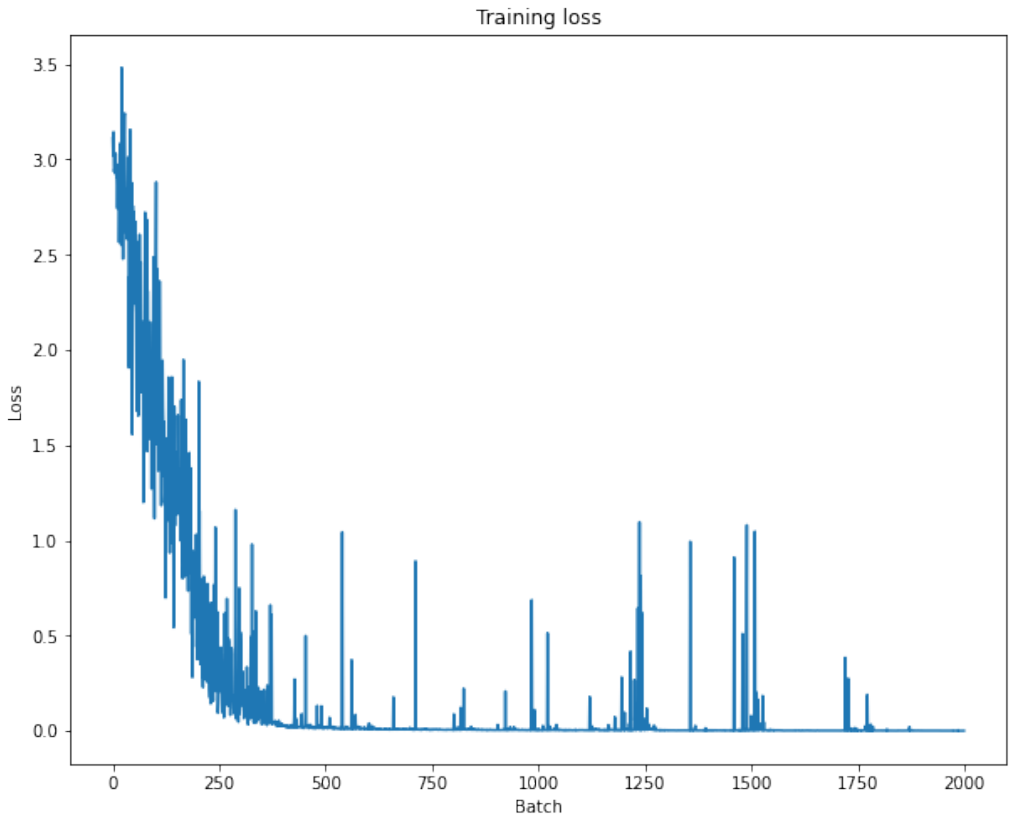
Obr. 4.1: Graf znázorňující průběh `Loss` funkce v poměru ke zpracovaným `batches` prvního scénáře

Druhý scénář

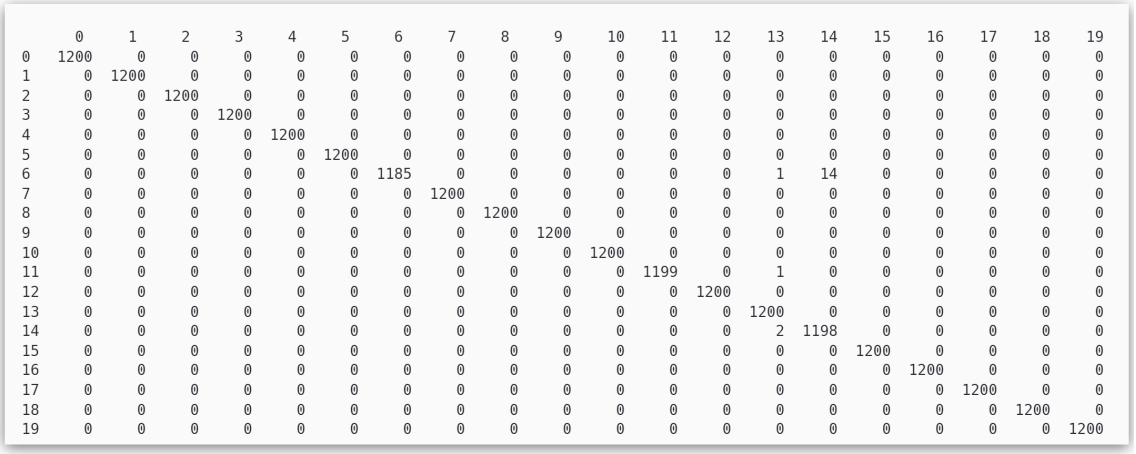
V rámci druhého scénáře byly pozměněny poměry datových sad na 20 % trénovací datová sada, 20 % validační datová sada a 60 % testovací datová sada. Tento scénář byl nejlepším z hlediska krátkého času nutného ke trénování a validaci a přijatelnými výsledky při trénování, validaci i testování:

- 1. epocha:
  - Čas epochy – 11 minut, 28 sekund;
  - Ztráta při trénování (*train loss*) – 0,4285;
  - Validační přesnost (*validation accuracy*) – 0,9915.
- 2. epocha:
  - Čas epochy – 11 minut, 29 sekund;
  - Ztráta při trénování (*train loss*) – 0,0163;
  - Validační přesnost (*validation accuracy*) – 0,9898.

Oproti prvnímu scénáři lze sledovat mírné navýšení ztráty při trénování v první epoše, ale markantní zlepšení v druhé epoše. Graf tohoto modelu viz obr.4.2. Z grafu lze vyčíst postupné snižování loss funkce, což může být způsobeno tím, že se model nedostal do fáze, kdy se po delší dobu loss funkce blíží nule a nedošlo tedy k větším výkyvům, jako v případě prvního scénáře. Při testování tohoto modelu bylo také dosaženo dobrých výsledků s 23 982 správnými predikcemi a pouze 18 špatnými. Na obr. 4.3 lze vidět predikce modelu na ose X a skutečné hodnoty na ose Y. S přihlédnutím k výsledkům testování modelu byla výsledná přesnost testování 0,99925.



Obr. 4.2: Graf znázorňující průběh `Loss` funkce v poměru ke zpracovaným `batches` druhého scénáře



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1185	0	0	0	0	0	0	1	14	0	0	0	0	0
7	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1199	0	1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1198	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1200	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1200	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1200	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1200	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1200

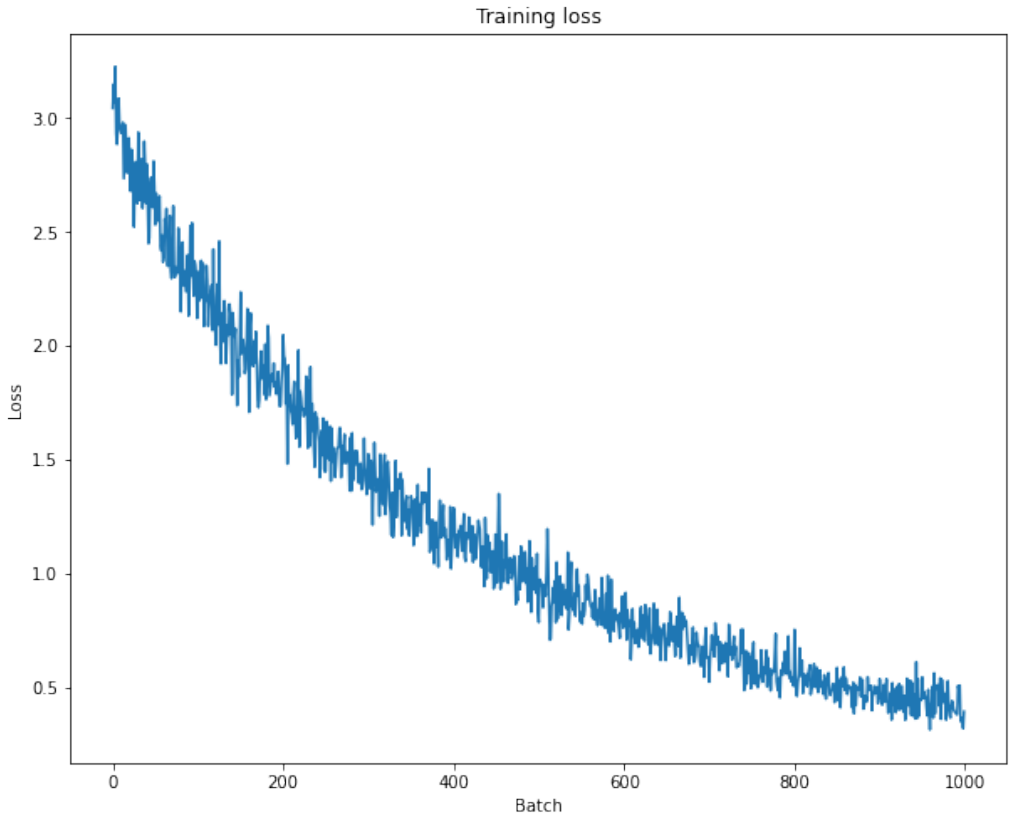
Obr. 4.3: Předpovězené hodnoty modelem (osa X) a reálné hodnoty (osa Y) pro druhý scénář

Třetí scénář

Ve posledním scénáři byl největší rozdíl mezi jednotlivými datovými sadami – trénovací datová sada 10 %, validační datová sada 10 % a testovací datová sada 80 %. Tohle řešení bylo ze všech nejrychlejší k testování a validaci. Tohle řešení s sebou sice přineslo mnohem horší výsledky, co se týče trénování, ale na druhé straně lepší výsledky při validaci:

- 1. epocha:
  - Čas epochy – 9 minut, 36 sekund;
  - Ztráta při trénování (*train loss*) – 1,7162;
  - Validační přesnost (*validation accuracy*) – 0,9995.
- 2. epocha:
  - Čas epochy – 9 minut, 38 sekund;
  - Ztráta při trénování (*train loss*) – 0,6301;
  - Validační přesnost (*validation accuracy*) – 0,9995.

Pokud se ovšem přihlédne ke grafu znázorňujícímu ztrátu při trénování modelu (obr. 4.4), lze vidět sestupný trend, který nejeví tak velké změny jak u předchozích scénářů. Jakmile se ovšem model dostal k testování, byly jak výsledky, tak časová náročnost o dost horší (viz obr. 4.5). Správných předpovědí zde bylo 30 632 a těch špatných 1 1368, což ve výsledku dává přesnost modelu 0,95725. Nutno podotknout, že hlavním důvodem pro tak velký počet špatných předpovědí je predikce 1 344 sekvencí dat patřících do třídy 5, ale model určil třídu 6. Tento jev je nejspíše způsoben nedostatečně objemnou trénovací datovou sadou a také tím, že se jedná o Linux a MacOS logy, které jsou si podobné.



Obr. 4.4: Graf znázorňující průběh `Loss` funkce v poměru ke zpracovaným `batches` třetího scénáře

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1576	0	0	0	22	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
1	0	1600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	256	1344	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1600	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1600	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1600	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1600	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1600	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1600	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1600	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1600	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1600	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1600	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1600	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1600	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1600	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1600

Obr. 4.5: Předpovězené hodnoty modelem (osa X) a reálné hodnoty (osa Y) pro třetí scénář

Shrnutí

V provedených třech scénářích s různými vlastnostmi byla vyhodnocena účinnost modelu Google BERT na úlohu klasifikace sekvencí dat. Všechny výsledky, co se týče přesnosti modelu při testování, vyšly nad 0,95. Tento výsledek by se dal považovat za *state-of-the-art*. Příliš dobré výsledky mohou být způsobeny tím, že model měl k dispozici pouze omezené množství dat (různých zdrojů). Shrnutí výsledků trénování a validace všech scénářů je možné vidět v tab. 4.1.

Tab. 4.1: Shrnutí učení modelu BERT v jednotlivých scénářích

Epocha	1			2		
Scénář	První	Druhý	Třetí	První	Druhý	Třetí
Čas trénování a validace	32 min 6 sek	11 min 28 sek	9 min 36 sek	31min 56 sek	11 min 29 sek	9 min 38 sek
Ztráta při trénování	0,1797	0,4285	1,7162	0,0250	0,0163	0,6301
Validační přesnost	1,0000	0,9915	0,9995	1,0000	0,9898	0,9995

V tabulce jde vidět, že v procesu učení modelu BERT jasně převažuje první scénář, kde byla trénovací datová sada nejrozsáhlejší. Ovšem pokud se vezme v potaz čas trénování a validace modelu, má nejlepší výsledky druhý scénář. Třetí scénář byl zahrnut pro otestování, zda model dokáže spolehlivě vykonávat úkol klasifikace sekvencí textu i v případě, že trénovací a validační datová sada tvoří pouze zlomek celku. Stále se ale jedná o dobrý výsledek (shrnutí výsledků testování viz tab. 4.2), který dokazuje, proč je model Google BERT účinným nástrojem v klasifikaci textu.

Tab. 4.2: Shrnutí testování modelu BERT v jednotlivých scénářích

Scénář	První	Druhý	Třetí
Správné předpovědi	8 000	23 982	30 632
Chybné předpovědi	0	18	1 344
Přesnost modelu	1,0000	0,9993	0,9573

Závěr

V teoretické části diplomové práce byly rozebrány základy bezpečnostního monitoringu v oblasti kybernetické bezpečnosti (kapitola 1.1). Tento rozbor obsahoval důležité poznatky o metodách bezpečnostního monitoringu a o důležitosti managementu bezpečnostních informací a událostí SIEM a o jejich možné kombinaci se systémy orchestrace zabezpečení, automatizace a odezvy SOAR. V návaznosti na bezpečnostní monitoring také byla stručně popsána analýza datových toků v síti (kapitola 1.2) a správa logů událostí(kapitola 1.3). V rámci bezpečnostního monitoringu byl probrán jeho účel, nástroje a způsoby, pomocí kterých lze realizovat a jeho nedostatky se zaměřením na nedostatek automatizace. Byla také zmíněna kybernetická operační centra SOC a jejich nezbytnost k zajištění bezpečnějšího kybernetického prostoru spolu s jejich závislostí na moderních prostředcích bezpečnostního monitoringu, a to SIEM a SOAR (kapitoly 1.4 a 1.6). Jelikož je diplomová práce především zaměřena na vylepšení stávajících systémů SIEM, byly vytyčeny jejich hlavní nedostatky v kapitole 1.5. Následně byla probrána základní teorie týkající se problematiky umělé inteligence a jejím odvětvím – tedy strojové učení (kapitola 2.1), neuronové sítě a hluboké učení (kapitola 2.2) a zpracování přirozeného jazyka (kapitola 2.3). Byly popsány základní přístupy strojového učení (učení s učitelem, učení bez učitele a zpětnovazební učení). V rámci neuronových sítí a hlubokého učení byly probrány základní typy architektur neuronových sítí (podkapitoly 2.2.1, 2.2.2 a 2.2.3) spolu známými modely, které jim přísluší – ULMFiT a Google BERT. Nutno podotknout, že v kapitole o hlubokém učení byly také zahrnuty základní pojmy, jako např. epocha, vzorek, dávka, váha, rychlost učení atd. V oblasti neuronových sítí je věnovaná pozornost především tzv. modelům přeneseného učení (*transfer learning*), které umožňují využití komplexních modelů neuronových sítí bez nutnosti jejich trénování od začátku, což by vyžadovalo velké množství dat a času. Velice důležitou částí je zde kapitola 2.4, kde jsou probrána možná vylepšení různých aspektů bezpečnostního monitoringu pomocí umělé inteligence. byla provedena rešerše na již existující řešení pro usnadnění práce operátora – DeepLog, DeepCASE a Tiresias. Tahle řešení jsou velice ambiciózní a u prvních dvou existuje i podrobný návod na jejich využití (v případě DeepCASe ovšem chybí hlavní datová sada). Navzdory těmto nedostatkům mohou ovšem tyto řešení sloužit jako dobrý podklad pro porozumění tomu, jak by se dal bezpečnostní monitoring vylepšit pomocí technik umělé inteligence. Závěrem kapitoly 2.4 je uvedena aplikace různých nástrojů strojového učení na případy užití týkající se bezpečnostního monitoringu. Díky této první části diplomové práce byly zpracovány důležité informace, které sloužily jako podklad pro zpracování praktické aplikace modelu neuronové sítě pro zlepšení práce uživatele platformy v rámci kybernetické bezpečnosti.

Ve třetí části diplomové práce bylo nastíněno možné vylepšení procesu bezpečnostního monitoringu pomocí klasifikace sekvence dat (logů událostí). Myšlenka byla taková, že při monitorování velké organizace pomocí kybernetického operačního centra může být těžké vyznat se ve všech kategoriích logů událostí a bylo by tedy vhodné automatizovat proces kategorizace všech zdrojů logů událostí (v případě větší organizace se může jednat i o stovky kategorií LES). Pokud by bylo možné takový proces automatizovat pomocí umělé inteligence, ušetřilo by to analytikům SOC velké množství času a námahy. První kapitolou v rámci praktické části byla charakteristika potřebných dat (kapitola 3.1). Zde je popsán proces shromažďování všech dat logů událostí, která byla sbírána výhradně v surové (*raw*) formě. Tato forma umožňuje s logy jednoduše pracovat, jelikož se jedná o jedinou sekvenci dat o určité události bez jakéhokoliv rozdělení (parsování). Bylo vytyčeno a nashromážděno 20 různých zdrojů LES ve formátech *.log* a *.txt*, kde každý soubor obsahuje data o 2 000 událostech. Dále, v kapitole 3.2 je uveden způsob, jakým byla data zpracována do datové sady (tedy do podoby, se kterou bude model neuronové sítě moci dále pracovat). Tohle zpracování zahrnovalo převedení na jednotný formát *.csv*, přidělení hlavičky *labels* a *log*, kde *labels* jsou kategorie podle jednotlivých zdrojů a *log* je samotná událost. V rámci zpracování dat byly vytvořeny tři základní datové sady: trénovací datová sada (24 000 entit), validační datová sada (8 000 entit) a testovací datová sada (8 000 entit). V následující kapitole 3.3 byla uskutečněna volba modelu neuronové sítě. Pro úlohu klasifikace sekvencí dat byl zvolen model BERT Base Uncased (`BertForSequenceClassification`). Tento model přeneseného učení je ideální z hlediska jeho výkonu a úspory času. V rámci této kapitoly byla také popsána knihovna PyTorch pro strojové učení, která obsahuje předtrénované modely architektury Transformers. V kapitole 3.4 je zahrnut technický popis modelu BERT. Pro pochopení, jak model funguje, byly popsány jeho různé varianty, speciální tokeny ([`CLS`], [`SEP`] a [`MASK`]) a v obr. 3.4 byl znázorněn proces jeho předtrénování, které využívá predikce následující věty NSP a maskované modelování jazyka MLM. Poslední v třetí části diplomové práce, kapitola 3.5 popisuje konečné zpracování dat před samotným procesem učení modelu. Konečné zpracování dat zahrnuje načtení datových sad k trénování, validaci a testování modelu BERT, následné rozdělení těchto dat na tokeny pomocí tokenizéru `BertTokenizer`, převedení dat na tenzory a nakonec sjednocení těchto tenzorů pomocí knihovny torch.utils.dat do konečné datové sady, která je následně předána na vstup modelu. Na konec třetí části diplomové práce byly popsány tři různé scénáře klasifikace sekvencí dat pomocí modelu BERT. Tyto scénáře se liší především v poměru rozdělení jednotlivých datových sad (první scénář – 60:20:20, druhý scénář – 20:20:60 a třetí scénář 10:10:80 –> trénovací:validační:testovací datová sada), ale také jsou odlišné rychlosti učení ($4e-5$ pro první scénář a $2e-6$ pro druhý a třetí scénář).

Čtvrtou a poslední částí diplomové práce bylo vyhodnocení výsledků měření. V rámci této byly popsány všechny tři provedené scénáře a bylo provedeno jejich zhodnocení podle toho, jaké měly výsledky. Nejefektivnějším z hlediska přesnosti trénování, validace a testování modelu byl první scénář, který dosáhl validační a testovací přesnosti 1,0000, zatímco ztráta při trénování byla pouze 0,1024. Tento scénář ovšem také zabral nejvíce času pro jeho trénování a validaci, celkem 1 hodinu a 2 minuty. Druhý scénář zahrnoval přijatelné výsledky (ztráta při trénování 0,2224; validační přesnost 0,9907 a testovací přesnost 0,9993) za zlomek času – 23 minut na trénování a validaci. Třetí scénář, jelikož zahrnoval nejmenší trénovací a validační datovou sadu, vykazoval nejhorší výsledky (ztráta při trénování 1,1732; validační přesnost 0,9995 a testovací přesnost 0,9573). Trénování a validace tohoto scénáře ovšem zabrala pouze 19 minut. Pokud by se bral v potaz poměr výkonu k ušetřenému času, nejlepší výsledky vykazoval model z druhého scénáře. Každopádně těmito experimentálními aplikacemi modelu BERT, na rozdělování logů událostí do kategorií, bylo dosaženo *state-of-the-art* výsledků. Navzdory tomu, že druhý a třetí scénář disponovaly menší množinou trénovacích a validačních dat byly výsledky stále přijatelné.

Na tuto diplomovou práci by se zajisté dalo navázat, např. naučením použitého modelu na datové sadě, která by obsahovala 100 a více kategorií logů událostí, aby se zjistilo, zda model BERT zvládne klasifikovat i takové množství dat. Zároveň by se model (menší úpravou kódu) dal využít k binární klasifikaci relevance, závažnosti logů událostí, detekce normálního a abnormálního chování v síti atd. Také by se vytvořené řešení dalo využít jako experiment v rámci kybernetického operačního centra ke zjištění, zda by šlo využít v reálném provozu SOC.

Literatura

[1] JIRÁSEK, Petr, Luděk NOVÁK a Josef POŽÁR. *Výkladový slovník kybernetické bezpečnosti: Cyber security glossary*. 2., aktualiz. vyd. Praha: Policejní akademie ČR v Praze, 2013. ISBN 978–80–7251–397–0.

[2] TUNAY, John. *Why an antivirus is not enough*. [online]. [cit. 27. 10. 2021]. Dostupné z URL: <https://www.secureage.com/article-antivirus-protection-is-not-enough>.

[3] *What is cyber monitoring?* [online]. [cit. 27. 10. 2021]. Dostupné z URL: <https://sdi.ai/blog/what-is-cyber-monitoring/>.

[4] *Outsourced SOC vs. internal SOC: how to choose?* [online]. [cit. 29. 10. 2021]. Dostupné z URL: <https://www.linkbynet.com/en/news/outsourced-soc-vs-internal-soc-how-to-choose>.

[5] Cisco Press. *Overview of Security Operations Center Technologies*. [online]. [cit. 30. 4. 2022]. Dostupné z URL: <https://www.ciscopress.com/articles/article.asp?p=2455014&seqNum=5>.

[6] HOFSTEDE, Rick, Pavel ČELEDA, Brian TRAMMELL, Idilio DRAGO, Ramin SADRE, Anna SPEROTTO a Aiko PRAS. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials*. [online]. 2014, 16(4), 2037-2064 [cit. 29. 10. 2021]. ISSN 1553-877X. Dostupné z URL: <https://ieeexplore.ieee.org/abstract/document/6814316>.

[7] SOUPPAYA, Murugiah a Karen KENT. *Guide to Computer Security Log Management*. [online]. Gaithersburg: National Institute of Standards and Technology. [cit. 29. 10. 2021]. ISBN 978-1548204815. Dostupné z URL: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-92.pdf>.

[8] *The 5 steps of log management*. [online]. [cit. 30. 10. 2021]. Dostupné z URL: <https://www.humio.com/whats-new/blog/the-5-steps-of-log-management/>.

[9] *Log formats – a complete guide*. [online]. [cit. 30. 10. 2021]. Dostupné z URL: <https://www.graylog.org/post/log-formats-a-complete-guide>.

[10] GERHARDS, Rainer. *The Syslog Protocol: RFC 5424*. [online]. [cit. 30. 10. 2021]. Dostupné z URL: <https://datatracker.ietf.org/doc/html/rfc5424>.

[11] *ArcSight Common Event Format*. [online]. [cit. 31. 10. 2021]. Dostupné z URL: <https://nxlog.co/documentation/nxlog-user-guide/cef.html>.

[12] *Mapování polí CEF a CommonSecurityLog*. [online]. [cit. 31. 10. 2021]. Dostupné z URL: <https://docs.microsoft.com/cs-cz/azure/sentinel/cef-name-mapping>.

[13] *Log File Formats*. [online]. [cit. 31. 10. 2021]. Dostupné z URL: <http://publib.boulder.ibm.com/tividd/td/ITWSA/ITWSA_info45/en_US/HTML/guide/c-logs.html>.

[14] AHOLA, Micke. *The Role of Human Error in Successful Cyber Security Breaches*. [online]. [cit. 3. 11. 2021]. Dostupné z URL: <https://blog.usecure.io/the-role-of-human-error-in-successful-cyber-security-breaches>.

[15] CONSTANTINE, Conrad. *What to log in a SIEM: SIEM and security logging best practices explained*. [online]. [cit. 3. 11. 2021]. Dostupné z URL: <https://cybersecurity.att.com/blogs/security-essentials/what-kind-of-logs-for-effective-siem-implementation>.

[16] LIAO, Hung-Jen, Chun-Hung Richard LIN a Ying-Chin LIN. *Intrusion detection system: A comprehensive review*. [online]. Journal of Network and Computer Applications, 2013. [cit. 3. 11. 2021]. ISSN 1084-8045. Dostupné z URL: <http://dl-maghaleh.ir/wp-content/uploads/2016/03/order-z-1426675381-750.pdf>.

[17] *The Lifecycle of a Security Event*. [online]. [cit. 4. 11. 2021]. Dostupné z URL: <https://www.coresecurity.com/blog/lifecycle-security-event>.

[18] RAI, Jatin. *ArcSight – A better insight security solution*. [online]. [cit. 20. 3. 2022]. Dostupné z URL: <https://www.trustradius.com/reviews/arcsight-enterprise-security-manager-formerly-hp-arcsight-2019-12-09-13-03-18>.

[19] *RSA Netwitness Evolved SIEM*. [online]. [cit. 20. 3. 2022]. Dostupné z URL: <https://www.tokenguard.com/datasheets/rsa-netwitness-evolved-siem.pdf>.

[20] *IBM QRadar Security Information Event Management platform (SIEM)*. [online]. [cit. 20. 3. 2022]. Dostupné z URL: <https://www.infoguard.ch/en/partners/ibm-qradar-security-information-event-management-siem>.

[21] *Elastic SIEM is free and open for security analysts everywhere.* [online]. [cit. 21. 3. 2022]. Dostupné z URL: <https://www.elastic.co/blog/elastic-siem-free-open>.

[22] *Splunk Enterprise Security product brief.* [online]. [cit. 21. 3. 2022]. Dostupné z URL: <https://www.splunk.com/pdfs/product-briefs/splunk-enterprise-security.pdf>.

[23] CISCO. *Anticipating the Unknowns - Chief Information Security Officer (CISO) Benchmark Study.* Technical report, 2019. [online]. [cit. 24. 4. 2022]. Dostupné z URL: <https://ebooks.cisco.com/story/anticipating-unknowns/page/6/1>.

[24] GONZÁLES-GRANADILLO, Gustavo, Susana GONZÁLES-ZARZOSA a Rodrigo DIAZ. *Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures.* [online]. Madrid: Cybersecurity Unit, Atos Research & Innovation, 2021 [cit. 4. 11. 2021]. ISSN 1424-8220. Dostupné z URL: <https://www.mdpi.com/1424-8220/21/14/4759>.

[25] KOSŤ, Dávid. *SOAR: Security Orchestration, Automation and Response.* [online]. [cit. 27. 3. 2022]. Dostupné z URL: <https://www.itsec-nn.com/soar-security-orchestration-automation-and-response/>.

[26] SHEA, Sharon. *SOAR (security orchestration, automation and response).* [online]. [cit. 29. 3. 2022]. Dostupné z URL: <https://www.techtarget.com/searchsecurity/definition/SOAR>.

[27] *What is a SOAR Playbook?.* [online]. [cit. 30. 3. 2022]. Dostupné z URL: <https://cyware.com/educational-guides/security-orchestration-automation-and-response/what-is-a-soar-playbook-dcad>.

[28] SAFONOV, Yehor. Filtrování spamových zpráv pomocí metod umělé inteligence. Brno, 2019, 150 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Martin Kolařík

[29] *Machine Learning.* [online]. [cit. 21. 3. 2022]. Dostupné z URL: <https://www.sas.com/en_us/insights/analytics/machine-learning.html>.

[30] DELUA, Julianna. *Supervised vs. Unsupervised Learning: What's the Difference?.* [online]. [cit. 24. 3. 2022]. Dostupné z URL: <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>.

[31] CAREW, Joseph. *Reinforcement learning*. [online]. [cit. 24. 3. 2022]. Dostupné z URL: <https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning>.

[32] KRIZHEVSKY, Alex. *The CIFAR-10 dataset*. [online]. [cit. 4. 11. 2021]. Dostupné z URL: <https://www.cs.toronto.edu/~kriz/cifar.html>.

[33] KRIZHEVSKY, Alex. *Training Data vs. Validation Data vs. Test Data for ML Algorithms*. [online]. [cit. 4. 11. 2021]. Dostupné z URL: <https://www.applause.com/blog/training-data-validation-data-vs-test-data>.

[34] GUPTA, Ayush. *A Comprehensive Guide on Deep Learning Optimizers*. [online]. [cit. 11. 4. 2022]. Dostupné z URL: <https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/>.

[35] HS13. *Deep Learning 101: Beginners Guide to Neural Network*. [online]. [cit. 27. 3. 2022]. Dostupné z URL: <https://www.analyticsvidhya.com/blog/2021/03/basics-of-neural-network/>.

[36] HARDESTY, Larry. *Explained: Neural networks*. [online]. [cit. 5. 11. 2021]. Dostupné z URL: <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>.

[37] *What Is Deep Learning?*. [online]. [cit. 6. 11. 2021]. Dostupné z URL: <https://www.baslerweb.com/en/vision-campus/markets-and-applications/what-is-deep-learning/>.

[38] *Deep Learning*. [online]. [cit. 6. 11. 2021]. Dostupné z URL: <https://www.ibm.com/cloud/learn/deep-learning>.

[39] BURNS, Ed a Kate BRUSH. *Deep learning neural network*. [online]. [cit. 6. 11. 2021]. Dostupné z URL: <https://searchenterpriseai.techtarget.com/definition/deep-learning-deep-neural-network>.

[40] CHOLLET, François. *Deep learning v jazyku Python: Knihovny Keras, TensorFlow*. Přeložil Rudolf PECINOVSKÝ. Praha: Grada Publishing, 2019. Knihovna programátora (Grada). ISBN 978-80-247-3100-1.

[41] JORDAN, Jeremy. *Evaluating a machine learning model*. [online]. [cit. 25. 4. 2022]. Dostupné z URL: <https://www.jeremyjordan.me/evaluating-a-machine-learning-model/>.

[42] ČERMÁK, Miroslav. *Co vyplývá z většího množství false positive a false negative událostí*. [online]. [cit. 25. 4. 2022]. Dostupné z URL: <https://www.cleverandsmart.cz/co-vyplyva-z-vetsiho-mnozstvi-false-positive-a-false-negative-udalosti/>.

[43] KORSTANJSE, Joos. *The F1 score*. [online]. [cit. 25. 4. 2022]. Dostupné z URL: <https://towardsdatascience.com/the-f1-score-bec2bbc38aa6>.

[44] OLAH, Christopher. *Understanding LSTM Networks*. [online]. [cit. 10. 11. 2021]. Dostupné z URL: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

[45] PHI, Michael. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. [online]. [cit. 10. 11. 2021]. Dostupné z URL: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>.

[46] SAHA, Sumit. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [online]. [cit. 11. 11. 2021]. Dostupné z URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

[47] DAR, Pranav. *8 Excellent Pretrained Models to get you Started with Natural Language Processing (NLP)*. [online]. [cit. 11. 11. 2021]. Dostupné z URL: <https://www.analyticsvidhya.com/blog/2019/03/pretrained-models-get-started-nlp/>.

[48] RUDD, Ethan a Ahmed ABDALLAH. *Training Transformers for Information Security Tasks: A Case Study on Malicious URL Prediction*. [online]. [cit. 12. 11. 2021]. Dostupné z URL: <https://arxiv.org/pdf/2011.03040.pdf>.

[49] JOSHI, Prateek. *How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models*. [online]. [cit. 12. 11. 2021]. Dostupné z URL: <https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/>.

[50] *Natural Language Processing (NLP)*. [online]. [cit. 13. 11. 2021]. Dostupné z URL: <https://www.sas.com/en_us/insights/analytics/what-is-natural-language-processing-nlp.html>.

[51] JOSHI, Prateek. *Tutorial on Text Classification (NLP) using ULMFiT and fastai Library in Python*. [online]. [cit. 13. 11. 2021]. Dostupné z URL: <https://www.analyticsvidhya.com/blog/2018/11/tutorial-text-classification-ulmfit-fastai-library/>.

[52] HOWARD, Jeremy a Sebastian RUDER. *Universal Language Model Fine-tuning for Text Classification*. [online]. [cit. 14. 11. 2021]. Dostupné z URL: <https://arxiv.org/pdf/1801.06146v5.pdf>.

[53] *Using artificial intelligence and machine learning to detect abnormal behavior*. [online]. [cit. 19. 11. 2021]. Dostupné z URL: <https://www.manageengine.com/log-management/siem/ueba-machine-learning-ai.html>.

[54] RIVAS, Genesis. *AI and SIEM: Increase the efficiency of your IT Security Team*. [online]. [cit. 19. 11. 2021]. Dostupné z URL: <https://www.gb-advisors.com/ai-and-siem/>.

[55] DU, Min, Feifei LI, Guineng ZHENG a Vivek SRIKUMAR. *DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning*. [online]. [cit. 21. 4. 2022]. Dostupné z URL: <https://www.cs.utah.edu/~lifeifei/papers/deeplog.pdf>.

[56] VAN EDE, Thijs. *DeepLog*. [online]. [cit. 21. 4. 2022]. Dostupné z URL: <https://github.com/Thijsvanede/DeepLog>.

[57] DU, Min a Feifei LI. *Spell: Streaming Parsing of System Event Logs*. [online]. [cit. 22. 4. 2022]. Dostupné z URL: <https://www.cs.utah.edu/~lifeifei/papers/spell.pdf>.

[58] VAN EDE, Thijs. *DeepLog's documentation*. [online]. [cit. 22. 4. 2022]. Dostupné z URL: <https://deeplog.readthedocs.io/en/latest/index.html>.

[59] VAN EDE, Thijs, Hojjat AGHAKHANI , Noah SPAHN , Riccardo BORTOLAMEOTTI , Marco COVA, Andrea CONTINELLA, Maarten VAN STEEN, Andreas PETER, Christopher KRUEGEL a Giovanni VIGNA. *DEEPCASE: Semi-Supervised Contextual Analysis of Security Events*. [online]. [cit. 23. 4. 2022]. Dostupné z URL: <https://vm-thijs.ewi.utwente.nl/static/homepage/papers/deepcase.pdf>.

[60] VAN EDE, Thijs. *DeepCASE*. [online]. [cit. 25. 4. 2022]. Dostupné z URL: <https://github.com/Thijsvanede/DeepCASE/>.

[61] VAN EDE, Thijs. *DeepCASE's documentation*. [online]. [cit. 26. 4. 2022]. Dostupné z URL: <https://deepcase.readthedocs.io/en/latest/index.html>.

[62] SHEN, Yun, Enrico MARICONTI, Pierre-Antoine VERVIER a Gianluca STRINGHINI. *Tiresias: Predicting Security Events Through Deep Learning*. [online]. [cit. 26. 4. 2022]. Dostupné z URL: <https://arxiv.org/pdf/1905.10328.pdf>.

[63] LOGPAI. *Loghub*. [online]. [cit. 30. 4. 2022]. Dostupné z URL: <https://github.com/logpai/loghub>.

[64] SCONZO, Mike. *SecRepo – Samples of Security Related Data*. [online]. [cit. 30. 4. 2022]. Dostupné z URL: <https://www.secrepo.com/>.

[65] *ThreatFox – Share Indicators Of Compromise (IOCs)*. [online]. [cit. 30. 4. 2022]. Dostupné z URL: <https://threatfox.abuse.ch/>.

[66] TCRUG *Marxdata*. [online]. [cit. 30. 4. 2022]. Dostupné z URL: <https://github.com/tcrug/marx_data>.

[67] *Free threat intelligence feeds*. [online]. [cit. 30. 4. 2022]. Dostupné z URL: <https://threatfeeds.io/>.

[68] COHEN, Ori. *BERT Fine-Tuning Sentence Classification*. [online]. [cit. 12. 12. 2021]. Dostupné z URL: <https://colab.research.google.com/github/orico/NLP/blob/master/BERT_Fine_Tuning_Sentence_Classification.ipynb>.

[69] SIMPLILEARN *What is PyTorch, and How Does It Work: All You Need to Know*. [online]. [cit. 3. 5. 2022]. Dostupné z URL: <https://www.simplilearn.com/what-is-pytorch-article>.

[70] HUGGINGFACE. *PyTorch Transformers*. [online]. [cit. 3. 5. 2022]. Dostupné z URL: <https://pytorch.org/hub/huggingface_pytorch-transformers/>.

[71] RAJASEKHARAN, Ajit. *What is a masked language model, and how is it related to BERT?* [online]. [cit. 12. 12. 2021]. Dostupné z URL: <https://www.quora.com/What-is-a-masked-language-model-and-how-is-it-related-to-BERT>.

[72] POGIATZIS, Andreas. *NLP: Contextualized word embeddings from BERT*. [online]. [cit. 13. 12. 2021]. Dostupné z URL: <https://towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b>.

[73] JAISWAL, Abhishek. *Manual for the First Time Users: Google BERT for Text Classification*. [online]. [cit. 1. 5. 2022]. Dostupné z URL: <https://www.analyticsvidhya.com/blog/2021/12/manual-for-the-first-time-users-google-bert-for-text-classification/>.

[74] DWIVEDI, Harshit. *How to Use Google Colab for Deep Learning*. [online]. [cit. 4. 5. 2022]. Dostupné z URL: <https://neptune.ai/blog/how-to-use-google-colab-for-deep-learning-complete-tutorial>.

[75] HUGGINGFACE. *BERT*. [online]. [cit. 4. 5. 2022]. Dostupné z URL: <https://huggingface.co/docs/transformers/model_doc/bert>.

[76] YE, Deming, Yankai LIN, Yufei HUANG, Maosong SUN. *TR-BERT: Dynamic Token Reduction for Accelerating BERT Inference*. [online]. [cit. 4. 5. 2022]. Dostupné z URL: <https://aclanthology.org/2021.naacl-main.463.pdf>.

[77] KAN, Enoch. *Quick ML Concepts: Tensors*. [online]. [cit. 4. 5. 2022]. Dostupné z URL: <https://towardsdatascience.com/quick-ml-concepts-tensors-eb1330d7760f>.

[78] *TORCH.UTILS.DATA*. [online]. [cit. 5. 5. 2022]. Dostupné z URL: <https://pytorch.org/docs/stable/data.html>.

[79] MCCORMICK, Chris and Nick RYAN. *BERT Fine-Tuning Tutorial with PyTorch*. [online]. [cit. 10. 5. 2022]. Dostupné z URL: <https://colab.research.google.com/drive/1Wd8pQDaSwLgyHsHF9UjN7-fP93cfPOFI?usp=sharing#scrollTo=EKOTlwcmxmej>.

[80] *BERT Fine-Tuning Sentence Classification*. [online]. [cit. 10. 5. 2022]. Dostupné z URL: <https://colab.research.google.com/drive/1ywsvwO6thOVOrfagjjfuxEf6xVRxbUNO#scrollTo=BJR6t_gCQe_x>.

[81] RAJAPAKSE, Thilina. *PyTorch Transformers Classification*. [online]. [cit. 10. 5. 2022]. Dostupné z URL: <https://github.com/ThilinaRajapakse/pytorch-transformers-classification>.

Seznam symbolů, veličin a zkratek

AI	Artificial Intelligence – Umělá inteligence
ANN	Artificial Neural Network – Umělá neuronová síť
APT	Advanced Persistent Threat – Pokročilá trvalá hrozba
BERT	Bidirectional Encoder Representations from Transformers
CEF	Common Event Format – Druh formátu logů událostí
CLF	Common Log Format – Druh formátu logů událostí
CNN	Convolutional Neural Network – Konvoluční neuronová síť
CVE	Common Vulnerabilities and Exposures – Databáze zranitelností
Discr	Discriminative fine-tuning – Strategie ladění
DLP	Data Loss Prevention – Ochrana před ztrátou dat
DNS	Domain Name System – Systém doménových jmen
EDR	Endpoint Detection and Response – Nástroj pro detekci a reakci na útoky na koncová zařízení
FTP	File Transfer Protocol – Protokol pro přenos souborů
GDPR	General Data Protection Regulation – Nařízení Evropské unie
IDS	Intrusion Detection System – Systém pro detekci podezřelé aktivity
IP	Internet Protocol – Protokol síťové vrstvy
IPS	Intrusion Prevention System – Systém pro detekci a prevenci podezřelé aktivity
JSON	JavaScript Object Notation – Druh formátu logů událostí
LES	Log Event Source – Zdroj logů událostí
LSTM	Long Short Term Memory – Druh architektury RNN
ML	Machine Learning – Strojové učení
MTTD	Mean Time To Detect – Průměrná doba detekce incidentu
MTTR	Mean Time To Respond – Průměrná doba reakce na incident

NLP	Natural Language Processing – Zpracování přirozeného jazyka
NLU	Natural Language Understanding – Porozumění přirozenému jazyku
NN	Neural Network – Neuronová síť
RNN	Recurrent Neural Network – Rekurentní neuronová síť
SIEM	Security Information and Event Management – Management bezpečnostních informací a událostí
SOAR	Security Orchestration, Automation and Response – Orchestrace zabezpečení, automatizace a odezvy
SOC	Security Operations Center – Kybernetické operační centrum
SLTR	Slantered Triangular Learning Rates – Strategie trénování
UEBA	User and Entity Behavior Analytics – Analýza chování uživatelů a entit
ULMFiT	Universal Language Model Fine-Tuning – Technika strojového učení
URL	Uniform Resource Locator – Jednotná adresa zdroje

A Obsah elektronických příloh

- `/` ..... kořenový adresář
  - `colab/` ..... sešity Google Colab
    - `BERT_log_categorization_final_1.ipynb` ..... první scénář
    - `BERT_log_categorization_final_2.ipynb` ..... druhý scénář
    - `BERT_log_categorization_final_3.ipynb` ..... třetí scénář
  - `datasets/` ..... datové sady užité pro jednotlivé scénáře
    - `first/` ..... datové sady pro první scénář
      - `Train_data_1.csv`
      - `Validate_data_1.csv`
      - `Test_data_1.csv`
    - `second/` ..... datové sady pro druhý scénář
      - `Train_data_2.csv`
      - `Validate_data_2.csv`
      - `Test_data_2.csv`
    - `third/` ..... datové sady pro třetí scénář
      - `Train_data_3.csv`
      - `Validate_data_3.csv`
      - `Test_data_3.csv`
  - `processing/` ..... zpracování datových sad
    - `main.py` ..... program pro vytvoření jednotlivých datových sad
    - `logs.zip` ..... shromážděné logy událostí ve formátu .log a .txt