VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE NEŽÁDOUCÍCH POŽADAVKŮ NA WEBU

DETECTION OF ANOMALOUS REQUESTS ON WEB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL SLOVÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK HRANICKÝ, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

153706

Ústav: Ústav informačních systémů (UIFS)
Student: **Slovák Michal**
Program: Informační technologie
Název: **Detekce nežádoucích požadavků na webu**
Kategorie: Bezpečnost
Akademický rok: 2023/24

Zadání:

1. Nastudujte principy fungování webových aplikačních bran firewall.
2. Seznamte se s metodami strojového učení. Zaměřte se na klasifikační algoritmy použitelné pro klasifikaci webových požadavků.
3. Vytvořte anotovanou datovou sadu legitimních (lidských) a nežádoucích (např. útoky, crawling) požadavků v rámci protokolu HTTP.
4. Vytvořte klasifikátor, který rozezná legitimní požadavky protokolu HTTP od nežádoucích.
5. Úspěšnost klasifikace zhodnoťte pomocí standardních metrik.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Razzaq, A., Hur, A., Shahbaz, S., Masood, M., & Ahmad, H. F. (2013, March). Critical analysis on web application firewall solutions. In *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)* (pp. 1-6). IEEE.
- Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). *HTTP: the definitive guide*. " O'Reilly Media, Inc.".
- Yu, L., Chen, L., Dong, J., Li, M., Liu, L., Zhao, B., & Zhang, C. (2020, July). Detecting malicious web requests using an enhanced textcnn. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 768-777). IEEE.

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3

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

Vedoucí práce: **Hranický Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Práce se zabývá vývojem klasifikátoru pro detekci nežádoucích požadavků na webový server s využitím metod strojového učení. Tento přístup vyžaduje vznik anotované datové sady a analýzu společných vlastnostní a charakteristik nelegitimních požadavků, které lze využít pro jejich kategorizaci. Dále se práce zabývá výběrem vhodného klasifikačního algoritmu. Výsledný model dosahuje váhovaného skóre F1 99.95 %, je spolehlivý a rychlý, což jej činí vhodným pro praktické nasazení.

Abstract

This thesis deals with the development of a classifier for detecting unwanted requests to a web server using machine learning methods. This approach requires the creation of an annotated dataset and the analysis of common features and characteristics of illegitimate requests that can be used to categorize them. Furthermore, the paper deals with the selection of an appropriate classification algorithm. The resulting model achieves a weighted F1 score of 99.95 %, is reliable and fast, making it suitable for practical deployment.

Klíčová slova

nežádoucí požadavky, webový aplikační firewall, klasifikace, datová sada, strojové učení, protokol HTTP, textová data, n-gram

Keywords

anomalous requests, web application firewall, classification, dataset, machine learning, HTTP protocol, text data, n-gram

Citace

SLOVÁK, Michal. *Detekce nežádoucích požadavků na webu*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický, Ph.D.

Detekce nežádoucích požadavků na webu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Radka Hranického, Ph.D. a uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Slovák
9. května 2024

Poděkování

Rád bych vyjádřil své upřímné poděkování Ing. Radkovi Hranickému, Ph.D., za jeho odborné vedení, cenné rady, vstřícnost a hlavně trpělivost během konzultací a vypracování této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Novodobý důraz na digitální transformaci silně ovlivňuje prostředí kolem nás. Stále více věcí kolem nás je digitalizováno. Fotky se již neukládají na kinofilmy, SMS zprávy nahradily dopisy a místo mincí a bankovek používáme kreditní karty. Vznikají nové elektronické výrobky, které lidem usnadňují život. Katalyzátorem této revoluce byl bezpochyby příchod Internetu. Ten ovlivňuje jakým způsobem lidé komunikují mezi sebou a stal se tak základním pilířem této revoluce. Každým rokem se procento celosvětové populace s přístupem k Internetu zvyšuje a v roce 2023 bylo k síti připojeno až 64.6 % celosvětové populace [17]. I proto se Internet stává čím dál tím více atraktivnějším prostorem pro útočníky, kteří můžou využít zranitelností např. na webových aplikacích.

Většina volně dostupných klasifikátorů využívá seznamy pravidel pro detekci nežádoucích požadavků. Tento přístup je v dnešní době nedostatečný – je náročné udržovat pravidla aktuální a jejich počet rapidně roste vzhledem k adaptivnímu trhu webových technologií. Využití metody strojového učení pro anotaci požadavků je mezi komerčními klasifikátory běžné [66, 10, 32].

Cílem této bakalářské práce je navrhnout klasifikátor pro detekci nežádoucích požadavků, který využívá metod strojového učení. Tento přístup vyžaduje důkladné zkoumání společných vlastností nelegitimních požadavků, které umožní jejich rozlišení od legitimních. Analyzovat a extrahovat tyto charakteristiky bude možné díky vytvořené anotované datové sadě obsahující maligní a benigní požadavky, která bude použita pro vývoj a testování navrženého klasifikátoru.

Přínos práce

Práce se zabývá návrhem a implementací systémů pro poloautomatický sběr legitimních a nelegitimních požadavků, které jsou následně využity k analýze společných vlastností těchto textových dat. Prozkoumávám využití n-gramů k zachycení charakteristik jednotlivých webových útoků, což může pomoci jejich identifikaci. Práce se dále věnuje výběru relevantních příznaků, včetně n-gramů, pro klasifikaci maligních a benigních požadavků. Výsledný model je schopen detekovat několik různých útoků zneužívajících zranitelností webových aplikací a dosahuje skóre F1 99.95 % a zpracovává data v jednotkách milisekund.

Struktura práce

V kapitole 2 se zabývám standardními přístupy pro detekci nežádoucích požadavků. Kapitola 3 se věnuje protokolu HTTP, který je využívaný pro webovou komunikaci. Na ni navazuje kapitola 4, v níž popisuju nejběžnější útoky na webové servery.

V kapitole 5 se věnuji využití přístupu strojového učení pro detekci nelegitimních požadavků. Kapitola 6 se zabývá vznikem anotované datové sady a analýzou společných vlastností textových dat. Vývoji modelu strojového učení je zasvěcena kapitola 7 a vše je shrnuto v kapitole 9.

Kapitola 2

Přehled existujících studií

Vědecká činnost v oblasti detekce nežádoucích požadavků na webový server byla započata prací od Kruegela a Vigny [43], kteří jako první navrhli systém pro identifikaci anomálií ve webovém provozu. Předpokládají, že normální provoz bude mít normální distribucí znaků a naopak nežádoucí provoz rozdělení znaků odlišné. Jejich přístup však zpracovává pouze jednotné identifikátory zdroje a nepracuje se zbytkem webového požadavku, ve kterém se mohou skrývat další hrozby.

2.1 Detekce průniků

Přístupy založené na detekci průniků v provozu (IDS) [44, 43, 48, 70, 18] kontrolují, zda webový požadavek odpovídá standardnímu provozu webové aplikace. Tyto systémy dosahují dobrých výsledků v oblasti detekce, jsou schopny identifikovat útoky nultého dne, ale nastává u nich problém s obecnosti řešení. Každá webová aplikace má odlišný normální provoz a je nutné systém individuálně naučit, jak provoz běžně vypadá a to i při změně chování webové aplikace díky vývoji.

Krueger a kol. [44] použili princip IDS a navrhli unikátní přístup k nežádoucím požadavkům – pokusit se z požadavku odstranit hrozbu a předat ho webové aplikaci ke zpracování namísto navrácení chyby. Útočník pak nezíská žádnou zpětnou vazbu ohledně detekce útoku, což mu může ztížit jeho penetrační testování.

Makiou a kol. [48] se specializovali na detekci útoků injekce SQL. Upozornili, že útoky mohou být přítomny nejen v URL, ale i v hlavičkách webových požadavků. Pro klasifikaci použili algoritmus naivní Bayes, který byl integrován do jejich bezpečnostního systému HIPS. Tento systém každému požadavku přiřazuje pravděpodobnost obsahu útoku injekce SQL. Požadavky s vysokou pravděpodobností jsou následně podrobeny detailnějšímu zkoumání pomocí seznamu pravidel, o kterých se podrobněji zmiňuji v sekci 2.3.

2.2 Modely umělé inteligence

Systémy založené na umělé inteligenci pro detekci nežádoucích požadavků [28, 51, 50, 34] jsou dnes již standardem v nejpoužívanějších komerčních systémech pro ochranu webových aplikací [66, 32, 10]. Umožňují vysokou rychlost klasifikace a jsou schopné detekovat nejrůznější typy útoků. Tento přístup je však velmi závislý na datové sadě, pomocí které je umělá inteligence trénována i testována. Příliš specifická datová sada by nemusela dosaho-

vat dostačující preciznosti na všech webových aplikacích. Naopak příliš obecná datová sada by mohla mít problém s pokrytí alespoň běžných hrozeb.

Díaz-Verdejo a kol. [20] shrnuli největší problémy s vývojem systémů pro detekci nežádoucích požadavků a mimo jiné i tvorba datové sady je mezi nimi. Zjistili, že většina autorů pro vývoj klasifikátorů využila zastaralých datových sad, které neodpovídají aktuálním webovým útokům, pracovali s malým množstvím dat aj.

Gharibeh a kol. [28] porovnali 4 různé algoritmy založené na umělé inteligenci a všechny dosahují přesnosti identifikace nad 98 %. Problémem řešení může být datová sada, která je specifická pro jeden webový e-shop a není tak jasné, jaká by byla preciznost modelů při provozu u jiné webové aplikace. Stejná otázka vzniká i u dalších prací (např. [70, 51]).

Vartouni a kol. [51] využili princip hlubokého učení, kde vytvořily tři modely umělé inteligence, které učili pouze na validních požadavcích – tzv. jednotřídní klasifikace. Tento přístup řeší problémy dvoutřídní klasifikace, jako je složitý sběr nežádoucích požadavků, disbalance tříd a další [1].

Wressnegger a kol. [69] zkoumali techniky klasifikace a detekce průniků v oblasti nelegitimních webových požadavků. Pro analýzu použili princip n-gramů a pro rozhodování mezi dvěma metodami stanovili kritéria jako odchylku, hustotu a proměnlivost dat. Ve svém závěrečném shrnutí dospěli k závěru, že nelze definitivně určit, že by jedna metoda byla lepší než druhá, jelikož každá má své specifické aplikace.

Kolter a Maloof [42] využili pro klasifikaci maligních a benigních spustitelných souborů přístup n-gramů. Z původních 255 miliónů n-gramů vybraly pouze stovky, které jsou relevantní pro klasifikaci a dosáhli přesnosti 98 % při identifikaci nelegitimních souborů.

2.3 Seznamy pravidel

Řešení na bázi seznamu pravidel spolehlivě funguje na již známé hrozby, ale je nutné tento index aktualizovat kvůli neustále měnícím se útokům nebo při kontinuálním vývoji webové aplikace [55]. Tato údržba může být velice drahá, útoky se mění každým dnem, a vyžaduje činnost kvalifikovaného správce v oblasti ochrany webového provozu [9]. Vzhledem k vážným nedostatkům se tento přístup v dnešní době využívá pouze dodatečně k již zmíněným metodám.

Kapitola 3

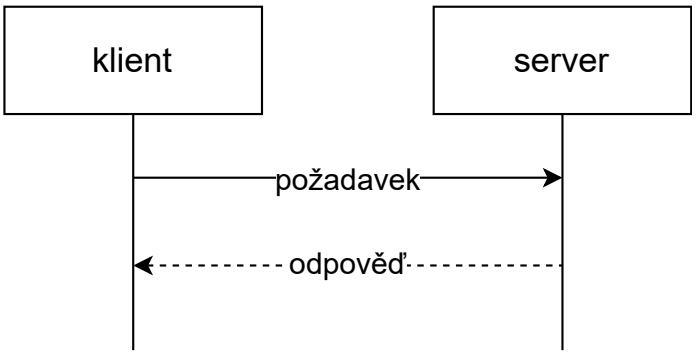
Hypertext Transfer Protocol

Hypertextový transferový protokol, anglickou zkratkou HTTP a takto i nadále v této práci, je internetový protokol na aplikační vrstvě pro přenos dat na Internetu. Využívá paradigmatu klient–server a komunikace je prováděna výhradně formou zpráv. Protokol je definován v několika verzích a tato práce se zabývá tou s označením HTTP/1.1 [52]. I když existují novější verze protokolu, tak jsou zpětně kompatibilní a případné novinky, které přinesly, nejsou pro účely této práce podstatné, viz 4.

3.1 Typy zpráv protokolu HTTP

Zpráva, kterou klient odesílá, se označuje jako požadavek a na něj server reaguje odpovědí. Oba tyto způsoby komunikace mají pevně stanovenou syntaxi definovanou protokolem, ale obsahují i sdílené prvky. Mezi ně patří mimo jiné i verze protokolu, kterou chtějí oba subjekty využívat pro komunikaci nebo volitelné tělo. Součástí obou typů zpráv mohou být i tzv. hlavičky.

Paradigma klient–server je znázorněno níže na obrázku 3.1. Komunikaci vždy zahajuje klient formou požadavku a nazpět od serveru získává odpověď.



Obrázek 3.1: Schéma HTTP komunikace

Oba typy zpráv se prezentují v textovém formátu a sdílejí podobnou syntaxi, s výjimkou prvního řádku, který se liší podle typu zprávy. U požadavku se mu říká požadavkový řádek, anglicky Request-Line, zatímco u odpovědi se jedná o statusový řádek, anglicky Status-Line. Poté mohou následovat hlavičky a volitelné tělo obsahující data. Níže jsou tyto struktury popsány podrobněji.

3.1.1 Požadavek

Základem této zprávy je jednotný identifikátor zdroje, anglickou zkratkou URI, který identifikuje zdroj nebo umístění, na které se požadavek vztahuje.

Další klíčovou součástí požadavku je specifikace metody, kterou má webový server na daném zdroji provést. Protokol HTTP definuje rozsáhlou škálu metod, ale povoluje i použití nespecifikovaných metod [52]. Nejzákladnější je následující čtveřice, často definována jako CRUD [1], která shrnuje základní operace s daty:

- GET pro získání dat zdroje,
- POST pro odesílání dat,
- PUT pro úpravu zdroje,
- DELETE pro smazání.

Komponenta query

Důležitou součástí URI [4] je i komponenta query, která umožňuje předávání nestrukturovaných dat mezi klientem a serverem. Společně s elementem cesty tvoří kompletní identifikaci zdroje [4]. Její začátek je indikován výskytem znaku otazníku a může obsahovat téměř jakýkoli text. Ve webovém kontextu tato komponenta zpravidla obsahuje páry hodnot klíč–hodnota, které jsou oddělené znakem ampersandu a jsou označovány jako parametry query.

Následuje ukázka požadavkového řádku, který využívá metody GET pro získání zdroje „index.html“ včetně výskytu parametrů query *id* a *p*:

```
GET /index.html?id=24&p=test HTTP/1.1
```

3.1.2 Odpověď

Nejdůležitější součástí odpovědi je numerický statusový kód s třemi ciframi a slouží k označení, jakým způsobem došlo ke splnění požadavku. Tyto numerické hodnoty jsou rozděleny do skupin na základě první cifry – například statusové kódy odpovídající formátu 2xx, kde x je libovolná číslice, značí, že došlo k úspěšnému splnění požadavku. Význam všech možných statusových kódu je definovaný protokolem HTTP. Níže je příklad statusového řádku s kódem 404, který znamená, že zdroj nebyl nalezen:

```
HTTP/1.1 404 Not Found
```

3.2 Hlavička

Hlavička, anglicky header, je definována jako pár klíč–hodnota, kde klíč přiřazuje dané hodnotě význam. Tyto dvojice slouží k přenosu doplňujících informací, jako jsou metadata, autentizační údaje, řízení mezipaměti, informace o obsahu a další důležité údaje, které umožňují klientovi a serveru lépe porozumět požadavku a odpovědi. Sémantika těchto klíčů je specifikována v definici protokolu, ten ovšem také povoluje klientovi a serveru vytvořit v případě potřeby vlastní klíče [52].

Hlavičky lze rozdělit do čtyř skupin na základě toho, jak jsou využívány:

[1] Create, Read, Update, Delete

- obecné, které se mohou vyskytnout v obou typech zpráv,
- požadavkové,
- odpověďové,
- entitní, které doplňují informace o zasílaných datech [52].

Protokol nezakazuje, aby se hlavičky typické pro požadavky vyskytly i v odpovědích, ale jejich sémantika je ponechána na webovém serveru.

V HTTP zprávě může být libovolné množství hlaviček a jejich uspořádání není přísně určeno, ale každý požadavek musí obsahovat hlavičku Host, která slouží pro upřesnění koncového serveru:

```
Host: www.vut.cz
```

3.3 Tělo

Pro přenos dat využívá protokol HTTP tělo, které může být obsaženo v obou druzích zpráv. Nachází se na konci zprávy za hlavičkami, s výjimkou situací, kdy jeho použití je omezeno metodou v případě požadavku nebo statusovým kódem u odpovědi. Přítomnost těla je podmíněna přítomností hlavičky *Content-Length*, která udává délku těla. Údaje o typu dat, použitém kódování, jazyce a další relevantní informace týkající se obsahu těla zprávy jsou obsaženy v hlavičkách. Ukázka výskytu těla v požadavku HTTP je znázorněna níže ve výpisu 3.1.

```
POST /post HTTP/1.1
Host: httpbin.org
Accept: */*
Content-Length: 47
Content-Type: application/x-www-form-urlencoded
User-Agent: Chrome/119.0.0.0 Safari/537.36

url=http%3A%2F%2Fwww.google.com&status_code=301
```

Výpis 3.1: Příklad komplexního požadavku

Kapitola 4

Útoky na protokolu HTTP

Je důležité si uvědomit, že protokol HTTP je pouze zodpovědný za organizaci komunikace mezi klientem a serverem na Internetu. Definuje pouze pravidla pro výměnu dat mezi těmito dvěma entitami a sám o sobě tak není standardně zranitelným bodem. Výjimkou je například nedávná zranitelnost HTTP2/ Rapid Reset, která využívá nedostatku definice protokolu HTTP verze 2 pro způsobení DDoS útoku [13].

Avšak zranitelnosti a útoky mohou vzniknout v různých oblastech kolem HTTP komunikace. Jedním z možných míst jsou chyby v implementaci webového klienta nebo serveru. Tyto závady mohou být využity v rámci útoků nultého dne – to jsou takové útoky, které nejsou všeobecně známé a jejich detekce a následné zamezení může trvat dny, týdny či dokonce měsíce. V letech 2021 a 2022 bylo objeveno až 41 takových útoků zneužívající zranitelnosti ve webových prohlížečích [64].

Dalším bodem potenciálních rizik jsou webové aplikace, které běží na webových serverech. V kontrastu k omezeným možnostem z hlediska výběru konkrétních webových serverů či klientů, je tvorba webových aplikací dostupná téměř každému. A právě proto mohou nepozorní programátoři během vývoje vytvořit zranitelnosti pro vniknutí do systému. Útočníci mohou těchto vad využít, aby získali neoprávněný přístup k datům, spustili neautorizované akce nebo způsobili výpadek či poškození systému.

4.1 Nejčastější zranitelnosti webových aplikací

OWASP Top 10 [21] je dokument určený pro vývojáře, jenž se zabývá zranitelnostmi webových aplikací a identifikuje deset nejčastějších. Tento seznam se průběžně, zhruba každé čtyři roky aktualizuje, což umožňuje sledovat dynamické trendy v oblasti narušení webových aplikací. V následujících podsekcích jsou detailně popsány včetně příkladů vybrané útoky převzaté jak z OWASP Top 10, tak z bezpečnostních zpráv komerčních WAF (např. [67]).

4.1.1 Injekce SQL

Structured Query Language, anglickou zkratkou SQL, je programovací jazyk používaný pro manipulaci s daty uloženými v relační databázi. Tento útok vkládá příkaz v jazyku SQL do argumentu, který webová aplikace používá pro komunikaci se zmíněným úložištěm informací. Využití této zranitelnosti může vést k neoprávněnému přístupu útočníka k citlivým datům uloženým v databázi a to včetně případné možnosti je upravit, smazat apod.

Systém řízení báze dat, anglickou zkratkou DBMS, zprostředkovává komunikaci s daty uloženými v databázi. Většina moderních webových aplikací má potřebu informace ukládat nebo modifikovat a často využívají právě databázi jako úložiště informací. S ohledem na jednoduché provedení této zranitelnosti je zřejmé, že s jejím pokusem o zneužití se setká sebemenší web [40]. Tomuto faktu nasvědčují i statistiky výskytu zranitelností, kdy je tento útok na prvních místech již roky (viz [67, 22, 21].

Pro porozumění tohoto útoku je vhodné si představit ukázku zranitelného dotazu SQL, který webová aplikace může použít pro dotazování databázového systému:

```
"SELECT * FROM "Uzivatele" WHERE id = " + hodnota_argumentu
```

Tento dotaz je navržen k získání všech záznamů z tabulky *Uzivatele*, kde hodnota pole s označením id odpovídá hodnotě poskytnuté v argumentu požadavku. Pokud by útočník vložil do požadavku data odpovídající útoku injekce SQL, například 1 OR 1=1, pak by konečný dotaz vypadal takto:

```
SELECT * FROM "Uzivatele" WHERE id = 1 OR 1=1
```

a útočník by tak získal všechna data z tabulky Uzivatele, protože podmínka 1=1 je platná pro všechny záznamy.

4.1.2 Cross-site scripting

Cross-site scripting, anglickou zkratkou XSS, představuje typ útoku, kdy se útočník snaží vložit škodlivý kód do webové aplikace. Tento kód je následně distribuován běžným uživatelům při návštěvě webu a jejich webový prohlížeč automaticky tento kód provádí.

Útočník tímto způsobem může získat přístup k citlivým údajům uložených v prohlížeči. Mezi tyto důležité informace typicky patří tzv. cookies, což jsou data v prohlížeči sloužící k identifikaci uživatele na webové stránce [41]. Jejich zneužití tak umožní útočníkovi se vydávat za jinou osobu.

Použití útoku XSS v požadavku je znázorněno na výpisu 4.1, kdy při úspěšném zneužití by útočník byl schopen u klienta spustit libovolný kód JavaScriptu.

```
POST /add_post
Host: www.forum.cz
Content-Length: 39

{"name":"<body onload=alert('test1')>"}
```

Výpis 4.1: Příklad nelegitimního požadavku s využitím útoku XSS

4.1.3 Path traversal

Útok path traversal zahrnuje snahu útočníka získat neoprávněný přístup k souborům nacházejícím se mimo kořenový adresář webového serveru [23]. Využívá textové sekvence „../“, která v souborovém systému odkazuje na rodičovskou složku základního adresáře. Této funkcionality může útočník využít, pokud webová aplikace očekává informace o cestě k souboru v požadavku a následně poskytuje jeho data v HTTP odpovědi.

Dle implementace webové aplikace je možné pomocí této zranitelností i spustit libovolný kód, jenž je hostovaný na nějaké webové adrese nebo spouštět systémové programy [23].

Výpis 4.2 znázorňuje požadavek, který umožňuje zobrazit uživateli soubor na webovém serveru. Využití útoku path traversal je znázorněno ve výpisu 4.3, kde útočník se snaží zobrazit textový soubor „/etc/passwd“, který v kontextu operačních systému s jádrem Linux obsahuje informace o systémových uživatelích.

```
GET /index.php?file=dokument.pdf
Host: www.google.com
```

Výpis 4.2: Příklad benigního požadavku

```
GET /index.php?file=../../../etc/passwd
Host: www.google.com
```

Výpis 4.3: Příklad maligního požadavku s využitím útoku path traversal

4.1.4 Útok XML External Entity

Extensible Markup Language, anglickou zkratkou XML, je jazyk, který umožňuje uspořádat data ve hierarchické formě podobné stromové struktuře. Jeho důležitou součástí jsou elementy, které definují strukturu dokumentu. Útok XML External Entity (XXE) zneužívá entit, které slouží k definování speciálních znaků nebo jiných textových dat [47].

Rozlišujeme dva typy entit – interní a externí. Druhý zmíněný typ je schopen přistupovat k fyzickým datům na systému a importovat je tak do dokumentu XML. Této funkcionality může útočník využít k neoprávněnému přístupu k systémovým informacím ale v okrajových případech i k získání kontroly nad systémem.

Výpis 4.4 znázorňuje příklad zneužití externí entity k získání obsahu souboru „/etc/shadow“, který v některých operačních systémech obsahuje citlivé údaje o heslech systémových uživatelích.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT bar ANY>
  <!ENTITY attack SYSTEM "file:///etc/shadow">
]>
<bar>&attack;</bar>
```

Výpis 4.4: Ukázka využití externí entity pro získání citlivých údajů

4.1.5 HTTP smuggling

HTTP smuggling je technika objevená v roce 2005 [45], využívá rozdílné implementace zpracování požadavků mezi servery zapojenými v komunikaci mezi webovým klientem a serverem.

Verze 1 protokolu HTTP dovoluje rozdělit zprávu na menší segmenty s využitím hlavičky *Transfer-Encoding*. Společně s hlavičkou *Content-Length*, která definuje délku těla zprávy,

může dojít k rozporu – pokud jsou obě tyto hlavičky obsaženy v jednom požadavku, může to vést k nejasnosti, kde tato zpráva končí. To vytváří prostor pro nekonzistenci v implementaci serverů - například mezi cachovacím a webovým serverem, kde jeden může preferovat hodnotu hlavičky *Content-Length* a druhý hlavičku *Transfer-Encoding* nebo naopak.

Díky této technice může narušitel využít jiného typu útoku, který může skrýt a vyhnout se tak bezpečnostním prvkům v komunikaci. Kromě toho je i často využívána metoda *cache poisoning* [39], která může vést k phishingu nebo spuštění maligního kódu u návštěvníka webové stránky, stejně jako v případě útoku XSS, viz 4.1.2.

Konkrétní ukázka této techniky je demonstrována ve výpisu 4.5, kdy v jedné zprávě jsou obsaženy dva požadavky.

```
POST / HTTP/1.1
Host: vulnerable.com
Content-Length: 43
Transfer-Encoding: chunked
0
GET /admin HTTP/1.1
Foo: X
```

Výpis 4.5: Příklad požadavku využívající metody HTTP smuggling [46]

4.1.6 Anomálie protokolu HTTP

Tato sekce se zabývá útoky, jež využívají místy volné definice protokolu HTTP či tímto protokolem referencovaných komponent, nebo nejsou natolik zásadní pro vlastní podsekci.

Chybějící hlavička User-Agent

Hlavička User-Agent obsahuje informaci o webovém klientovi, který požadavek odeslal. Protokol doporučuje, aby tato hlavička byla obsažena ve všech požadavcích. Tento návrh je dodržován a všechny webové prohlížeče ho implementují [68]. Pokud požadavek tuto hlavičku neobsahuje, může se jednat o manuálně vytvořený požadavek, který není legitimní.

Z hlavičky User-Agent je možné získat informace o klientovi, jako je operační systém včetně verze, použitá klientská aplikace a další:

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36
```

Duplikace parametrů query

Vzhledem k flexibilní definici query komponenty v URI je možné, aby unikátní identifikátor zdroje v požadavku obsahoval duplicitní klíče parametrů query. Jak se s tímto scénářem zachází, závisí čistě na implementaci webové aplikace nebo jiných systémů zapojených v komunikaci mezi klientem a serverem, včetně bezpečnostní infrastruktury ochraňující webové servery před útoky.

Některé tyto systémy mohou zpracovat pouze první výskyt parametru query s duplicitním klíčem, zatímco webová aplikace může využít druhý výskyt, který by mohl obsahovat útok využívající danou zranitelnost. Dalším možným scénářem je, když webová aplikace

spojí hodnoty těchto parametrů dohromady. To umožňuje útočníkovi rozdělit škodlivý obsah do několika parametrů a tak obcházet detekci ochranných systémů.

Ukázka výskytu duplicitních parametrů query, které při jejich konkatenaci obsahuje útok injekce SQL:

```
https://www.example.com?id=1%20OR&id=%201%3D1
```

Výskyt speciálních znaků

Nulový bajt, také nazývaný nulový znak, má nejen v programovacím jazyce C speciální význam jako označení konce textového řetězce. Tato charakteristika může být využita k předčasnému ukončení textového řetězce, což může ovlivnit zamýšlenou logiku webové aplikace. Tento druh útoku bývá často kombinován s dalšími útoky, jako jsou například injekce SQL, path traversal a další [53].

Injekce CRLF

Sekvence Carriage Return Line Feed, zkratkou CRLF, v textové formě reprezentována jako "\r\n", slouží k oddělení částí zpráv a hlaviček v protokolu HTTP. Útočník může tuto sekvenci využít ke změně odpovědi HTTP [24], protože tato kombinace umožňuje vložení libovolných hlaviček nebo definování obsahu těla zprávy. Takový útok může být prováděn v kombinaci s jinými útoky, například s XSS [24]. V případě úspěchu může uživatel, který není podezřívavý, kliknout na odkaz, který je schopen spustit libovolný skript definovaný útočníkem.

4.2 Nežádoucí procházení webu

Web crawling, v českém překladu procházení webu, je proces, při kterém automatizovaný software nazývaný pavouk systematicky prochází webové stránky za účelem indexace obsahu. Významné vyhledávací nástroje využívají tyto programy k získání relevantních informací o webových stránkách, což umožňuje poskytovat kvalitní informace uživatelům při vyhledávání [27, 12].

Administrátoři webových stránek mají možnost zakázat přístup určitým webovým pavoukům, aby nedocházelo k procházení celého webu nebo omezení přístupu k určitému obsahu. Nicméně tyto programy nemusí vždy dodržovat tato pravidla, obzvlášť je-li pavouk v rukou útočníka [27].

Každý požadavek na webový server spotřebovává výpočetní výkon. Pokud útočník využívá pavouky s minimálními prodlevami pro procházení, může to způsobit nadměrné zatížení serveru. To může vést v nejhorším případě až k znepřístupnění celé aplikace a ztrátě dat a případného ušlého zisku.

4.3 Ochrana webových aplikací před útoky

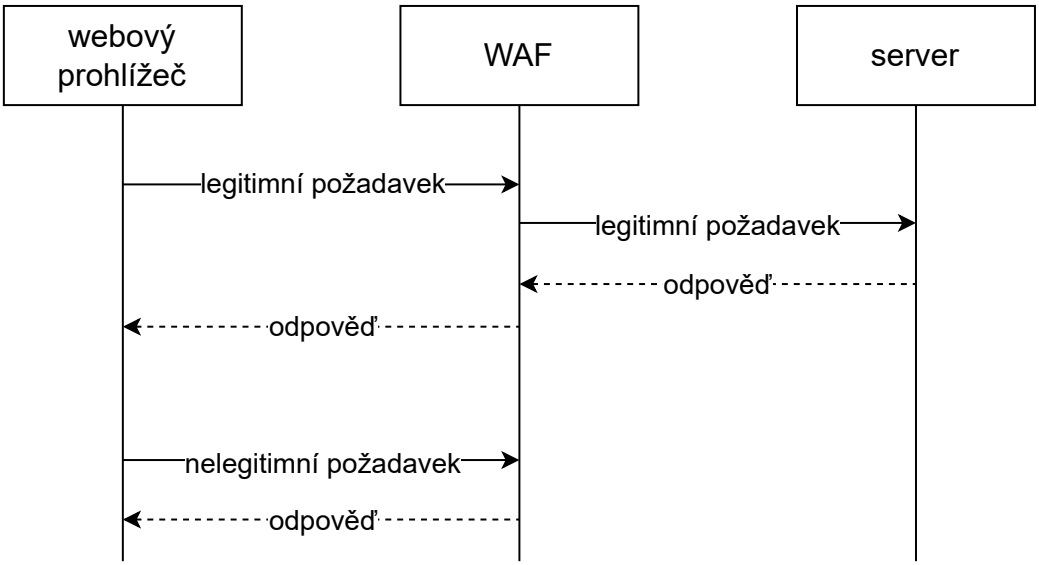
Firewall webových aplikací, anglickou zkratkou WAF a takto i nadále v této práci, poskytuje webům ochranu před nežádoucími požadavky [56], které by mohly vést nejen ke kompromitaci dat, ale i celého systému. Brána firewall pracuje na aplikační vrstvě modelu OSI [33] a zabývá se pouze zranitelnostmi vyskytující se ve webových aplikací.

Jejím úkolem je sledovat a filtrovat příchozí požadavky a rozhodovat o jejich malignosti. V případě identifikace jakéhokoli rizika je požadavek blokován ještě před tím, než by mohl dosáhnout webové aplikace a využít potenciální zranitelnosti [9].

Kromě standardní obecné ochrany před běžnými formami webových útoků mají správci možnost definovat vlastní pravidla, která mohou cíleně chránit proti konkrétním typům zneužití. Tato pravidla umožňují blokovat příchozí IP adresy podle jejich geografické lokace, specifikovat regulární výrazy, které nesmí být přítomny v příchozích požadavcích a kontrolovat další specifické vlastnosti. Například společnost Cloudflare zjistila, že až 40% pravidel definovaných zákazníky se týká blokování IP adres s nevhodnou geolokací [67].

Firewall může být součástí fyzického serveru, implementován jako modul pro webové servery nebo zaveden jako reverzní proxy server, který je prostředníkem v komunikaci mezi klientem a serverem a umožňuje sledovat a případně modifikovat síťový provoz [11].

Obrázek 4.1 popisuje chování WAF jako reverzní proxy server. Kontroluje příchozí požadavky a nežádoucí odkloní tak, aby nemohly využít zranitelnosti na webovém serveru. Naopak legitimní přesměruje na webový server. Role prostředníka v komunikaci mimo jiné umožňuje i skrytí IP adresy cílového webového serveru před uživateli, což posiluje úroveň bezpečnosti.



Obrázek 4.1: Schéma HTTP komunikace s využitím WAF

Kapitola 5

Strojové učení

Strojové učení je vědní obor, který poskytuje počítačům schopnost učit se bez explicitního programování [58]. Tento proces umožňují algoritmy strojového učení, které definují způsob a pravidla učení trénované úlohy. Identifikují vzory v datech relevantních pro daný úkol [8], které algoritmus využije k vytvoření modelu strojového učení schopného vykonat danou úlohu. V kontextu této bakalářské práce se model stává klasifikátorem, který je schopen požadavek označit jako maligní, či benigní.

Proces učení algoritmu není zcela automatický; nemůžeme jednoduše přikázat, co se má naučit. Je nezbytné, aby algoritmus měl k dispozici příklady dat, na základě kterých analyzuje a vytváří rovnici, jež odpovídá očekávaným výstupům [6]. Vstupy této rovnice jsou organizovány do uspořádaných n-tic známých jako vektory příznaků, anglicky *feature vector*. Každý příznak ve vektoru reprezentuje určitou vlastnost datového záznamu. Výběr těchto příznaků je klíčový pro vytvoření úspěšného modelu strojového učení [8, 61], což je téma, kterému se podrobněji věnuji v kapitole 6.

Využití strojového učení pro splnění úlohy je vhodné v případech, kdy je úloha natolik komplexní, že je těžko popsatelná člověkem formou algoritmů [49]. To se týká i zpracování obrovských množství dat, tzv. big data, které je motivováno nejen vědci pro získání užitečných postřehů [35], ale i společnostmi pro získání konkurenční výhody [57]. Dalším příkladem je i nutnost adaptace algoritmu v průběhu času kvůli personalizaci pro uživatele [49], nebo v případě, že se data mění v čase [15].

5.1 Rozdělení algoritmů

Standardně se algoritmy strojového učení klasifikují do několika kategorií podle principu jejich fungování a formátu vstupních dat. V následujících podsekcích je toto dělení popsáno podrobněji.

5.1.1 Učení s učitelem

Princip učení s učitelem vychází z využití anotovaných vstupních dat pro trénink algoritmu, kde každý vektor příznaků obsahuje výstupní označení, neboli anotaci [31]. Tento přístup umožňuje algoritmu se naučit, jak předpovídat výstupní hodnoty pro nová, zatím neanotovaná data [8, 31].

Jedná se o nejrozšířenější kategorii v tomto dělení, obsahující především dvě základní úlohy: klasifikaci a regresi [61]. Jak jsem již zmínil na začátku této kapitoly, první zmíněná varianta hraje v této bakalářské práci klíčovou roli. Umožňuje kategorizovat vstupní data

do předem definovaných tříd [2] a v kontextu této práce rozlišit, zda je vstupní HTTP požadavek legitimní.

Na rozdíl od klasifikace, regrese se zaměřuje na predikci spojitých numerických hodnot [2, 8], což najde uplatnění například při odhadu cen ojetých vozidel nebo při předpovídání teplot.

5.1.2 Učení bez učitele

V kontextu učení bez učitele nejsou data, se kterými algoritmus pracuje, dopředu opatřena anotacemi nebo výstupními hodnotami [2]. Tento přístup umožňuje algoritmu identifikovat vzory a souvislosti v datech samotných, aniž by o nich programátor měl předchozí znalosti [8]. Tento přístup se uplatní zejména v situacích, kdy jsou data natolik komplexní, že je nelze snadno pochopit, nebo když je anotace dat příliš časově a finančně náročná.

Do této kategorie patří i technika jednotřídní klasifikace [30], na kterou jsem se již dříve odkazoval v sekci 2.2.

5.1.3 Kombinované učení

Princip kombinovaného učení, v angličtině jako *semi-supervised learning*, spojuje již zmíněné techniky učení s učitelem a bez učitele do jediného přístupu [61]. Vstupní sada dat obsahuje jak anotované, tak neanotované záznamy a tím řeší situace, kdy správná anotace dat je obtížná, ale získat velké množství dat je snadné [14]. Tato metoda tedy adresuje jednu z klíčových výzev při využití učení s učitelem [25].

Tento přístup také nachází uplatnění v generativních soupeřících sítích, známých pod anglickou zkratkou GAN, jež se používají k vytváření generovaného obsahu, jako jsou obrázky, texty a další [7].

5.1.4 Posilované učení

Systém založený na principu posilovaného učení je označován jako agent, který aktivně shromažďuje informace o svém prostředí a na základě nich vykonává akce s cílem maximalizovat získané odměny [61, 8]. Agent se učí samostatně [65], bez přímého vedení, a postupně objevuje, které strategie jsou nejúčinnější pro získávání maximální odměny v čase. Proces lze srovnat s metodou cukru a biče, kdy každá akce je odměněna buď pozitivně nebo negativně, což napodobuje lidský proces učení se principem pokus-omyl [65].

Využití posilovaného učení se rozšířilo do různých oblastí, včetně autonomního řízení, robotiky a počítačových her [65, 61]. Zvláště pozoruhodným úspěchem se stalo poražení nejlepšího hráče v Go programem AlphaGo v roce 2017, díky kterému obdržela tato metoda značnou pozornost [8].

5.2 Klasifikační algoritmy

V této sekci se zaměřím na algoritmy, jež jsou vhodné pro úlohu klasifikace dat, které použiji při implementaci klasifikátoru pro třídění legitimních a nelegitimních požadavků.

5.2.1 Algoritmus k-nejbližších sousedů

Algoritmus k-nejbližších sousedů, anglickou zkratkou KNN a takto i nadále v této práci, patří mezi tzv. paměťově učící se algoritmy [61]. Tyto metody se nezaměřují na generalizaci

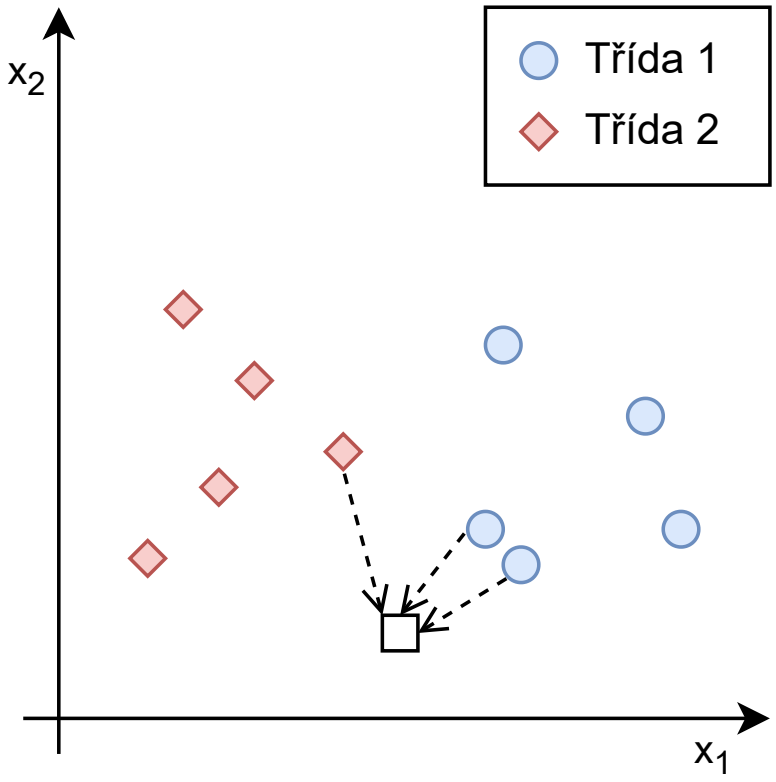
problému, ale na porovnávání nových dat s těmi, která algoritmus již zpracoval během trénovací fáze.

V rámci metody KNN algoritmus nejprve transformuje všechny vektory příznaků do dimenzionálního prostoru, jehož velikost odpovídá počtu atributů záznamů [61]. Následně probíhá klasifikace nového prvku na základě měření vzdáleností k nejbližším k sousedům v tomto prostoru [2]. Výsledná anotace je pak odvozena z průměru jejich anotací. Počet nejbližších sousedů je konfigurovatelný a vychází z názvu metody.

Algoritmus KNN je relativně jednoduchý, avšak má problém s rovnocenným přikládáním důležitosti všem příznakům [3]. Toto může vést k nepřesnostem ve výsledcích modelu, jelikož ne všechny atributy jsou pro přesnou predikci stejně relevantní.

Dalším problémem je tzv. prokletí dimenzionality, které nastává s narůstajícím počtem příznaků a tím i dimenzí, v nichž se data nachází. To způsobuje, že i podobná data mohou být ve vysokodimenzionálním prostoru vnímána jako vzájemně vzdálená. Tento fenomén postihuje nejen metodu KNN, ale také další algoritmy pracující s daty ve vícedimenzionálních prostorech [3].

Na obrázku 5.1 je ilustrován proces predikce nového datového záznamu, na němž se podílejí tři sousedé ze dvou tříd.



Obrázek 5.1: Ukázka klasifikace nového datového záznamu s využitím metody KNN

5.2.2 Bayesovské metody

Skupina Bayesovských metod pro predikci využívá Bayesovu větu, která obecně „vyjadřuje vztah mezi podmíněnými pravděpodobnostmi dvou jevů“ [19]. Zapisuje se jako $P(A|X)$, což v kontextu této práce značí pravděpodobnost, že datový záznam X náleží do třídy A.

Obecně tento zápis značí pravděpodobnost výskytu jevu A za předpokladu výskytu jevu X).

Naivní Bayesův klasifikátor vychází z předpokladu nezávislosti jednotlivých příznaků [3, 26]. Tento předpoklad často neodpovídá realitě a jeho omezení řeší například Bayesovské sítě [26], které však pro jejich komplexnost v této práci dále nerozvíjím. Během tréninku algoritmus zkoumá tyto atributy a sestavuje statistiky, které poté využívá k výpočtu pravděpodobnosti pro predikci anotace.

Přestože je tento přístup poměrně jednoduchý, často dosahuje srovnatelných výsledků se složitějšími algoritmy a efektivně zvládá zpracování velkých datových sad [31]. To se stává výzvou zejména u lineárních modelů, které vyžadují delší dobu tréninku kvůli své časové náročnosti [3] a více se jim věnuji v podsekci 5.2.3.

5.2.3 Lineární modely

Lineární modely pro predikce využívají lineární funkci vstupních parametrů ve tvaru:

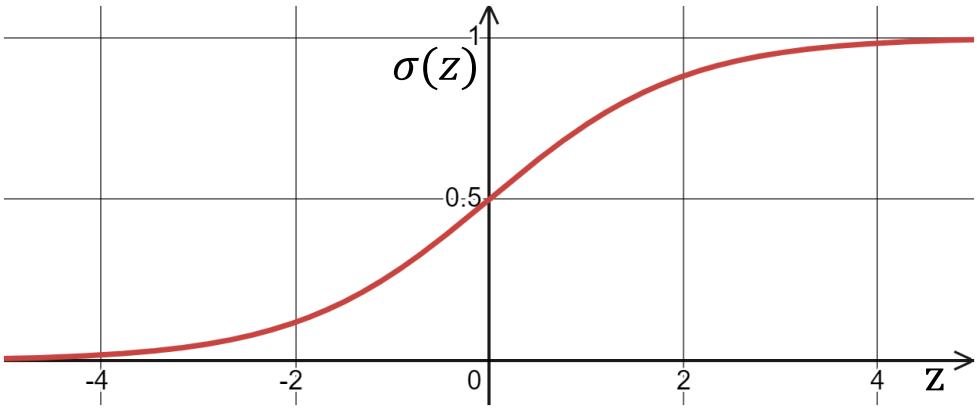
$$y = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + b \tag{5.1}$$

kde b je absolutní člen, x_n odpovídá hodnotám jednotlivých příznaků a β_n je koeficient, který každému atributu přikládá váhu podle jeho důležitosti [2]. Tyto koeficienty jsou vypočítány algoritmem během trénovacího procesu. Výsledek této rovnice může teoreticky nabývat jakékoli reálné hodnoty, a proto je tento přístup použitelný pouze pro regresní úlohy, viz 5.1.1.

Logistická regrese je i přes své jméno klasifikační algoritmus, který vychází z předešlé lineární rovnice 5.1, jež dále rozšiřuje o logistickou funkci, též označovanou jako sigmoida:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.2}$$

kde y pochází z lineární rovnice 5.1. Sigmoida nabývá hodnot v intervalu od 0 do 1, což lze přímo interpretovat jako pravděpodobnost, že určitý datový záznam spadá do specifické třídy [36, 31]. Graf této funkce je zobrazen na obrázku níže 5.2.

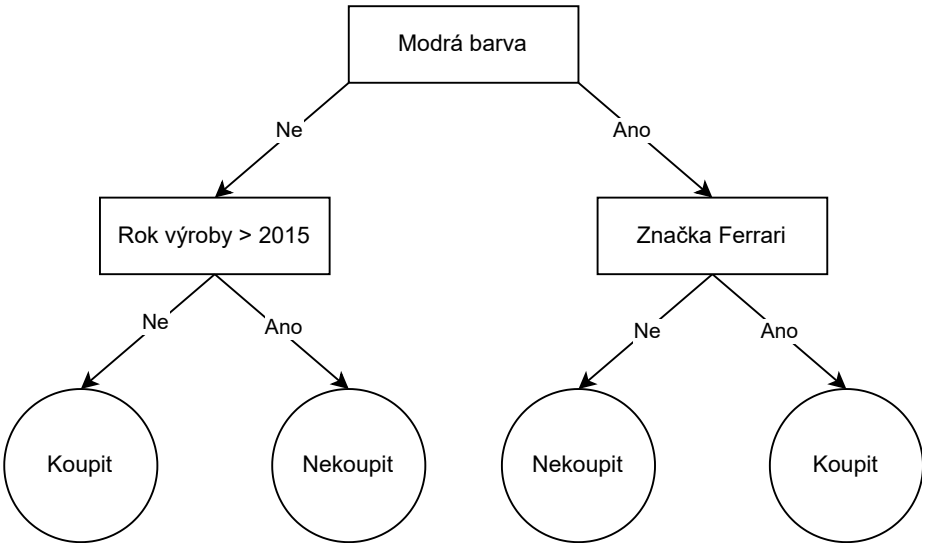


Obrázek 5.2: Zobrazení grafu logistické funkce

Tento přístup, využívající podmíněné pravděpodobnosti podobně jako Bayesovské modely zmíněné v podsekci 5.2.2, se odlišuje tím, že místo zkoumání příznaků, které definují třídu, využívá diskriminační metodu k identifikaci příznaků, které s třídou nesouvisí [36].

5.2.4 Rozhodovací stromy

Rozhodovací stromy vycházejí ze stromových grafů, kde každý uzel reprezentuje rozhodovací kritérium, které určuje další směr v procesu rozhodování. Při predikci model sleduje cestu od kořene k listům stromu, přičemž listové uzly poskytují konečnou klasifikaci [31]. Typicky jsou rozhodovací stromy konstruovány jako binární kořenové stromy. Na obrázku 5.3 je ukázán příklad rozhodovacího stromu, který na základě barvy auta, roku výroby a modelu predikuje, zda je vhodné vozidlo koupit nebo ne.



Obrázek 5.3: Graf rozhodovacího stromu pro určení výhodnosti koupě auta

Během tréninkového procesu algoritmus postupně konstruuje rozhodovací strom od kořene k listům, přičemž dává přednost atributům s nejvyšším informačním ziskem [2]. Tímto způsobem vybírá takové příznaky, které efektivně rozdělují data do jednotlivých tříd.

Tento přístup však vede k přetrénování – nadměrnému přizpůsobení stromu na trénovací data [3]. To znamená, že klasifikátor bude přesně reflektovat vzory v těchto datech, ale může selhávat v případě nových, odlišných dat.

V praxi se tak často omezuje hloubka stromu, aplikují se další techniky, jako je prořezávání uzlů s nízkou informační hodnotou nebo se nejčastěji využívají soubory rozhodovacích stromů [31, 2].

Soubory rozhodovacích stromů

Problémy s přetrénováním rozhodovacích stromů jsou natolik závažné, že se často upřednostňují kombinace rozhodovacích stromů, které tvoří model strojového učení [2, 61]. Tyto přístupy, které kombinují několik modelů do jednoho, se nazývají ensemble metody, a jejich využití s rozhodovacími stromy budu dále rozebírat.

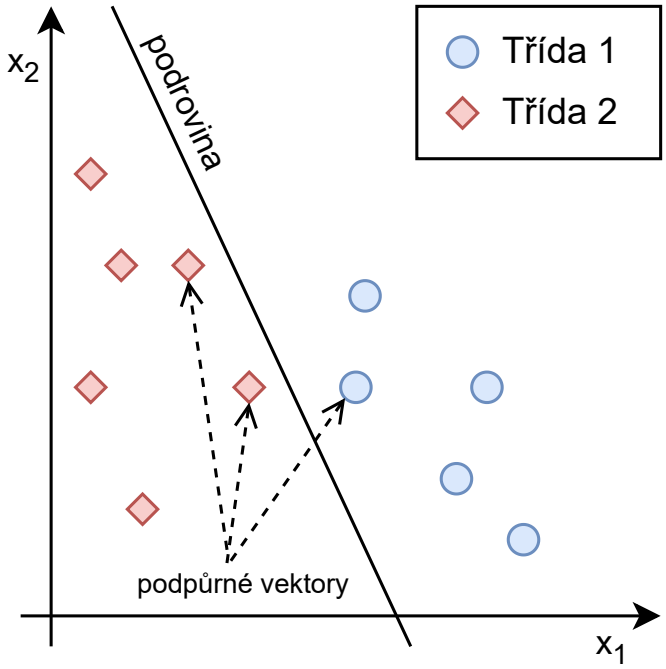
Přístup náhodných lesů, v anglickém jazyce Random forest a takto i nadále v této práci, paralelně vytváří mnoho rozhodovacích stromů, přičemž každý strom zpracovává náhodný výběr příznaků [31]. Do každého stromu je vnesen prvek náhody, což zaručuje odlišnost každého stromu. Ačkoli každý strom může být přetrénovaný, jejich kombinace dokáže problém dostatečně generalizovat [3, 31].

Boosting algoritmy rovněž vytvářejí více stromů, avšak postupují sériově, pomocí tzv. boosting iterací, kde každý následující strom se zaměřuje na opravu chyb svého předchůdce [2, 8]. Metoda, jakou se stromy postupně zlepšují, závisí na konkrétním použitém algoritmu. Mezi běžně používané algoritmy patří například AdaBoost, XGBoost a další.

5.2.5 Algoritmus podpůrných vektorů

Algoritmus podpůrných vektorů, anglickou zkratkou SVM a takto i nadále v této práci, klasifikuje data tím, že vyhledává nadrovinu sloužící jako hranice pro rozlišení tříd. Toho dosahuje pomocí transformace datových prvků do prostoru s vyšší dimenzí, kde je možné nalézt efektivnější oddělovací hranici [31, 2]. Podpůrné vektory, po kterých je metoda pojmenována, jsou prvky ležící nejblíže k hranici mezi třídami a mají klíčovou roli v definování nadroviny [3].

Na obrázku 5.4 jsou zobrazeny prvky dvou tříd, jejich podpůrné vektory a klasifikační podrovina, která je odděluje.



Obrázek 5.4: Vizualizace rozhodovací hranice v 2D prostoru

Metoda SVM je vhodná pro práci s velkým počtem příznaků [2, 8], avšak narazí na obtíže při zpracování velkých datových sad, což vede k vysokému počtu podpůrných vektorů a tím zvyšuje výpočetní náročnost [31].

5.3 Zpracování textových dat

Oblast strojového učení, která se specializuje na analýzu mluveného a psaného jazyka, se nazývá zpracování přirozeného jazyka [5], anglicky *natural language processing*. Ačkoliv textová data z HTTP požadavků na první pohled mohou připomínat přirozený jazyk, ob-

sahují spíše izolovaná slova než smysluplné věty. Přesto tato oblast představuje hlavní zdroj informací pro zpracování textových dat.

Algoritmy strojového učení obvykle pracují pouze s numerickými hodnotami [6, 8], což vyžaduje převod textových řetězců. Tento proces obvykle zahrnuje dvě fáze: tokenizaci a vektorizaci. V následujících podsekcích se oběma technikám budu věnovat podrobněji.

5.3.1 Tokenizace textových řetězců

Tokenizace je proces rozdělování textových řetězců na menší jednotky, známé jako tokeny [5]. V oblasti zpracování přirozeného jazyka obvykle tokeny odpovídají slovům. Jak jsem zmínil na začátku této sekce, pro účely této práce je třeba jít nad rámec běžné tokenizace podle slov a aplikovat například metodu n-gramů.

N-gramy

N-gramy jsou definovány jako „posloupnosti n jednotek stejného druhu“ [37], jako jsou slova nebo znaky. V oblasti zpracování přirozeného jazyka se používají k zachycení kontextu v rámci vět. Například při analýze souvětí „Je to dobré, vůbec to není špatné.“ a „Je to špatné, vůbec to není dobré.“ by přístup založený pouze na jednotlivých slovech obě věty vyhodnotil jako shodné [2], i přesto, že jejich význam je přesně opačný.

S ohledem na data této bakalářské práce se vybízí tvorba n-gramů podle znaků. Například textový řetězec „Pes“ by pomocí bigramů, tedy n-gramů o dvou znacích, byl reprezentován jako: „Pe“ a „es“. Aplikaci metody n-gramů na datovou sadu této bakalářské práce a dalším příkladům serializace se věnuji v sekci 6.2.2.

5.3.2 Vektorizace tokenů

Vektorizace textu představuje proces převodu textových dat na numerické hodnoty [8]. Metody vektorizace se liší v závislosti na konkrétním použití a sahají od jednoduchého počítání výskytů jednotlivých tokenů až po pokročilé reprezentace slov pomocí vektorů, jak to demonstruje algoritmus *word2vec*[1].

V této bakalářské práci se experimentuje s dvěma přístupy k vektorizaci textu: prvním přístupem je počítání výskytů jednotlivých tokenů, v anglické literatuře pod pojmem *bag-of-words*, případně zkratkou BoW a takto i nadále v této práci, zatímco druhým je využití metody TF-IDF. Ta slouží k vyjádření důležitosti tokenu ve specifickém textovém řetězci s ohledem na všechny ostatní [2], což umožňuje lépe zachytit významnost jednotlivých slov v kontextu celé datové sady.

5.4 Proces trénování algoritmů

Proces trénování modelů strojového učení je systematický a zahrnuje několik iterativních kroků, které se opakují v reakci na nové poznatky nebo neuspokojivé výsledky. Každý krok je zásadní a přispívá k postupnému zdokonalování výsledného modelu. Obrázek 5.5 tento iterativní vývoj zobrazuje. Jednotlivé kroky jsou popsány níže:

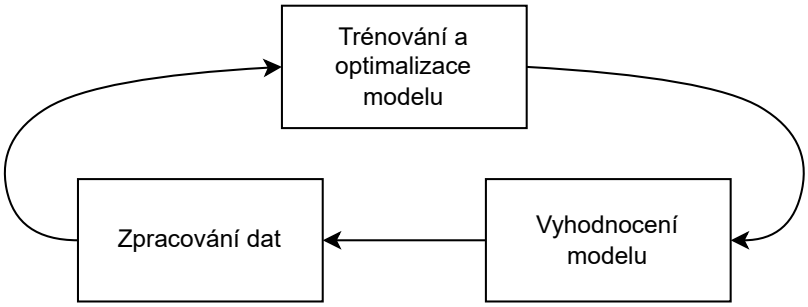
1. **Příprava dat:** V této fázi jsou data sbírána, normalizována a případně filtrována, například od okrajových hodnot. Extrahují se také příznaky z vstupních dat, v poz-

[1] https://arxiv.org/pdf/1301.3781

dějších iteracích zahrnuje i výběr těch relevantních pro daný cíl. Vznikem datové sady, její normalizací a analýzou společných vlastností se věnuji v kapitole 6.

2. **Trénování a optimalizace modelu:** Během tohoto procesu algoritmus hledá vzory ve vstupních datech a hledá mezi nimi vzory pro splnění zadaného cíle. Způsob, jakým toto dělá, závisí na zvoleném algoritmu, viz sekce 5.2. V pokročilejších iteracích může zahrnovat i ladění hyperparametrů pro optimalizaci výkonu, o nichž se zabývá sekce 5.4.2.

3. **Vyhodnocení modelu:** Konečný výkon modelu je vyhodnocen pomocí relevantních metrik, které odpovídají vstupním datům a cílům problému. Tento krok pomáhá posoudit efektivitu a účinnost vytvořeného modelu a věnuji se mu v kapitolách 7 a 8.



Obrázek 5.5: Obecný iterativní proces vývoje modelu strojového učení

5.4.1 Křížová validace

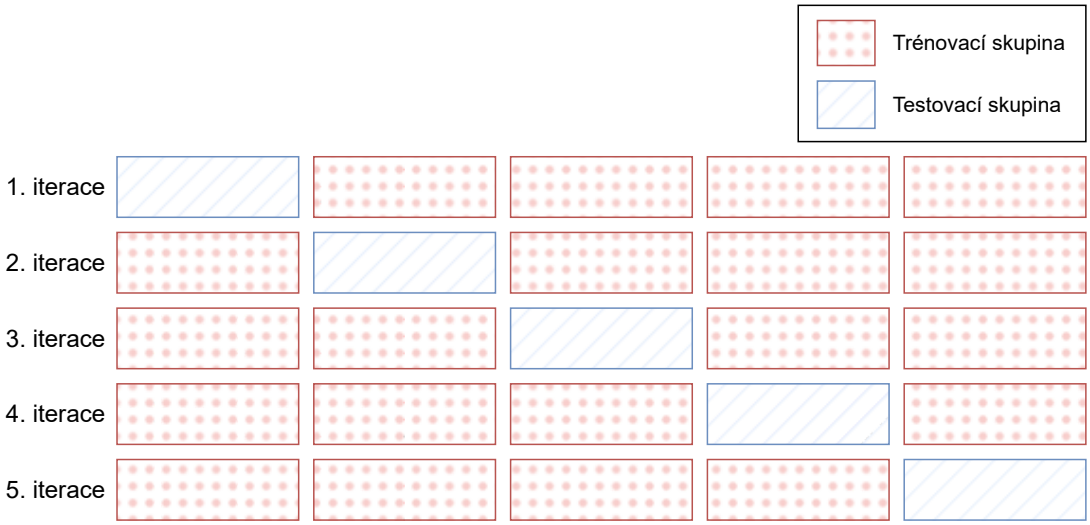
Křížová validace, anglicky *cross-validation*, slouží k ověření generalizační schopnosti modelů strojového učení [2]. Nejčastěji se využívá metoda k-násobné křížové validace, při které jsou trénovací data rozdělena do k skupin. Model je poté trénován na k - *1* z těchto skupin a testován na zbývající, dosud nevyužité skupině. To umožňuje modelům ověřit svou přesnost na nových datech a zároveň poskytuje náhled do jejich schopnosti generalizovat na nová, dosud neznámá data.

5.4.2 Ladění hyperparametrů

Hyperparametry algoritmu mírně upravují proces trénování, jsou zvoleny před zahájením procesu trénování [54] a liší se podle zvoleného algoritmu. Tyto parametry jsou klíčové pro zamezení přetrénování, kdy model nedokáže efektivně generalizovat na nová data, nebo případů, kdy model není dostatečně přesný ani na trénovací data. Hyperparametry tedy určují, jak konzervativní, tedy jednoduchý nebo naopak složitý výsledný model bude.

Úprava hyperparametrů může významně ovlivnit výkonnost modelu strojového učení, a proto je proces optimalizace, známý jako ladění, zásadní. Vzhledem k rozsáhlému počtu hyperparametrů a jejich variabilitě je vhodné tento proces automatizovat [8, 2].

Jednou z běžně používaných metod pro automatické ladění je metoda hledání v mřížce [2], anglicky *grid-search*. V této metodě uživatel specifikuje sadu možných hodnot pro každý hyperparametr a algoritmus je následně testován všemi možnými kombinacemi těchto hodnot.



Obrázek 5.6: Příklad rozdělení dat při 5násobné křížové validaci

Pro zajištění robustnosti výsledků je doporučeno kombinovat tuto metodu s křížovou validací [2], jež je popsána v sekci 5.4.1. Při každé kombinaci hodnot tak dochází k opakovanému testování na různých datech, čímž se minimalizuje riziko náhodně dobrých výsledků způsobených výběrem konkrétní testovací sady. Aplikací obou těchto metod se věnuji v sekci 7.

5.4.3 Metriky pro vyhodnocení validity modelu

Pro účinné hodnocení finálního modelu strojového učení je zásadní vybrat odpovídající statistické metriky, které jsou specifické pro daný problém [2]. Každá metrika nabízí odlišný úhel pohledu na kvalitu modelu, a proto je doporučeno v hodnocení využívat jejich kombinaci pro komplexní posouzení.

Před samotným výčtem jednotlivých metrik, které budou aplikovány při evaluaci klasifikátoru této bakalářské práce, je nejprve potřeba objasnit některé základní statistické termíny:

- **TP** – Počet datových záznamů správně identifikovaných jako pozitivní (maligní).
- **TN** – Počet datových záznamů správně identifikovaných jako negativní (benigní).
- **FP** – Množství záznamů chybně klasifikovaných jako pozitivní.
- **FN** – Množství záznamů chybně klasifikovaných jako negativní.

Metriky vhodné pro vyhodnocení klasifikátoru jsou následující [31, 38]:

- **Přesnost (accuracy)** – Poměr správně klasifikovaných záznamů ke celkovému počtu záznamů v testovací sadě, jak je ukázáno v rovnici (5.3):

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.3}$$

- **Preciznost (precision)** – Specifikuje poměr správně klasifikovaných pozitivních záznamů vůči všem záznamům, které byly klasifikovány jako pozitivní. Rovnice pro výpočet je zobrazena v rovnici (5.4)

$$precision = \frac{TP}{TP + FP} \tag{5.4}$$

- **Úplnost (recall)** – Podíl správně identifikovaných pozitivních záznamů ke všem skutečně pozitivním záznamům, viz rovnice (5.5). V kontextu této práce je tato metrika obzvlášť důležitá, jelikož právě chybně označená maligní textová data mohou využít nějaké zranitelnosti. Naopak, klasifikace benigních zpráv jako maligních nevykazuje žádné bezpečnostní riziko.

$$recall = \frac{TP}{TP + FN} \tag{5.5}$$

- **Skóre F1 (F-score)** – Harmonický průměr dvou předchozích metrik, úplnosti a preciznosti, je zvláště klíčový pro nevyvážené datové sady, protože poskytuje kvalitativní hodnocení, které reflektuje tuto nerovnováhu. Výpočet tohoto průměru je znázorněn v rovnici (5.6) níže:

$$F\text{-}score = \frac{2 * precision * recall}{precision + recall} \tag{5.6}$$

- **Váhovaná přesnost (weighted accuracy)** – Udává poměr správně klasifikovaných záznamů k celkovému počtu záznamů se stejnou třídou. Tento ukazatel zohledňuje možnou nerovnováhu mezi třídami a tento přístup je aplikovatelný i na další dříve zmíněné metriky, nicméně kvůli stručnosti nejsou zde tyto metriky podobně definovány. Váhová přesnost pro binární klasifikaci je definována vzorcem (5.7):

$$weighted\_accuracy = \frac{1}{2} \times \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \tag{5.7}$$

Kapitola 6

Datová sada

V sekci 2.2 jsem naznačil důležitost vhodné datové sady pro úspěšný vývoj modelů umělé inteligence. François Chollet ve své knize Hluboké učení v jazyku Python [8] doporučuje, aby, převzato do kontextu této práce, datová sada obsahovala dostatečné množství maligních i benigních požadavků, zachycující co nejširší spektrum možných variant. Nedostatečná variabilita těchto informací by mohla vést k situaci, kdy výsledný model umělé inteligence dokáže správně klasifikovat známé útoky, ale selhává při identifikaci i nejmenších modifikací těchto nežádoucích požadavků.

Díaz-Verdejo a kol. [20] definují problémy s datovými sadami standardně využívané pro tvorbu klasifikátorů detekující nežádoucí požadavky. Zmiňují, že tyto sbory informací nejsou pro kategorizaci vůbec vhodné nebo dostatečně obecné. Často zachycují provoz na nízkem počtu webových aplikací a není možné tak kvalitně určit obecnost klasifikátoru, který takovou datovou sadu využil.

Mimo jiné se zmíněná práce zabývá i anotováním maligních a benigních požadavků, jež může být časově ale i kvalitativně vytižující. Zpravidla jsou datová sady vytvořeny z monitorování webového provozu, během kterého jsou zachyceny i zprávy HTTP využívající nějaké zranitelnosti. Takové požadavky je nutné označit jako nežádoucí. Vzhledem k tomu, že se typicky jedná o manuální práci, tak může vést k nedokonalým výsledkům, které ovlivní kvalitu výsledného klasifikátoru.

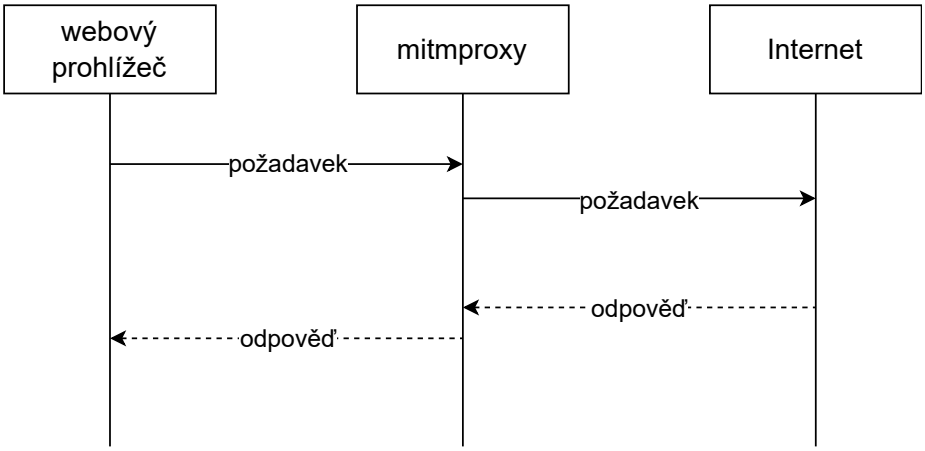
6.1 Vznik datové sady

Vzhledem k problémům uvedeným výše jsem se rozhodl pro potřeby této práce vytvořit vlastní anotovanou datovou sadu. Tato sada by měla zahrnovat široký rozsah jak žádoucích, tak nežádoucích požadavků, být co nejuniverzálnější a správně anotovaná. Kromě toho plánuji využít i volně dostupné datové sady zaměřené na zranitelnosti webových aplikací [62], abych předešel jakýmkoli případným nepříznivým vzorům v mé metodice sběru dat. Proces vytváření datové sady jsem rozdělil do dvou fází: sběr benigních a maligních požadavků.

6.1.1 Sběr legitimních požadavků

Pro sběr legitimních požadavků jsem se rozhodl sledovat vlastní síťový provoz na Internetu. Tato strategie zajistí přesnou anotaci, protože je málo pravděpodobné, že by webový prohlížeč generoval jakékoliv nežádoucí požadavky. Zvolený přístup také umožňuje dostatečnou variabilitu požadavků, bude-li monitorování provozu probíhat dlouhodobě.

Sledování provozu zajišťuje reverzní proxy server mitmproxy [16] speciálně zaměřený pro pozorování webového provozu, jehož implementace umožňuje širokou škálu konfigurace. Součástí této aplikace je i nástroj pro příkazovou řádku mitmdump, který umožňuje provést libovolný kód pro každou událost, v mém případě každý zachycený požadavek. Této funkcionality jsem využil pro uložení každého požadavku HTTP jako soubor v jeho čiré textové formě, která zachycuje kompletní informace, jež budou využity pro vývoj modelu pro klasifikaci požadavků. Princip komunikace je znázorněn v obrázku 6.1.



Obrázek 6.1: Schéma HTTP komunikace s využitím mitmproxy

6.1.2 Sběr nelegitimních požadavků

Útočníci často využívají penetrační testovací nástroje k objevení zranitelností. Tyto aplikace umožňují automatizované provedení různých útoků na cílovou webovou aplikaci a zahrnují různé techniky a databáze zranitelností, což umožňuje i uživatelům bez hlubších znalostí tyto útoky provádět.

Různé penetrační testovací nástroje se obvykle zaměřují na konkrétní typ zranitelnosti a často disponují vlastními seznamy zneužití. Pro zajištění co nejkomplexnější znalosti daných zranitelností je vhodné využít co nejširší spektrum těchto programů.

Pro účely sběru nežádoucích požadavků jsem naimplementoval jednoduchou webovou aplikaci, která na každý požadavek HTTP odpoví zprávou s úspěšným stavovým kódem, který byl empiricky zvolen. Tyto odpovědi pro účely této práce nejsou důležité a proto jejich podoba není podstatná. Nelegitimní požadavky ukládám stejně jako ty legitimní, čili jako soubor v čiré textové formě.

Injekce SQL

Sqlmap [29] je software s volně šiřitelným zdrojovým kódem, který automatizuje proces detekce a zneužití útoku injekce SQL. Nabízí širokou paletu možností manipulace s daty pomocí patřičných parametrů, obsahuje útoky pro různé databázové systémy a podporuje šest technik injekce SQL.

Cross-site scripting

Pro sběr požadavků se zaměřením na útoky typu XSS byl využit nástroj XSStrike [59]. Tento nástroj disponuje vlastní databází znalostí a umožňuje penetrační testování i pomocí uživatelem definovaných datových sad. Oproti konkurenčním volně dostupným nástrojům podporuje několik technik pro provedení útoku typu XSS [60].

6.1.3 Zpracování veřejných datových sad

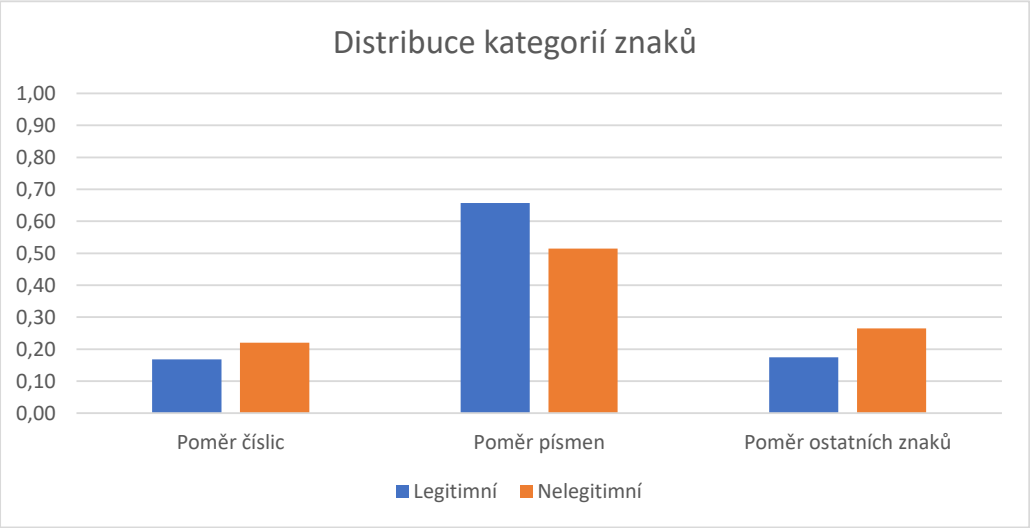
Jak jsem již naznačil v úvodu této kapitoly, datová sada této bakalářské práce byla rozšířena o veřejně dostupné datové sady [62]. Ty neobsahují požadavky HTTP v textové formě, ale pouze textové řetězce doplněné o anotaci přítomnosti nebo nepřítomnost zneužití zranitelnosti. Každá datová sada měla svůj vlastní formát, který jsem sjednotil pro jednodušší práci s daty. Data z veřejných zdrojů tvoří téměř 85 % celkového objemu datové sady použité v této bakalářské práci.

6.2 Analýza společných vlastností

Jak jsem již zmínil v kapitole 5, z datové sady je potřeba extrahovat příznaky relevantní pro klasifikaci. V této sekci se zabývám analýzou datové sady a studiem takových vlastností, které mohou být zajímavé pro klasifikaci.

Analýzou průměrných a mediánových délek textových řetězců je patrné, že i když průměrné délky mezi maligními a benigními texty jsou si podobné, mediány délek mezi nimi vykazují významné rozdíly. Pro legitimní skupinu je medián délek 9, zatímco pro maligní skupinu je to 64.

Pohled na distribuce kategorií znaků odhalil, že legitimní hodnoty obvykle obsahují vyšší počet písmen, zatímco nelegitimní hodnoty mají větší zastoupení ostatních znaků, včetně závorek, mezer a podobně. Tyto poznatky jsou vizualizovány na grafu 6.2.



Obrázek 6.2: Distribuce kategorií znaků v textových řetězcích

Při zkoumání frekvence jednotlivých znaků v nelegitimních požadavcích vyniká především výskyt znaku procenta, který se vyskytuje ze všech znaků nejčastěji. Tento trend

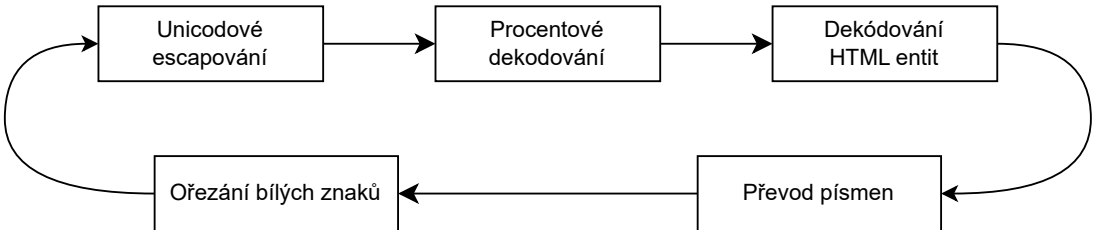
lze pravděpodobně připsat používání procentového kódování v komponentě query 3.1.1, jež podle specifikace URI umožňuje bezkonfliktní vkládání rezervovaných znaků. Standard nicméně umožňuje také kódování dalších znaků, což může být útočníky využito k maskování vzorů běžně spojovaných s útočnými aktivitami. Normalizaci textových řetězců včetně zavedení procentového dekódování se podrobně věnuji v následující podsekci 6.2.1.

6.2.1 Normalizace textových řetězců

Vstupní data pro algoritmus je vhodné standardizovat a normalizovat, aby byla ve stejném formátu, což umožňuje efektivnější zpracování [8, 6, 5]. U numerických hodnot to obvykle zahrnuje transformaci do rozsahu od 0 do 1. V případě textových řetězců se často doporučuje převést všechna písmena na malá [5]. Pro data relevantní k této bakalářské práci se vybízí provést i procentové dekódování, o jehož výskytu jsem již v této sekci psal.

Implementace zmíněných úprav ovlivnila všechny sledované metriky z této sekce v jednotkách procentních bodů. Během manuální inspekce jsem identifikoval další způsoby kódování znaků, jako je unicodové escapování, využití entit HTML a bajtových řetězců. Použití těchto metod však samo o sobě neindikuje přítomnost útoku, protože mají i legitimní využití – například již zmíněné zahrnutí rezervovaných znaků v komponentě URI nebo pro řešení problémů s kódováním textových řetězců.

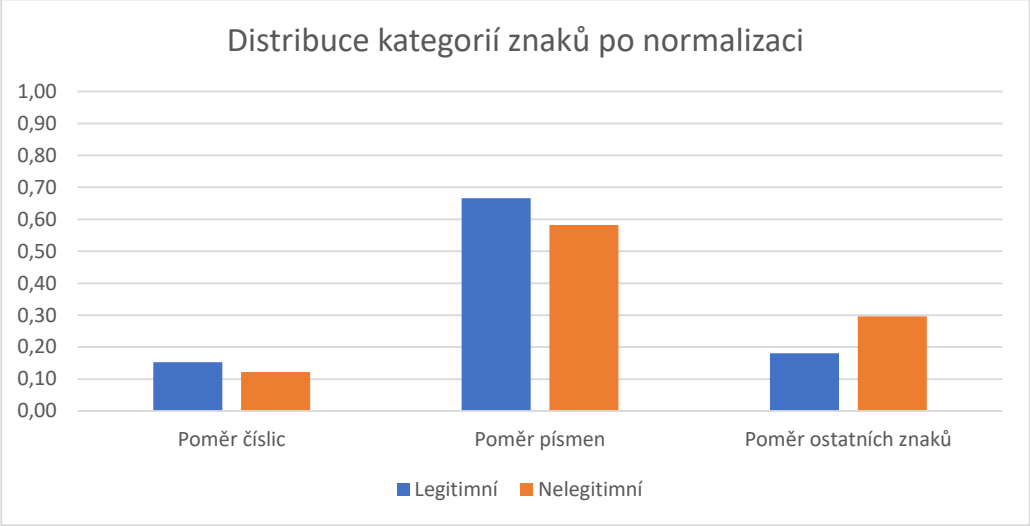
Všechny zmíněné kódovací techniky jsou převedeny na odpovídající znaky v pevně stanoveném, empiricky zvoleném pořadí, dokud ve vstupním textovém řetězci nezůstanou žádné přeložitelné kódovací sekvence. Když normalizační algoritmus narazí na kódovací sekvenci odpovídající netisknutelnému znaku, ignoruje ji a nezahrnuje do normalizace. Dále jsou na krajích textových řetězců odstraněny bílé znaky a celý text je převeden do malých písmen. Sekvence těchto kroků je ilustrována na přiloženém obrázku 6.3.



Obrázek 6.3: Proces normalizace textových řetězců

Odstranění kódovacích sekvencí

Zavedení těchto technik vedlo ke snížení počtu unikátních textových řetězců na zhruba 230 tisíc, z toho 122 tisíc tvoří legitimní hodnoty. Drobným změnám došlo i ve sledovaných vlastnostech, ale i přesto nelegitimní hodnoty stále obsahují vyšší počet speciálních znaků. Pro hodnoty zneužívající zranitelnosti injekce SQL jsou časté znaky kulatých závorek, čárky a středníku. Útok XSS je charakteristický hojným výskytem lomených závorek, uvozovek a znaku plus. Nová distribuce kategorií znaků v normalizovaných textových řetězcích je zobrazena na grafu 6.4.



Obrázek 6.4: Distribuce kategorií znaků v normalizovaných textových řetězcích

6.2.2 Využití n-gramů

Při analýze nejčetnějších n-gramů o velikostech 2, 3 a 4, o nichž jsem již psal v předešlé sekci 5.3.1, lze pozorovat značné rozdíly mezi maligními a benigními texty.

Bigramy v maligních textech často obsahují sekvence jako „..“, „./“ a „/.“, které jsou typické pro útok path traversal, diskutovaný v podsekci 4.1.3. Dále jsou patrné bigramy „le“ a „se“, které mohou být spojeny s útoky injekce SQL 4.1.1 a XSS 4.1.2, neboť pocházejí z anglických slov *select* a *alert*, která jsou běžně používána v obou typech útoků.

Legitimní texty obvykle zahrnují bigramy jako „":“ a „,"“, což pravděpodobně vychází ze zahrnutí těl požadavků s objekty JSON do datové sady. Obsahují také slabiky jako „er“, „re“ a „on“, které patří mezi nejčastěji používané v anglickém jazyce [63].

U trigramů, tedy n-gramů o velikosti 3, jsou výrazné kombinace „ull“, „nul“ tvořící slovo *null*, jež je relevantní jak pro jazyk SQL, tak pro JavaScript. Opět jsou zřetelné vzory pro útok path traversal, jako „./.“, „...“ a další.

Mezi legitimními 4-gramy převažuje výskyt „http“ a „ttps“, což pravděpodobně reflektuje hodnoty v hlavičkách *Referer* a *Origin*. Ty slouží k uvedení adresy URL momentálně navštívené stránky. Nelegitimní texty, kromě slova „null“, často obsahují také segmenty slova *script*, jako jsou „ript“, „crip“ a „scri“.

N-gramy umožnily odhalit specifické charakteristiky různých útoků, což poskytlo hlubší vhled do jejich vzorů. Při testování na datové sadě lze pozorovat, že existují výrazné rozdíly v frekvenci výskytu jednotlivých n-gramů mezi legitimními a nelegitimními textovými řetězci. Analýze, jaké n-gramy jsou relevantní pro klasifikaci požadavků, se věnuji v sekci 7.2.

6.3 Datový model

V rámci bakalářské práce bylo shromážděno přes 21 tisíc nelegitimních a téměř 12 tisíc legitimních požadavků HTTP. Z těchto dat jsem extrahoval klíčové prvky, kde se útoky typicky skrývaly: hodnoty hlaviček, query parametrů a bylo-li to možné i z těla požadavků.

Spolu s daty získanými z volně dostupných zdrojů na Internetu datová sada čítá 240 tisíc unikátních textových řetězců, kde bezmála 120 tisíc patří do nelegitimní skupiny.

Po normalizaci datové sady, zmíněné v sekci 6.2.1, klesl počet unikátních hodnot na zhruba 220 tisíc, přičemž téměř 108 tisíc patří do maligní skupiny. Vlastní sesbíraná data tvoří zhruba 16 % celkového objemu datové sady, zatímco zbytek představují textová data z veřejně dostupných zdrojů. Tyto informace, společně s poměrem benigních a maligních textových dat, jsou prezentovány v tabulce 6.1 níže. Z této tabulky také vyplývá mírná disbalance tříd, která se musí zohlednit při experimentech popsaných v kapitole 7.

Zdroj	**Legitimní**	**Nelegitimní**	**Poměr**
Vlastní	16 893	20 270	0.4545
Z jiných zdrojů	96 935	87 441	0.5257
Celkem	113 828	107 711	0.5283

Tabulka 6.1: Přehled poměru legitímních a nelegitímních textových dat v datové sadě

Datová sada zahrnuje všechny útoky popsané v sekci 4.1 a díky zařazení relevantních veřejně dostupných datových sad může obsahovat i další, nespecifikované typy útoků. Pro zjednodušení anotace a vývoje klasifikátoru jsou záznamy kategorizovány pouze jako legitimní nebo nelegitimní, což odpovídá problému binární klasifikace.

Kapitola 7

Tvorba klasifikátoru webových požadavků

V sekci 5.2 se zaměřuji na algoritmy strojového učení určené pro klasifikační úlohy. Není možné předem jednoznačně určit, který algoritmus bude pro data této bakalářské práce nejúčinnější, a proto se tato sekce věnuje iterativnímu vývoji, popsaného v sekci 5.4, všech zmíněných algoritmů.

Klasifikátor by měl být schopen efektivně rozlišovat mezi legitimními a nelegitimními webovými požadavky. Pro tento účel byla sestavena datová sada, která zahrnuje všechny typy útoků popsané v sekci 4.1. Navíc, díky metodám strojového učení, klasifikátor nabízí potenciál pro identifikaci i jiných útoků, které nejsou v této sadě explicitně uvedeny.

Pro realizaci experimentů byl vybrán programovací jazyk Python, který je široce užívaný v oblasti strojového učení, zejména díky knihovně jako `scikit-learn`, která poskytuje implementace téměř všech algoritmů specifikovaných v této práci a pomocné funkce pro statistické zhodnocení.

7.1 Selekce klasifikačního algoritmu

Z textových dat jsem extrahoval celkem 12 atributů, z nichž některé jsou detailně popsány v sekci 6.2 této bakalářské práce. Ostatní atributy jsou zařazeny nově, jelikož jejich přínos pro model je dopředu těžko ověřitelný a jejich relevantnost je potřeba ověřit experimenty. Prvních 8 příznaků je přeškálováno do rozsahu 0 až 1, zatímco zbylé vlastnosti jsou již přeškálované. Tyto příznaky zahrnují:

- počet číslic, písmen, bílých a ostatních znaků,
- délka textového řetězce,
- Shannonovu entropii, vyjádřující informační kvalitu textového řetězce,
- počet provedených normalizačních technik popsaných v sekci 6.2.1,
- nejdelší počet stejného písmene,
- poměr číslic, písmen, bílých a ostatních znaků vzhledem k celkové délce textového řetězce.

Dále jsem z textových dat extrahoval unigramy, bigramy a trigramy. Pro vektorizaci jsem použil metody BoW a TF-IDF, viz sekce 5.3.2. Vyextrahováno bylo přes 150 tisíc n-gramů a již dopředu je patrné, že pro klasifikaci bude relevantní pouze hrstka.

7.1.1 Porovnání výsledků algoritmů

Všechny algoritmy byly trénovány s využitím až[1] 5násobné křížové validace a metody hledání v mřížce pro optimalizaci hyperparametrů s maximalizací metriky váhovaného skóre F1. Kombinací těchto dvou strategií se zabývá sekce 5.4.2.

Během jednotlivých iteracích byly vzhledem k nevyváženosti datové sady shromažďovány a průměrovány váhované metriky jako přesnost, preciznost, úplnost a skóre F1. Kromě toho byla zaznamenávána i míra falešně negativních výsledků, dále označovaná jako FNR z anglického spojení *false negative rate*, viz sekce 5.4.3. Tato metrika udává poměr maligních požadavků nesprávně anotovaných jako benigní a tedy ukazuje kolik nelegitimních požadavků uniklo detekci klasifikátorem a mohlo by potenciálně využít zranitelnosti na webovém serveru. Naopak nesprávná anotace legitimního požadavku jako maligního nemá na bezpečnost serveru vliv.

Výsledky testování jsou uspořádány dle metriky váhovaného skóre F1 a jsou prezentovány v tabulce 7.1. Tabulka obsahuje seznam použitých algoritmů spolu s váhovanými metrikami přesnosti, preciznosti, úplnosti a skóre F1, mírou FNR, velikostí datové sady a použitou metodou vektorizace n-gramů. Pro stručnost jsou pro každou metodu vektorizace u každého algoritmu uvedeny pouze ty nejlepší výsledky podle váhovaného skóre F1. Sloupec VDS je zkratkou pro velikost datové sady a vedle MV pro metodu vektorizace.

Algoritmus	**Preciznost**	**Úplnost**	**Skóre F1**	**Přesnost**	**FNR**	**VDS**	**MV**
XGBoost	99.50 %	99.50 %	99.50 %	99.50 %	0.61 %	221 540	BoW
XGBoost	99.45 %	99.45 %	99.45 %	99.44 %	0.76 %	221 540	TF-IDF
Log. regrese	99.17 %	99.17 %	99.17 %	99.16 %	1.10 %	221 540	TF-IDF
SVM	99.04 %	99.04 %	99.04 %	99.03 %	1.10 %	50 000	TF-IDF
Log. regrese	98.77 %	98.77 %	98.77 %	98.75 %	1.18 %	221 540	BoW
SVM	98.26 %	98.25 %	98.24 %	98.23 %	2.79 %	50 000	BoW
KNN	97.87 %	97.87 %	97.87 %	97.86 %	2.36 %	20 000	TF-IDF
Random forest	97.26 %	97.19 %	97.19 %	97.12 %	4.87 %	221 540	BoW
Random forest	97.23 %	97.14 %	97.14 %	97.07 %	5.14 %	221 540	TF-IDF
KNN	91.62 %	90.85 %	90.78 %	90.64 %	1.65 %	50 000	BoW
Naivní Bayes	93.10 %	93.00 %	93.00 %	93.06 %	4.84 %	10 000	TF-IDF
Naivní Bayes	81.72 %	75.20 %	74.07 %	75.82 %	2.47 %	10 000	BoW

Tabulka 7.1: Srovnání výkonu klasifikačních algoritmů

U některých algoritmů bylo nezbytné omezit velikost trénovací datové sady kvůli paměťovým nebo časovým omezením spojeným s trénováním. Tato omezení byla nejvýraznější u algoritmu naivního Bayese 5.2.2, jenž neumožňuje použití řídkých matic pro optimalizaci paměťové náročnosti a tak každý datový záznam měl kvůli vysokému počtu příznaků velikost přesahující 600 kB.

Modely založené na algoritmu XGBoost dominovaly v předních pozicích napříč všemi metrikami. Metoda vektorizace TF-IDF obecně dosahovala lepších výsledků, s výjimkou

[1] Počet iterací křížové validace byl stanoven na základě potřebného času k trénování a počtu hyperparametrů konkrétního algoritmu, přičemž minimálně byly provedeny dvě iterace.

modelů založených na rozhodovacích stromech 5.2.4, jako je XGBoost nebo Random forest, kde je metoda BoW mírně přesnější.

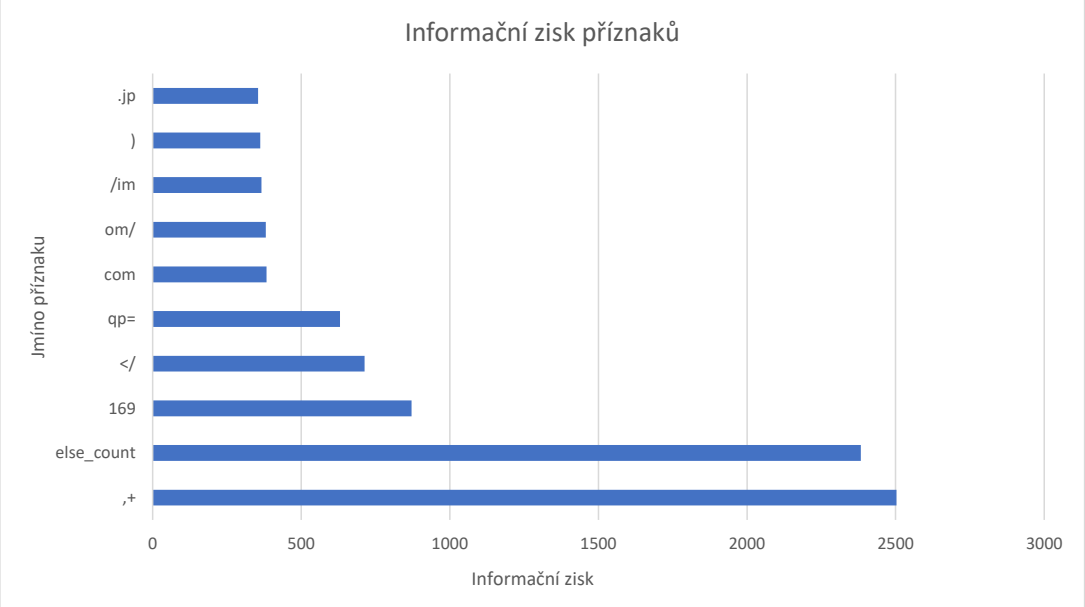
Za zmínku stojí dodat, že modely založené na algoritmu *Random forest* vykazují díky metrice FNR více než dvojnásobnou míru nesprávně anotovaných nelegitimních požadavků ve srovnání s ostatními modely s podobnými výsledky.

7.2 Analýza relevantních příznaků

Pro identifikaci nejrelevatnějších příznaků pro klasifikaci jsem využil model strojového učení, který v předešlé sekci dosáhl nejlepších výsledků. Konkrétně jde o model založený na algoritmu XGBoost s využitím metody vektorizace BoW. Pro analýzu relevance příznaků jsem využil dvou technik: zkoumání informačního zisku a využití metody SHAP.

7.2.1 Informační zisk příznaku

Informační zisk příznaku, v anglickém jazyce *feature gain*, udává, jak moc velký vliv má daný příznak na klasifikaci. Deset příznaků s největším vlivem jsou zachyceny na grafu 7.1 níže.



Obrázek 7.1: Graf příznaků s největším informačním ziskem

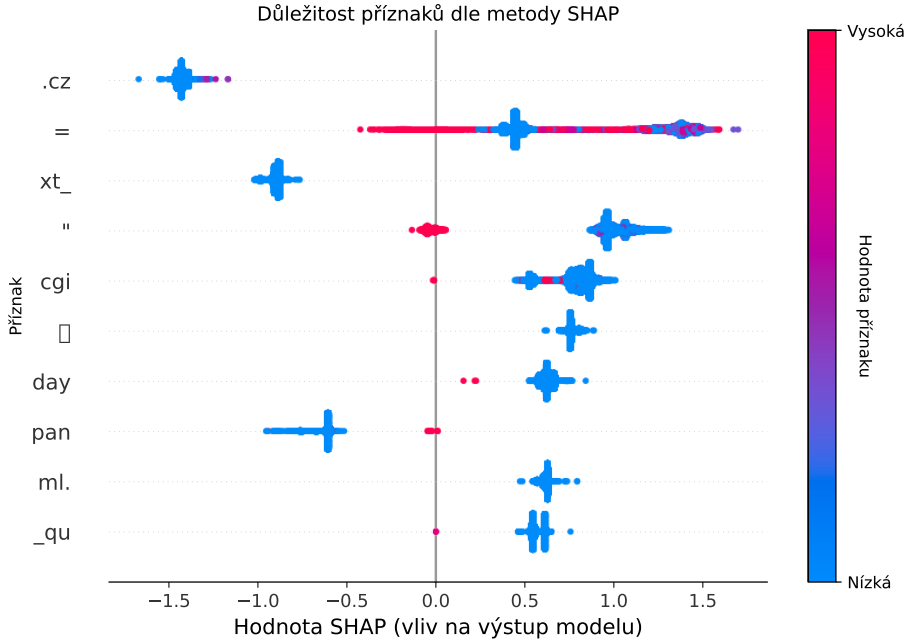
Na prvním místě se umístil příznak bigramu „,+“, jež se vyskytoval primárně v legitimních textech z veřejně dostupných sad. Na druhém místě se umístil příznak `else_count`, který udává počet speciálních znaků v textovém řetězci a o jehož motivaci jsem hovořil v sekci 6.2.

Mezi trigramy vyniká sekvence „169“, jež pravděpodobně odpovídá Unixovému času a zachycuje tak období sběru převážně legitimních dat do listopadu roku 2023, kdy poté se Unixový čas překlopil na hodnotu 1 700 000 000. Na podobném principu operují i n-gramy „.jp“ a „com“, které odpovídají doménám nejvyššího řádu.

Při sběru dat se tak projevil nežádoucí vzor v mé metodice sběru dat, na jehož možný výskyt jsem upozornil již v sekci 6.1. Za zmínku stojí i výskyt bigramu „</“, jež je zodpovědný pro útoky XSS 4.1.2 nebo unigram „)“ naopak častý u injekcí SQL 4.1.1.

7.2.2 Metoda SHAP

Metoda SHAP, zkratka pro anglický termín *SHapley Additive exPlanations*, se používá k odhalení významu jednotlivých příznaků a jejich možných vzájemných interakcí. Tato metoda, vycházející z teorie her, poskytuje důkladnější pohled na význam příznaků než tradiční analýza informačního zisku. Graf 7.2 ukazuje deset nejdůležitějších příznaků identifikovaných pomocí této metody.



Obrázek 7.2: Graf nejdůležitějších příznaků pro klasifikaci dle metody SHAP

Stejně jako u předchozí metody, i metoda SHAP odhalila problematické vzory ve sběru dat. Konkrétně se jedná o výskyt trigramu „.cz“, který odpovídá doméně nejvyššího řádu pro Českou republiku a převážně se objevuje v legitimních textech. Unigram znaku rovnítka se vyskytuje v obou skupinách, ale v nelegitimních textových řetězcích ve vyšším množství.

Zajímavý je také výskyt znaku uvozovek, který se spojuje s maligními textovými řetězci pouze při jeho nízkém počtu výskytů, zatímco vysoký počet uvozovek obvykle indikuje benigní text. To je možné vysvětlit zahrnutím velkých objektů JSON v legitimních požadavcích, které obsahují vysoké množství uvozovek.

Sekvence „cgi“, což je přípona pro soubory formátu *Common Gateway Interface*, které mohou být součástí webových serverů a zpracovávat požadavky, byla často zneužívána v nelegitimních požadavcích pro útoky hrubou silou.

7.3 Výběr nejvhodnějšího algoritmu

V předešlé sekci 7.2 jsem podrobně analyzoval nejdůležitější příznaky pro klasifikaci požadavků. V této sekci se opět zaměřím na vývoj algoritmů, ale tentokrát budu pracovat s omezeným počtem příznaků. Z původních 150 tisíc příznaků jsem vybral 997 příznaků, které měly alespoň nějaký vliv na klasifikaci – ostatní příznaky nebyly nejlepším modelem z předešlé sekce využity. Mezi těmito vybranými příznaky se nachází také dvanáct charakteristik textových řetězců.

Všechny algoritmy byly trénovány pomocí 5násobné křížové validace a metody hledání v mřížce pro optimalizaci hyperparametrů s maximalizací metriky váhovaného skóre F1, ale byla využita pouze metoda vektorizace BoW, která dosahovala v průměru lepšího zhodnocení. Výsledky testování jsou opět seřazeny podle metriky váhovaného skóre F1 a jsou zobrazeny v tabulce 7.2. Tato tabulka má téměř identickou strukturu jako tabulka 7.1 a pro stručnost není zde detailně popisována.

Algoritmus	**Prec.**	**Úplnost**	**Skóre F1**	**Přesnost**	**FNR**	**VDS**
XGBoost	99.55 %	99.55 %	99.55 %	99.55 %	0.62 %	221 540
Random forest	99.20 %	99.19 %	99.19 %	99.18 %	1.14 %	221 540
SVM	98.64 %	98.64 %	98.64 %	98.62 %	1.89 %	221 540
Log. regrese	98.34 %	98.33 %	98.33 %	98.31 %	2.44 %	221 540
KNN	97.62 %	97.62 %	97.62 %	97.61 %	2.81 %	221 540
Naivní Bayes	73.66 %	54.35 %	43.94 %	55.68 %	0.74 %	221 540

Tabulka 7.2: Srovnání výkonu klasifikačních algoritmů

Výsledky algoritmů jsou ve srovnání s experimenty popisovanými v předešlé sekci 7.2 relativně podobné. Přední pozice v rámci všech metrik byly obsazeny zástupci rozhodovacích stromů, jako jsou XGBoost a Random forest, kteří jsou zdokumentováni v sekci 5.2.4.

Je třeba zmínit zhoršený výkon algoritmu Logistické regrese, který obvykle pracuje efektivně i s velkým počtem příznaků, ale se zvoleným omezeným rozsahem příznaků nedosahuje tak dobrých výsledků. Tento jev může být způsobený výběrem příznaků, které nejsou pro tento algoritmus ideální. Aby bylo možné dosáhnout přesnějších výsledků, bylo by vhodné pro každý algoritmus extrahovat specifické, pro něj nejrelevantnější příznaky.

Naopak algoritmus KNN si polepšil o šest procentních bodů ve srovnání s předchozími výsledky používajícími stejnou metodu vektorizace BoW. Zde se nabízí také použití metody TF-IDF, se kterou KNN dosahovalo lepších výsledků v minulých experimentech. Toto by mohlo platit i pro ostatní algoritmy, kde by se mohly výsledky otočit a využití vektorizační metody TF-IDF by mohlo dospět k lepšímu zhodnocení.

Pro dosažení přesnějších výsledků by bylo vhodné určit a testovat nejrelevantnější příznaky pro každý algoritmus a metodu vektorizace zvlášť. Z časových důvodů spojených s trénováním algoritmů není toto testování v této práci provedeno. Nepředpokládám však, že by došlo k razantním změnám výsledků.

7.4 Ladění hyperparametrů algoritmu XGBoost

V obou předešlých fázích experimentů dosáhl model využívající algoritmus XGBoost nejlepších výsledků ve všech zvolených metrikách. Proto byl tento algoritmus vybrán jako základ pro vývoj finálního klasifikátoru.

Ladění hyperparametrů jsem již prováděl v obou předchozích kolech experimentů, avšak ne do takové detailní úrovně vzhledem k rychlému nárostu trénovacího času při podrobném ladění. Tento nešvar bych chtěl v této sekci odčinit a podrobně se zaměřit na ladění hyperparametrů algoritmu XGBoost s použitím stejných příznaků jako v předchozí iteraci.

7.4.1 Popis hyperparametrů algoritmu XGBoost

Algoritmus XGBoost definuje několik kategorií hyperparametrů závislé na zvoleném posilovači, anglicky *booster*. Při experimentování se nejvhodněji jeví ten s označením *gbtree* a jeho relevantní hyperparametry jsou následující:

- `eta` – Definuje faktor tlumení vah příznaků během každé iterace.
- `gamma` – Specifikuje minimální hodnotu ztrátové funkce nutné pro další rozdělení listového uzlu.
- `max_depth` – Definuje maximální hloubku rozhodovacího stromu.
- `subsample` – Udává poměr trénovacích dat použitých během boosting iterací pro trénování rozhodovacího stromu.
- `colsample_bytree`, `colsample_bylevel`, `colsample_bynode` – Tato trojice hyperparametrů určuje, jaká část příznaků je v iteraci vzorkována na úrovni stromu, úrovni uzlu a jednotlivých uzlech.
- `lambda`, `alpha` – Udávají tzv. L1 a L2 regulizační výrazy vah příznaků.
- **scale_pos_weight** – Upřesňuje poměr mezi jednotlivými anotačními třídami a tím pomáhá částečně řešit problém disbalance tříd.

7.4.2 Experimentování s hyperparametry

Ladění hyperparametrů algoritmu XGBoost s využitím vektorizační metody BoW bylo prováděno pomocí pětinásobné křížové validace a metody hledání v mřížce. Kombinací těchto dvou strategií se zabývá sekce 5.4.2. Optimalizace byla rozdělena do čtyř iterací kvůli časové náročnosti, jelikož počet kombinací prudce roste s počtem testovaných hodnot. V každé další iteraci jsem algoritmus inicializoval s nejúčinnějšími nastaveními z předchozí iterace.

V první iteraci byla testována první sada základních hyperparametrů: `eta`, `max_depth`, `gamma` a `subsample`. Následně byla zkoumána trojice parametrů spojených s vzorkováním příznaků, a to `colsample_bytree`, `colsample_bylevel` a `colsample_bynode`. Nakonec proběhla optimalizace regulizačních parametrů `lambda`, `alpha` a parametr pro balanci tříd `scale_pos_weight`. Hodnoty, s kterými byly tyto parametry testovány, jsou uvedeny v tabulce 7.3.

Nejlepší hodnoty hyperparametrů z každé iterace byly sloučeny do finální mřížky parametrů, ke kterým byly přidány další okrajové hodnoty. Ladění opět probíhalo iterativně stejně jako předešle a výsledné optimalizované hodnoty hyperparametrů jsou zobrazeny v tabulce 7.4.

Za zmínku stojí hyperparametr `scale_pos_weight`, jež má finální hodnotu 0.95 a přesně tak odpovídá poměru nerovnováhy mezi třídami, viz sekce 6.3.

Hyperparametr	**Testované hodnoty**
eta	0.1, 0.3, 0.5, 0.7
gamma	0, 0.2, 0.4, 0.6, 0.8
max_depth	8, 9, 10, 11
subsample	0.4, 0.6, 0.8, 0.9
colsample_bytree	0.7, 0.8, 0.9, 1
colsample_bylevel	0.7, 0.8, 0.9, 1
colsample_bynode	0.7, 0.8, 0.9, 1
lambda	0.1, 0.3, 0.5, 1
alpha	0.1, 0.3, 0.5, 1
scale_pos_weight	0.9, 1, 1.1

Tabulka 7.3: Tabulka hyperparametrů a testovaných hodnot

Hyperparametr	**Finální hodnota**
eta	0.30
gamma	0.40
max_depth	10
subsample	0.80
colsample_bytree	1.00
colsample_bylevel	1.00
colsample_bynode	1.00
lambda	1
alpha	0.25
scale_pos_weight	0.95

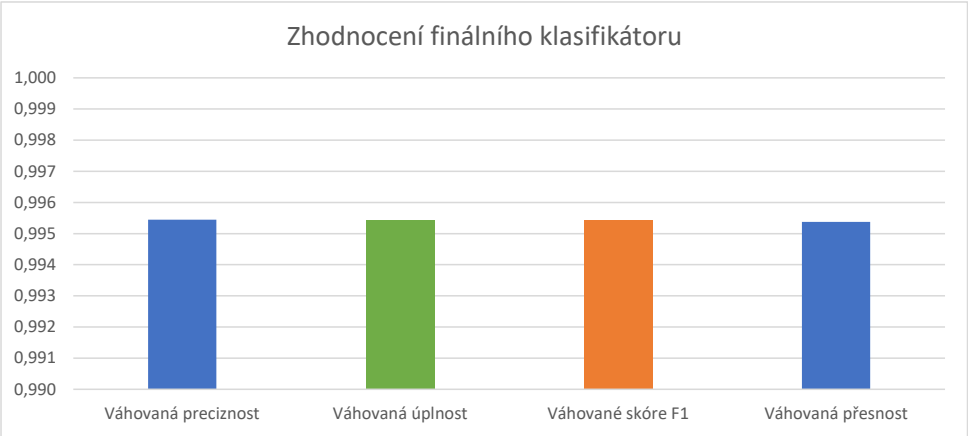
Tabulka 7.4: Tabulka výsledných hodnot hyperparametrů

Kapitola 8

Experimentální vyhodnocení klasifikátoru

V této kapitole se zabývám experimenty s finálním modelem strojového učení, který byl vyvinut v předchozí kapitole 7. Tento klasifikátor jsem otestoval pomocí 10násobné křížové validace. Výsledky jsou prezentovány na grafu 8.1, který zobrazuje váhované metriky přesnosti, preciznosti, úplnosti a skóre F1.

Míra falešně negativních výsledků (FNR) klasifikátoru je 0.6 %, což ukazuje, že pouze malý zlomek nelegitimních požadavků je mylně identifikován jako legitimní. Ostatní čtyři váhované metriky dosahují hodnot kolem 99.55 %.



Obrázek 8.1: Graf evaluace finálního klasifikátoru

Model je schopen provést klasifikaci textového řetězce, včetně jeho normalizace a vektorizace, během jednotek milisekund. Pro několik textových řetězců s variabilními délkami bylo provedeno sto opakování celého klasifikačního procesu, přičemž nebyl zjištěn významný vliv délky textu na rychlost zpracování.

8.1 Manuální inspekce validity

Při manuálním testování finálního klasifikátoru na celých požadavcích jsem zaznamenal, že model přikládá značnou důležitost počtu speciálních znaků, což vede k chybné klasifikaci běžných hodnot hlavičky *Accept*, jako je „*/*“. Tento jev také vyplývá z analýzy

relevantních příznaků v sekci 7.2. V reálném nasazení by to mohlo způsobit, že by model mylně klasifikoval velké množství legitimních požadavků jako škodlivé, kvůli specifickým hodnotám. Tento nedostatek však nemá žádný vliv na klasifikaci nelegitimních požadavků.

Jednou z možností řešení by bylo vytvoření bílé listiny obsahující tyto problematické, ale legitimní textové řetězce. Avšak udržování takové listiny by bylo komplikované, protože by pravděpodobně vyžadovala pravidelné aktualizace o nově objevující se texty.

Alternativně zvažuji možnost zahrnutí duplicitních hodnot do datové sady, což by klasifikátoru umožnilo teoreticky lépe se adaptovat a vytvořit spolehlivější profil výskytu některých n-gramů.

8.1.1 Obfuskace útoků

Při experimentování s klasifikátorem jsem objevil metodu, jak útoky efektivně skrýt. Přidáním srozumitelného textu, například pseudolatinský text Lorem ipsum, na začátek nebo konec maligního textového řetězce se útoky dokáží vyhnout detekci. Přidání náhodných sekvencí stejných znaků filtrem klasifikátoru neprojdou. Pravděpodobně díky zahrnutí příznaku Shannonovy entropie do klasifikace, viz 7.2, kdy takové řetězce mají nízkou informační kvalitu.

Pro příklad textový řetězec zneužívající injekce SQL 4.1.1: „' OR '1'='1';“ je v této formě správně klasifikován, nicméně po transformaci znázorněnou ve výpisu 8.1 se hrozba stává nerozpoznanou.

```
' OR '1'='1'; --Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
```

Výpis 8.1: Příklad obfuskace útoku injekce SQL

Vzhledem k těmto závažným zjištěním navrhuji rozčlenění vstupního textového řetězce na menší segmenty, které se budou klasifikovat postupně. Pokud se jakákoli část textu identifikuje jako nelegitimní, zpracování bude okamžitě zastaveno a celý řetězec bude označen jako maligní. Důležité je také zavést překrytí těchto textových jednotek, aby nedošlo k „přetrhání“ útoku.

Rozdělení slova „Vyhodnocení“ na textové segmenty o délce 5 znaků s překryvem 2 znaků je znázorněno na obrázku 8.2.

Obrázek 8.2: Ukázka manuálního rozdělení textového řetězce

V experimentech s hledáním optimálních obou parametrů dosahovala kombinace textových řetězců o délce 50 znaků s překrytím 5 znaků nejlepších výsledků podle všech dříve zmíněných metrik. Manipulace s textovými řetězci neměla na testování v datové sadě žádný negativní dopad a klasifikátor vykazoval stejné výsledky. Úspěšnost by však měla být u obfuskovaných textových řetězců vyšší a klasifikátor by tak měl být schopen lepší generalizace.

Kapitola 9

Závěr

Cílem této práce bylo prozkoumat aplikaci metod strojového učení k detekci nežádoucích požadavků na webu. Tento přístup vyžaduje vznik anotované datové sady, pro který jsem navrhl dva systémy, jež umožňují poloautomatické shromáždění dat.

Podrobně jsem zkoumal charakteristiky maligních a benigních požadavků, identifikoval jejich rozdíly a použil je pro klasifikaci. K zachycení detailnějších textových vzorů jsem využil n-gramy, jež umožnily nahlédnout do vzorů jednotlivých útoků a poskytnout tak informace o útocích bez žádné předešlé znalosti. Tento přístup je možný použít v budoucnu i pro nové útoky a opět zjistit jejich vzory, jež mohou být použity pro jejich detekci.

Dále jsem se věnoval normalizací textových řetězců, bez které mohou útočníci charakteristiky útoků skrýt pomocí využití technik jako procentové nebo unicodové escapování a další.

Experimentoval jsem s různými klasifikačními algoritmy, přičemž každý byl hodnocen pomocí váhovaných metrik za využití technik křížové validace a hledání v mřížce. Také jsem se věnoval podrobnému zkoumání užitečných příznaků pro klasifikaci požadavků díky využití metodě SHAP a zkoumání informačního zisku příznaků.

Finální klasifikační model používá algoritmus XGBoost, který dosáhl váhovaného skóre F1 99.55 %. Model spolehlivě klasifikuje jak legitimní, tak nelegitimní požadavky s mírou falešně negativních predikcí 0.6 %, což naznačuje, že pouze malé procento nelegitimních textů je nesprávně označeno za legitimní. Ovšem v sekci 8.1.1 se zabývám problémy, jež mohou při reálném provozu klasifikátoru nastat.

Během vývoje jsem narazil na možnosti, jak teoreticky dosáhnout lepších výsledků prostřednictvím podrobnějšího zkoumání užitečných příznaků, které však z časových důvodů spojených s trénováním algoritmů nebyly v této práci provedeny. V budoucnosti bych tyto nedostatky chtěl napravit.

Plánuji také doplnit klasifikátor o algoritmickou část pro zvýšení jeho validity. Tato část by měla díky černým listinám zabránit některým útokům zmíněných v sekci 4.1.6, ale i například o analýzu kódované řetězce `ja3`, které jsou však dostupné pouze u protokolu HTTPS. Navíc bych chtěl integrovat klasifikátor do reverzního proxy serveru, což by umožnilo jeho praktické využití jako webového aplikačního firewallu, přesunující práci z teoretické roviny do praktické aplikace.

Literatura

[1] Al Stouhi, S. a Reddy, C. K. Transfer learning for class imbalance problems with inadequate data. *Knowledge and Information Systems*. 1. vyd. Červenec 2016, sv. 48, č. 1, s. 201–228. DOI: 10.1007/s10115-015-0870-3. ISSN 0219-3116. Dostupné z: https://doi.org/10.1007/s10115-015-0870-3.

[2] Andreas C. Müller, S. G. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. 1. vyd. O'Reilly Media, Inc., září 2016. ISBN 9781449369897.

[3] Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L. et al. An Introduction to Machine Learning. *Clinical Pharmacology & Therapeutics*. 1. vyd. 2020, sv. 107, č. 4, s. 871–885. DOI: https://doi.org/10.1002/cpt.1796. Dostupné z: https://ascpt.onlinelibrary.wiley.com/doi/abs/10.1002/cpt.1796.

[4] Berners Lee, T., Fielding, R. T. a Masinter, L. M. *Uniform Resource Identifier (URI): Generic Syntax* [RFC 3986]. RFC Editor, leden 2005. DOI: 10.17487/RFC3986. Dostupné z: https://www.rfc-editor.org/info/rfc3986.

[5] Bird, S., Klein, E. a Loper, E. *Natural Language Processing with Python*. 1. vyd. O'Reilly Media, Inc., 2009. ISBN 0596516495.

[6] Burkov, A. *The Hundred-Page Machine Learning Book*. 1. vyd. Andriy Burkov, leden 2019. ISBN 9781999579517. XXX a str. 51.

[7] China, C. R. Five machine learning types to know. *IBM Blog*. 20. prosince 2023 [cit. 2024-04-13]. Dostupné z: https://www.ibm.com/blog/machine-learning-types/.

[8] Chollet, F. *Deep Learning with Python*. 1. vyd. Manning, listopad 2017. ISBN 9781617294433.

[9] Clincy, V. a Shahriar, H. Web Application Firewall: Network Security Models and Configuration. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. 2018, sv. 1, s. 835–836. DOI: 10.1109/COMPSAC.2018.00144. ISBN 978-1-5386-2667-2.

[10] Cloudflare, I. *Web application firewall (WAF) | Application security | Cloudflare*. 2023 [cit. 2023-12-11]. Dostupné z: https://www.cloudflare.com/application-services/products/waf/.

[11] Cloudflare, I. *What is a WAF? | Web Application Firewall explained | Cloudflare*. 2024 [cit. 2024-04-04]. Dostupné z: https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/.

[12] Cloudflare, I. *What is a web crawler? | How web spiders work | Cloudflare*. 2024 [cit. 2024-04-04]. Dostupné z: https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/.

[13] Corporation, M. *NVD - CVE-2023-44487*. MITRE Corporation, září 2023 [cit. 2024-01-16]. Dostupné z: https://www.cve.org/CVERecord?id=CVE-2023-44487.

[14] Corporation, N. SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? *Difference Between Supervised, Unsupervised, & Reinforcement Learning | NVIDIA Blog*. Srpen 2018 [cit. 2024-04-13]. Dostupné z: https://blogs.nvidia.com/blog/supervised-unsupervised-learning/.

[15] Corporation, N. *Machine Learning – What Is It and Why Does It Matter?* Květen 2023 [cit. 2024-01-13]. Dostupné z: https://www.nvidia.com/en-us/glossary/machine-learning/.

[16] Cortesi, A., Hils, M., Kriechbaumer, T. a contributors. *Mitmproxy: A free and open source interactive HTTPS proxy*. 2010– [cit. 2023-12-14]. [Version 10.1]. Dostupné z: https://mitmproxy.org/.

[17] DataReportal, . W. A. S. Number of internet and social media users worldwide as of July 2023 (in billions) [Graph]. *Internet and social media users in the world 2023 | Statista* [online]. 20. července 2023. 2023-10-06 [cit. 2023-10-06]. Dostupné z: https://www.statista.com/statistics/617136/digital-population-worldwide/. Path: Internet; Demographics & Use.

[18] Dawadi, B. R., Adhikari, B. a Srivastava, D. K. Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks. *Sensors*. Únor 2023, sv. 23, č. 4. DOI: 10.3390/s23042073. ISSN 1424-8220. Dostupné z: https://www.mdpi.com/1424-8220/23/4/2073.

[19] Downey, A. B. *Think Stats*. O'Reilly Media, Inc., 2011. ISBN 1449307116.

[20] Díaz Verdejo, J. E., Estepa Alonso, R., Estepa Alonso, A. a Madinabeitia, G. A critical review of the techniques used for anomaly detection of HTTP-based attacks: taxonomy, limitations and open challenges. *Computers & Security*. 1. vyd. 2023, sv. 124, č. 1, s. 102997. DOI: https://doi.org/10.1016/j.cose.2022.102997. ISSN 0167-4048. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167404822003893.

[21] Foundation, O. *OWASP Top Ten | OWASP Foundation*. OWASP Foundation, 2021 [cit. 2023-12-10]. Dostupné z: https://owasp.org/www-project-top-ten/.

[22] Foundation, O. *OWASP Top Ten 2017 | OWASP Foundation*. OWASP Foundation, 2021 [cit. 2023-12-10]. Dostupné z: https://owasp.org/www-project-top-ten/2017/.

[23] Foundation, O. *Path Traversal | OWASP Foundation*. OWASP Foundation, 2023 [cit. 2023-12-02]. Dostupné z: https://owasp.org/www-community/attacks/Path_Traversal.

[24] Foundation, O. *HTTP Response Splitting | OWASP Foundation*. OWASP Foundation, 2024 [cit. 2024-04-04]. Dostupné z: https://owasp.org/www-community/attacks/HTTP_Response_Splitting.

[25] Fredriksson, T., Mattos, D. I., Bosch, J. a Olsson, H. H. Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies. In: Morisio, M., Torchiano, M. a Jedlitschka, A., ed. *Product-Focused Software Process Improvement*. Cham: Springer International Publishing, 2020, s. 202–216. ISBN 978-3-030-64148-1.

[26] Friedman, N., Geiger, D. a Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning*. Listopad 1997, sv. 29, s. 131–163. DOI: 10.1023/A:1007465528199.

[27] Ganguly, B. a Sheikh, R. A Review of Focused Web Crawling Strategies. *International Journal of Advanced Computer Research*. Prosinec 2012, sv. 2, č. 4, s. 261–267. Copyright - Copyright International Journal of Advanced Computer Research Dec 2012; Last updated - 2023-09-05. Dostupné z: https://www.proquest.com/scholarly-journals/review-focused-web-crawling-strategies/docview/1288633462/se-2.

[28] Gharibeh, S., Melhem, S. a Najadat, H. Classification on Web Application Requests. In: *2020 11th International Conference on Information and Communication Systems (ICICS)*. 2020, s. 1–5. DOI: 10.1109/ICICS49469.2020.239537. ISBN 978-1-7281-6227-0.

[29] Guimaraes, B. D. A. a Stampar, M. *Sqlmap: automatic SQL injection and database takeover tool*. 2006– [cit. 2023-12-14]. Dostupné z: https://sqlmap.org/.

[30] Géron, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017. ISBN 978-1491962299.

[31] Han, J., Kamber, M. a Pei, J. *Data mining. Concepts and techniques*. 3. vyd. Amsterdam: Elsevier/Morgan Kaufmann, 2012. The Morgan Kaufmann Series in Data Management Systems. ISBN 978-0-12-381479-1. Dostupné z: www.sciencedirect.com/science/book/9780123814791.

[32] HUMAN. *HUMAN Bot Defender | Safeguard Websites, Mobile Apps & APIs From Bots*. 2023 [cit. 2023-12-11]. Dostupné z: https://www.humansecurity.com/products/human-bot-defender.

[33] Standardization, I. O. for. *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. Standard. Geneva, CH: International Organization for Standardization, listopad 1994.

[34] Ito, M. a Iyatomi, H. Web application firewall using character-level convolutional neural network. In: *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*. 2018, s. 103–106. DOI: 10.1109/CSPA.2018.8368694. ISBN 978-1-5386-0389-5.

[35] Jordan, M. I. a Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science*. 2015, sv. 349, č. 6245, s. 255–260. DOI: 10.1126/science.aaa8415. Dostupné z: https://www.science.org/doi/abs/10.1126/science.aaa8415.

[36] Jurafsky, D. a Martin, J. H. *Speech and Language Processing (3rd ed. draft)*. 3. Feb 2024. Dostupné z: https://web.stanford.edu/~jurafsky/slp3/.

[37] Karlík, P., Nekula, M., Pleskalová, J., Bachmannová, J., Balhar, J. et al. *Nový encyklopedický slovník češtiny* [tištěná verze "print"]. 1. vyd. Praha: Nakladatelství Lidové noviny, 2016. ISBN 978-80-7422-480-5.

[38] Kelleher, J. D., MacNamee, B. a D'Arcy, A. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA: MIT Press, 2015. ISBN 978-0-262-02944-5.

[39] Kettle, J. *HTTP Desync Attacks: Request Smuggling Reborn | PortSwigger Research*. PortSwigger, září 2019 [cit. 2024-04-04]. Dostupné z: https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn.

[40] kingthorin. *SQL Injection | OWASP Foundation*. OWASP Foundation, 2023 [cit. 2023-12-01]. Dostupné z: https://owasp.org/www-community/attacks/SQL_Injection.

[41] KirstenS a contributors. *Cross Site Scripting (XSS) | OWASP Foundation*. OWASP Foundation, 2023 [cit. 2023-12-02]. Dostupné z: https://owasp.org/www-community/attacks/xss/.

[42] Kolter, J. Z. a Maloof, M. A. Learning to Detect and Classify Malicious Executables in the Wild. *J. Mach. Learn. Res.* JMLR.org. Prosinec 2006, sv. 7, s. 2721–2744. ISSN 1532-4435.

[43] Kruegel, C. a Vigna, G. Anomaly Detection of Web-Based Attacks. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2003, s. 251–261. CCS '03. DOI: 10.1145/948109.948144. ISBN 1581137389. Dostupné z: https://doi.org/10.1145/948109.948144.

[44] Krueger, T., Gehl, C., Rieck, K. a Laskov, P. TokDoc: A Self-Healing Web Application Firewall. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2010, s. 1846–1853. SAC '10. DOI: 10.1145/1774088.1774480. ISBN 9781605586397. Dostupné z: https://doi.org/10.1145/1774088.1774480.

[45] LINHART, C., KLEIN, A., HELED, R. a ORRIN, S. *HTTP request smuggling - CGISECURITY*. Watchfire, červen 2005 [cit. 2023-12-11]. Dostupné z: https://www.cgisecurity.com/lib/http-request-smuggling.pdf.

[46] Ltd., P. *What is HTTP request smuggling? Tutorial & Examples | Web Security Academy*. 2023 [cit. 2023-12-13]. Dostupné z: https://portswigger.net/web-security/request-smuggling.

[47] Ltd., P. *What is XXE (XML external entity) injection? Tutorial & Examples | Web Security Academy*. 2024 [cit. 2024-01-13]. Dostupné z: https://portswigger.net/web-security/xxe.

[48] Makiou, A., Begriche, Y. a Serhrouchni, A. Improving Web Application Firewalls to detect advanced SQL injection attacks. In: *2014 10th International*

Conference on Information Assurance and Security. 2014, s. 35–40. DOI: 10.1109/ISIAS.2014.7064617.

[49] Mitchell, T. *The discipline of machine learning*. Pittsburgh, PA 15213: Carnegie Mellon University, 2006.

[50] Mooosa, A. Artificial Neural Network based Web Application Firewall for SQL Injection. *International Journal of Computer and Information Engineering*. World Academy of Science, Engineering and Technology. 2010, sv. 4, č. 4, s. 610 – 619. ISSN eISSN: 1307-6892. Dostupné z: https://publications.waset.org/vol/40.

[51] Moradi Vartouni, A., Teshnehlab, M. a Sedighian Kashi, S. Leveraging deep neural networks for anomaly-based web application firewall. *IET Information Security*. 2019, sv. 13, č. 4, s. 352–361. DOI: https://doi.org/10.1049/iet-ifs.2018.5404. Dostupné z: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-ifs.2018.5404.

[52] Nielsen, H., Mogul, J., Masinter, L. M., Fielding, R. T., Gettys, J. et al. *Hypertext Transfer Protocol – HTTP/1.1* [RFC 2616]. RFC Editor, červen 1999. DOI: 10.17487/RFC2616. Dostupné z: https://www.rfc-editor.org/info/rfc2616.

[53] Nsrav, Maas, T. a ADubhlaoich. *Embedding null code | OWASP Foundation*. OWASP Foundation, 2023 [cit. 2023-12-10]. Dostupné z: https://owasp.org/www-community/attacks/Embedding_Null_Code.

[54] Nyuytiymbiy, K. Parameters and Hyperparameters in Machine Learning and Deep Learning. *Towards Data Science*. Prosinec 2020. Dostupné z: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac.

[55] Pałka, D. a Zachara, M. Learning Web Application Firewall - Benefits and Caveats. In: Tjoa, A. M., Quirchmayr, G., You, I. a Xu, L., ed. *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, s. 295–308. ISBN 978-3-642-23300-5.

[56] Razzaq, A., Hur, A., Shahbaz, S., Masood, M. a Ahmad, H. F. Critical analysis on web application firewall solutions. In: *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*. Březen 2013, s. 1–6. DOI: 10.1109/ISADS.2013.6513431. ISBN 978-1-4673-5070-9.

[57] Sagiroglu, S. a Sinanc, D. Big data: A review. In: *2013 International Conference on Collaboration Technologies and Systems (CTS)*. 2013, s. 42–47. DOI: 10.1109/CTS.2013.6567202. ISBN 978-1-4673-6404-1.

[58] Samuel, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*. USA: IBM Corp. Červenec 1959, sv. 3, č. 3, s. 210–229. DOI: 10.1147/rd.33.0210. ISSN 0018-8646. Dostupné z: https://doi.org/10.1147/rd.33.0210.

[59] Sangwan, S. *S0md3v/XSStrike: Most advanced XSS scanner*. 2018–. [cit. 2023-12-17]. [Version 3.15]. Dostupné z: https://github.com/s0md3v/XSStrike.

[60] Sangwan, S. *Comparing XSStrike with other XSS Scanners · s0md3v/XSStrike Wiki*. 2023. [cit. 2023-12-17]. Dostupné z: https://github.com/s0md3v/XSStrike/wiki/Comparing-XSStrike-with-other-XSS-Scanners.

[61] Sarker, I. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*. Březen 2021, sv. 2. DOI: 10.1007/s42979-021-00592-x.

[62] shramos. *Awesome-Cybersecurity-Datasets*. 2018 [cit. 2024-04-30]. Dostupné z: https://github.com/shramos/Awesome-Cybersecurity-Datasets?tab=readme-ov-file#webapps.

[63] Snoswell, A. *The 322 most common syllables (ordered most common to least) in the 5000 most common English words*. 5. února 2022. [cit. 2024-04-22]. Dostupné z: https://gist.github.com/aaronsnoswell/41fb311d85f3130e690d559b2df2d1be#file-common-syllables-txt-L8.

[64] Stone, M. *The ups and downs of 0-Days: A year in review of 0-days exploited in-the-wild in 2022*. Google, červenec 2023 [cit. 2023-12-13]. Dostupné z: https://security.googleblog.com/2023/07/the-ups-and-downs-of-0-days-year-in.html.

[65] Sutton, R. S. a Barto, A. G. *Reinforcement Learning: An Introduction*. 2. vyd. The MIT Press, 2018. ISBN 978-0262039246. Dostupné z: http://incompleteideas.net/book/the-book-2nd.html.

[66] Technologies, A. *App & API Protector Product Brief | Akamai*. 2022 [cit. 2023-12-11]. Dostupné z: https://www.akamai.com/resources/product-brief/app-and-api-protector.

[67] Tremante, M., Belson, D. a Zejnilovic, S. *Application security report: Q2 2023*. Cloudflare, Inc., Aug 2022 [cit. 2023-12-02]. Dostupné z: https://blog.cloudflare.com/application-security-report-q2-2023/.

[68] Tremante, M., Zejnilovic, S. a Belson, D. *Application security: Cloudflare's view*. Cloudflare, Inc., březen 2022 [cit. 2024-01-16]. Dostupné z: https://blog.cloudflare.com/application-security/.

[69] Wressnegger, C., Schwenk, G., Arp, D. a Rieck, K. A close look on n-grams in intrusion detection: anomaly detection vs. classification. In: *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*. New York, NY, USA: Association for Computing Machinery, 2013, s. 67–76. AISec '13. DOI: 10.1145/2517312.2517316. ISBN 9781450324885. Dostupné z: https://doi.org/10.1145/2517312.2517316.

[70] Zhang, Z., George, R. a Shujaee, K. Efficient detection of anomolous HTTP payloads in networks. In: *SoutheastCon 2016*. 2016, s. 1–3. DOI: 10.1109/SECON.2016.7506638. ISBN 978-1-5090-2246-5.

Příloha A

Obsah přiloženého paměťového média

Přiložené paměťové médium obsahuje:

- kompletní zdrojové kódy systémů pro sběr maligních a benigních dat,
- kompletní zdrojové kódy pro normalizaci a analýzu textových dat,
- kompletní zdrojové kódy pro extrakci a standardizaci příznaků,
- zdrojový kód s použitelným finálním klasifikačním modelem,
- anotované datové sady s různými stupni normalizace,
- a tuto práci ve formátu PDF včetně jejich zdrojových souborů ve formátu LATEX.

Celý obsah paměťového média je dostupný i na službě GitHub[1].

[1] https://github.com/zzmichalqq/web-request-classificator