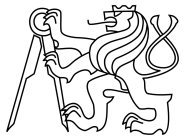
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Identifikace příznaků infikovaného uživatele z proxy logu
Student:	Daniel Nemčík
Vedoucí:	Ing. Martin Kopp
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

V současnosti je strojové učení nedílnou součásti síťové bezpečnosti. Umožňuje analyzovat obrovské množství dat s nebývalou přesností, je schopné odhalit zcela nové druhy útoků a nachází souvislosti tam, kde by je analytik snadno přehlédl. Většina komunikace je dnes navíc šifrovaná, a tak je informace, se kterou může analytik přímo pracovat, omezená. Tato práce by měla pomoci analytikům lépe pochopit rozhodnutí algoritmů strojového učení, na které jsou nuceni stále více spoléhat.

1) Nastudujte základy strojového učení, zaměřte se na jeho aplikaci v síťové bezpečnosti.
2) Nabyté znalosti ověřte na problému hledání infikovaných uživatelů v reálném síťovém provozu (data dodá vedoucí).
3) Zaměřte následný výzkum na identifikaci klíčových příznaků a na kombinaci slabých indikátorů chování uživatele pro odhalovaní infekce.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémov

Bakalárska práca

Identifikácia infikovaného užívateľa z proxy logu

Daniel Nemčík

Vedúci práce: Ing. Martin Kopp

15. mája 2017

Poďakovanie

Ďakujem vedúcemu tejto práce, pánovi Ing. Martinovi Koppovi, za cenné rady, ochotu a trpezlivosť pri písaní mojej práce. Ďakujem aj rodine a blízkym za podporu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 15. mája 2017 .....

Odkaz na túto prácu

Nemčík, Daniel. *Identifikácia infikovaného užívateľa z proxy logu*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Táto práca sa zaoberá aplikáciou algoritmov strojového učenia na detekciu malvéru v sieťovej komunikácii. Práca analyzuje vopred pripravené dáta pochádzajúce zo záznamov sieťovej komunikácie a upravuje ich pre účely spracovania strojovým učením. Pomocou rôznych algoritmov strojového učenia vytvára modely používané na detekciu malvéru a hodnotí ich presnosť. Práca sa zaoberá aj hodnotením modelov naučených pomocou príznakov šifrovanej komunikácie a slabých indikátorov.

Kľúčové slová strojové učenie, detekcia malvéru, výber príznakov, slabé indikátory

Abstract

This thesis deals with malware detection in network communication using machine learning algorithms. Thesis analyses proxy logs and adjusts them so they can be processed by machine learning. Then it compares the malware detection accuracy of different machine learning models. This thesis evaluates models learned using features extracted from encrypted communication and features based on weak labels.

Keywords machine learning, malware detection, feature selection, weak labels

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Téma internetových hrozieb určite nie je cudzia vlastníkom počítačových systémov každého rozsahu. Všetok softvér s účelom škodiť pokrýva pojem malvér. Mnoho typov malvéru využíva internet na svoje rozširovanie s úmyslom zneužiť výpočetný výkon nakazených počítačov na DoS útoky, na rozosielanie nevyžiadanej pošty, na zberanie potenciálne citlivých informácií či na šírenie nechcenej reklamy.

Ak vlastníme dáta, ktoré si chceme chrániť, musíme sa zaoberať prevenciou pred ich zneužitím alebo odcudzením vplyvom malvéru, či už ide o veľké korporácie, malé firmy alebo domácnosti. Na ochranu pred ním nám slúžia bezpečnostné prvky, ako napríklad antivírusové programy a zariadenia na detekciu a prevenciu prieniku. Tieto prvky musia byť neustále vylepšované, aby dokázali odhaliť škodlivú činnosť nových druhov malvéru. Existujúci malvér môže byť upravený tak, aby sa jeho činnosť v záznamoch sieťovej komunikácie podobala bežnej komunikácii užívateľa. Na tieto zmeny v správaní sú veľmi necitlivé detektory založené na princípe statických filtrov a musia byť aktualizované po tom, ako sa nový malvér identifikuje iným spôsobom.

Na riešenie problému detekcie malvéru v sieťovej komunikácii v súčastnosti s čoraz vyšším úspechom slúžia algoritmy strojového učenia. Používajú sa na kategorizáciu a detekciu anomálií v komunikácii. Dokážu dobre generalizovať, čo ich robí odolné voči zmenám v správaní. Tieto algoritmy sa na základe záznamov komunikácie naučia ako vyzerá štandardná komunikácia a všetky odchýlky vyhodnotia ako anomáliu, alebo ich zaradia do kategórie podľa druhu malvéru. To však prináša zvýšenie počtu falošných poplachov v detekcii, keďže napríklad nepravidelná aktualizácia užívateľskej aplikácie môže byť kategorizovaná ako anomálna. Množstvo vylepšení týchto algoritmov pomáha tento problém potláčať.

Nie všetok malvér sa dá zachytiť pred tým, ako sa rozšíri. Často sa o existencii malvéru v počítačovom systéme dozvieme až po tom, čo sa usadí a začne páchať škody. Preto väčšie inštitúcie v súčasnosti zbierajú rôzne záznamy sieťovej komunikácie vo vnútri svojej siete. Tieto záznamy sa po úprave dajú

spracovať strojovým učením, ktoré môže v dátach odhaliť neobvyklé vzory, pomôcť v detekcii škodlivej činnosti a v ideálnom prípade odhaliť nakazené zariadenia v sieti.

Táto práca skúma využitie a úspešnosť algoritmov strojového učenia pri identifikácii užívateľa nakazeného malvérom na základe sieťovej komunikácie. V poslednej dobe malvér využíva stále viac šifrovanú komunikáciu. Preto sme sa v tejto práci zamerali na príznaky, ktoré sú priamo dostupné z protokolu HTTPS. Na základe experimentov sme sa rozhodli pre reprezentáciu nezávislú od protokolu. Táto reprezentácia využíva slabé indikátory správania sa užívateľa. Konkrétne sa zameriavame na kombináciu indikátorov, ktorá môže znamenať vážnejší bezpečnostný incident. Experimenty boli robené na dátach reálnej sieťovej komunikácie v spolupráci so spoločnosťou Cisco.

Cieľe práce

- Analyzovať, pripraviť a strojovým učením spracovať dáta zo záznamov reálnej sieťovej komunikácie poskytnuté vedúcim práce
- Pomocou algoritmov strojového učenia vytvoriť a ohodnotiť prediktívne modely určené na detekciu užívateľov infikovaných malvérom v záznamoch šifrovanej sieťovej komunikácie
- Ohodnotiť presnosť detekcie vytvorených modelov pri detekcii infikovaných užívateľov na dátach obsahujúcich slabé indikátory správania užívateľov

Štruktúra práce

Prvá kapitola práce je venovaná teoretickému úvodu do strojového učenia. Zaoberá sa definíciami používaných pojmov, spôsobmi hodnotenia modelov a opisuje algoritmy strojového učenia použité v tejto práci. Druhá kapitola sa zaoberá popisom šifrovaných dát a ich analýzou. V tretej kapitole je rozobratý princíp slabých indikátorov a analýza poskytnutých dát obsahujúcich slabé indikátory správania sa užívateľa. Posledná časť práce sa venuje experimentom s algoritmami strojového učenia, pri ktorých sú vytvárané a testované modely detekcie nakazených užívateľov.

Kapitola 1

Úvod do strojového učenia

Táto kapitola definuje pojmy používané pri strojovom učení a zaoberá sa opisom algoritmov strojového učenia, ktoré boli použité v tejto práci. Tiež rozoberá stratégie hodnotenia naučených modelov a vysvetľuje, čo tieto stratégie ukazujú.

1.1 Učenie z dát

Cieľom použitia algoritmov strojového učenia je predpovedať výsledok na základe vopred naučených dát. Prístup, keď sa algoritmus učí pravidlá z dát, ktoré boli vopred označené, buď iným algoritmom, alebo ručne, sa nazýva *učenie pod dohľadom*[1]. Tieto algoritmy sa pokúšajú nájsť vzťah medzi vstupnými veličinami, tiež nazývanými *príznaky*[2] a výstupnou hodnotou, ktorá je pri klasifikačných problémoch nazývaná *štítok*[3] alebo *trieda*. Skupinu vstupných veličín opisujúcu jeden riadok v dátach nazývame *vstupný vektor* alebo *vzorka*.

Zdroj [1] uvádza formálnejší zápis: Uvažujme *prípady* alebo *objekty* vybrané z univerza Ω. Zoraďme množinu meraní prípadu do vopred stanoveného poradia, čiže nech sú hodnoty $x_1, x_2, \ldots, x_p$ vstupné hodnoty, kde $x_j \in \mathcal{X}_j$ (pre $j = 1, \ldots, p$) značí hodnotu vstupnej veličiny X_j. Dohromady tieto vstupné hodnoty tvoria p-dimenzionálny vstupný vektor $\mathbf{x}$, ktorý naberá svoje hodnoty z $\mathcal{X}_1 \times \ldots \times \mathcal{X}_j = \mathcal{X}$, kde $\mathcal{X}$ je definované ako vstupný priestor. Podobne zadefinujme $y \in \mathcal{Y}$ ako hodnotu výstupnej veličiny Y, kde $\mathcal{Y}$ je definované ako výstupný priestor. Podľa definície vstupný a výstupný priestor obsahuje všetky možné vstupné vektory, respektíve výstupné hodnoty.

Veličiny definujúce problém delíme do dvoch hlavných typov. Prvým typom sú kvantitatívne veličiny reprezentované hodnotami z množín celých

[1] Z anglického *supervised learning*
[2] Z anglického *feature*
[3] Z anglického *label*

alebo reálnych čísel. Druhým typom sú kvalitatívne veličiny, ktorých hodnoty sú symbolické a predstavujú jednotlivé kategórie veličiny. Formálne definície podľa zdroja [1] znejú:

Definícia 1.1.1. *Veličina X_j je* zoradená*, ak $\mathcal{X}$ je úplne zoradená množina. Konkrétne, veličina X_j je* numerická*, ak $\mathcal{X}_j \in \mathbb{R}$.*

Definícia 1.1.2. *Veličina X_j je* kategorická*, ak premenná X_j je konečná množina hodnôt bez prirodzeného zoradenia.*

V každej typickej úlohe učenia pod dohľadom sú predchádzajúce pozorovania zhrnuté v množine dát nazývanej *učiaca množina*. Pozostáva z množiny vstupných vektorov pozorovania s ich výstupnou hodnotou. Formálne ju zdroj [1] definuje takto:

Definícia 1.1.3. Učiaca množina $\mathcal{L}$ *je množina N párov vstupných vektorov a výstupných hodnôt $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, kde $\boldsymbol{x}_i \in \mathcal{X}$ a $y_i \in \mathcal{Y}$.*

Ekvivalentne môžeme množinu p-rozmerných vektorov $\mathbf{x}_i$ (pre $i = 1, \ldots, N$) zapísať pomocou matice $\mathbf{X}$ s rozmermi $N \times p$, ktorej riadky $i = 1, \ldots, N$ zodpovedajú vstupným vektorom $\mathbf{x}_i$ a stĺpce $j = 1, \ldots, p$ premenným X_j. Podobne môžeme vyjadriť množinu výstupných hodnôt ako vektor $\mathbf{y} = (y_1, \ldots, y_N)$.

Úloha učenia pod dohľadom teraz môže byť vyjadrená ako učenie funkcie $\varphi : \mathcal{X} \mapsto \mathcal{Y}$ z učiacej množiny $\mathcal{L} = (\mathbf{X}, \mathbf{y})$. Cieľom takejto úlohy je nájsť model, ktorého predikcie $\varphi(\mathbf{x})$, tiež označované $\hat{Y}$, sú čo najlepšie. Ak je Y kategorická veličina, tak úlohu učenia nazývame klasifikačná úloha. Formálna definícia výsledného modelu zneje takto:

Definícia 1.1.4. Klasifikátor *nazývame funkciu $\varphi : \mathcal{X} \mapsto \mathcal{Y}$, kde $\mathcal{Y}$ je konečná množina tried (štítkov) značená $\{c_1, c_2, \ldots, c_J\}$.* [1]

1.2 Hodnotenie výkonnosti modelu

1.2.1 Matica zámen

Po naučení algoritmu je potrebné zistiť, ako dobre dokáže daný model predpovedať správny výsledok. Zdroj [1] uvádza, že hľadanie modelu, ktorý má čo najlepšie odhady sa dá definovať ako minimalizácia predikčnej chyby. Predikčná chyba značí, ako veľmi sa odhad modelu líši od skutočnosti pomocou *stratovej funkcie*[4].

Druhým prístupom je odhad pomocou testovacej množiny. Spočíva v rozdelení učiacej množiny $\mathcal{L}$ na dve disjunktné množiny $\mathcal{L}_{train}$ a $\mathcal{L}_{test}$, nazývané *trénovacia množina* a *testovacia množina*. Algoritmus sa naučí na $\mathcal{L}_{train}$, za úlohu dostane uhádnuť správnu triedu jednotlivých vzorkov z $\mathcal{L}_{test}$ a na základe jeho predikcií sa ohodnotí jeho presnosť.

[4] Z anglického *loss function*

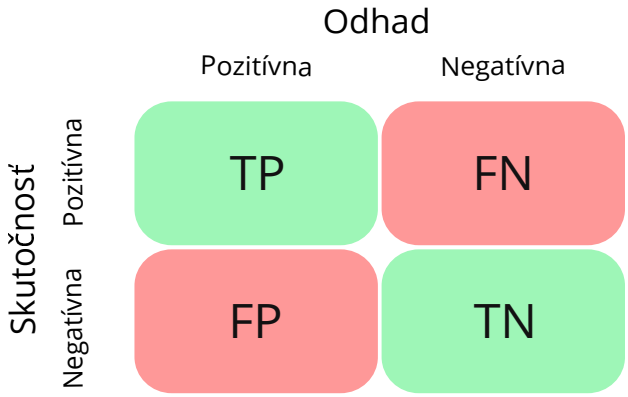
V tejto práci sú naučené modely hodnotené pomocou metrík odvodených z *matice zámen*[5]. Matica zámen sumarizuje výsledok klasifikácie klasifikátora v ohľade na testovaciu množinu. Táto matica má riadky indexované podľa tried výstupnej veličiny a stĺpce podľa tried, ktoré odhadol model. Obrázok 1 obsahuje príklad tejto matice pre klasifikačný problém, pri ktorom výstupná veličina obsahuje dva rôzne hodnoty, tiež nazývaný *binárny* klasifikačný problém. Obsahuje nasledujúce informácie:

Správne pozitívny (TP)[6] – Počet odhadov správne zaradených do pozitívnej triedy. Odhad testovacej vzorky zodpovedá skutočnej hodnote.

Nesprávne pozitívny (FP)[7] – Počet odhadov nesprávne zaradených do pozitívnej triedy. Tiež nazývaný chyba prvého druhu.

Správne negatívny (TN)[8] – Počet odhadov správne zaradených do negatívnej triedy. Odhad testovacej vzorky zodpovedá skutočnej hodnote.

Nesprávne negatívny (FN)[9] – Počet odhadov nesprávne zaradených do negatívnej triedy. Tiež nazývaný chyba druhého druhu.



Obr. 1: Matica zámen

1.2.2 Precíznosť a senzitivita

Na základe matice zámen si zadefinujeme pojmy *precíznosť*[10] a *senzitivita*[11]. Precíznosť hovorí o schopnosti modelu správne klasifikovať pozitívnu triedu.

[5] Z anglického *confusion matrix*
[6] Z anglického *True Positive*
[7] Z anglického *False Positive*
[8] Z anglického *True Negative*
[9] Z anglického *False Negative*
[10] Z anglického *precision*
[11] Z aglického *recall*, tiež nazývaný *sensitivity*

Ako príklad slúži model, ktorého úlohou je klasifikovať malvér, ktorý je v dátach pozitívnou triedou. Precíznosť v tomto prípade hovorí, koľko zo vzoriek označených ako malvér je naozaj malvérom. Senzitivita hovorí o schopnosti modelu správne odhadnúť všetky pozitívne vzorky ako pozitívne, napr. koľko vzorkov zo všetkých vzorkov malvéru v testovacích dátach bolo označených, čiže ako často je model úspešný v detekcii. Formálnejší zápis zneje takto:

Definícia 1.2.1. Precíznosť *je pomer počtu vzoriek označených za správne pozitívne (TP) k počtu správne pozitívnych a nesprávne pozitívnych (FP).*

$$precíznosť = \frac{TP}{TP + FP}$$

Definícia 1.2.2. Senzitivita *je pomer počtu vzoriek označených za správne pozitívne (TP) k počtu správne pozitívnych a nesprávne negatívnych (FN).*

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

1.2.3 PR krivka

Kombináciou precíznosti a senzitivity môžeme získať nástroj na hodnotenie binárneho klasifikátora, ktorý umožňuje vizuálne ohodnotenie jeho výkonnosti v rozsahu prahov[12]. Tento nástroj nazývame PR krivka[13]. Vlastnosti tejto krivky umožňujú dobré ohodnotenie výkonnosti klasifikátora, a to hlavne pri nevyvážených počtoch vzoriek v triedach výstupnej veličiny, keď je počet vzoriek jednej triedy podstatne vyšší ako počet vzoriek druhej.[2]

Zdroj [2] uvádza formálnejší popis. Uvažujme binárny klasifikačný problém, kde klasifikátor pre každú zo vzoriek priradí hodnotu zo spojitého rozsahu medzi nulou a jednotkou. Množinu týchto hodnôt označíme M. Keďže binárny klasifikačný problém znamená, že výstupná veličina nadobúda dve hodnoty, rozdelíme hodnoty množiny M do dvoch množín. Množina A obsahuje všetky negatívne výstupné hodnoty a množina B všetky pozitívne. Množina M je vlastne zmesou množín A a B. Vyššie výstupné hodnoty sú priradené vzorkám z pozitívnej triedy, takže vzorka je modelom odhadnutá za pozitívnu pre daný prah c, ak je výstupná hodnota vyššia ako prah c. Triedu reprezentujeme indikátorom D, kde $D = 1$ značí pozitívnu triedu a $D = 0$ negatívnu. Z hľadiska pravdepodobnosti si môžeme označiť nepomer tried π ako $\pi = P(D = 1)$.

PR krivku si teraz zadefinujeme podľa zdroja [2]:

Definícia 1.2.3. PR krivka *je množinou bodov:*

$$PR(\cdot) = \{(Senzitivita(c), Precíznosť(c)), 0 \leq c \leq 1\}$$

kde $Senzitivita(c) = P(B > c)$ *a* $Precíznosť(c) = P(D = 1|M > c)$.

[12] Z anglického *threshold*

[13] Z anglického *Precision-recall curve*

Keďže vyššie výstupné hodnoty znamenajú pozitívnu triedu, hodnota funkcie $Senzitivita(c)$ so zmenšujúcim sa c stúpa a hodnota $Precíznosť(c)$ klesá k hodnote π. Ak sa c zvyšuje, hodnota $Precíznosť(c)$ dosiahne hodnotu jedna a $Senzitivita(c)$ sa priblíži k nule.

Okrem vizuálneho ohodnotenia krivky existujú algoritmy na výpočet odhadu plochy pod PR krivkou, ktorá sa nazýva PR AUC[14]. Hodnota plochy pod krivkou poskytuje generálny odhad výkonnosti klasifikátora bez ohľadu na konkrétny prah.

1.3 Výber modelu

Podľa zdroja [1] by bolo na vyriešenie problému učenia pod dohľadom postačujúce nájsť odhad $P(Y|X)$ z učiacej vzorky $\mathcal{L}$ a tento odhad použiť na vytvorenie modelu. Tento prístup ale nie je možné využiť v praxi, keďže obtiažnosť nájdenia takéhoto odhadu rastie exponenciálne s veľkosťou dimenzie vstupných vektorov.

Ak chceme riešiť problém učenia pod dohľadom na vstupných priestoroch s vysokou dimenziou, musíme štruktúru optimálneho modelu φ_B zjednodušiť. Konkrétne, algoritmus učenia pod dohľadom počíta s tým, že φ_B, alebo jeho dostatočne presná aproximácia sa dá nájsť v množine kandidátov na model s obmedzenou štruktúrou, ktorú značíme $\mathcal{H}$. Je tiež známa ako *hypotéza* v teórii štatistického učenia. V tomto prípade môžeme definovať problém *selekcie modelu* ako problém hľadania najlepšieho modelu v $\mathcal{H}$ na základe učiacej množiny $\mathcal{L}$.

Formálnejšie zdroj [1] uvádza: Nech θ predstavuje vektor hyper-parametrov učiaceho algoritmu $\mathcal{A}$. Aplikácia algoritmu $\mathcal{A}$ s hyper-parametrami θ na učiacu množinu $\mathcal{L}$ je deterministický proces, ktorého výsledkom je model

$$\mathcal{A}(\theta, \mathcal{L}) = \varphi_{\mathcal{L}} \in \mathcal{H}$$

a na základe toho je našim cieľom nájsť vektor hodnôt hyper-parametrov dávajúci najlepší možný model naučiteľný medzi modelmi $\mathcal{H}$ na učiacej množine $\mathcal{L}$ minimalizáciou generalizačnej chyby.

Tento problém v praxi tiež nie je efektívne riešiteľný, no existuje niekoľko spôsobov, ako získať dostatočne dobrý odhad vektoru θ. Jeden z nich spočíva v rozdelení učiacej množiny $\mathcal{L}$ na množiny $\mathcal{L}_{train}$ a $\mathcal{L}_{test}$ a v minimalizácii testovacej chyby definovanej zdrojom [1]. V tomto prípade je to v praxi tiež obtiažna úloha, no napríklad manuálnym ladením parametrov, alebo pomocou algoritmu *grid search* dokážeme získať dostatočne dobrý odhad vektoru parametrov.

Alternatívou k odhadu testovacej chyby je krížová validácia pomocou K častí[15]. Učiaca množina $\mathcal{L}$ sa rozdelí na K rovnako veľkých disjunktných pod-

[14] Z anglického *Area Under Curve*

[15] Z anglického *K-fold cross validation*

množín $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_K$ a generalizačná chyba algoritmu sa odhadne ako priemer predikčných chýb jednotlivých podmnožín $\mathcal{L}_k$ na modeloch $\varphi_{\mathcal{L}\setminus\mathcal{L}_k}$.

V tejto práci budeme namiesto odhadov chýb na hodnotenie modelov používať odhad PR AUC a PR krivku. PR AUC je v našom prípade lepšou metrikou, keďže pracujeme s nevyváženým pomerom tried a nepomer tried nemá vplyv na schopnosť PR AUC ohodnotiť model. PR krivka nám naviac umožňuje pomocou prahov prispôsobiť pomer precíznosti a senzitivity, ktorý je dosiahnuteľný vo výslednom modeli.

1.4 Náhodný les

Náhodný les[16] je v súčasnosti jedným z najpopulárnejších algoritmov strojového učenia. Svoju popularitu si získal uplatnením v mnohých odboroch, keďže jeho vlastnosti poskytujú veľkú prispôsobivosť. Môže byť použitý na predikčné problémy dvoch, ale aj viacerých tried, modeluje vzťahy medzi príznakmi, a poradí si s kategorickými a aj so spojitými premennými. Tiež poskytuje mieru dôležitosti príznakov a dosahuje dobrú presnosť predikcií aj v prípade, že dáta obsahujú viac príznakov ako vzoriek, čo môže znamenať veľmi zašumené a vysoko korelované príznaky.[3]

Náhodný les je tvorený súborom rozhodovacích stromov. Zdroj [4] hovorí, že rozhodovací strom je množinou uzlov a hrán zaradených do hierarchickej štruktúry. Uzly sú rozdelené na vnútorné uzly, tiež nazývané deliace uzly, a koncové uzly nazývané listy. Fungovanie rozhodovacieho stromu môžeme rozdeliť do fázy trénovania a fázy testovania. Trénovacia fáza spočíva v naučení stromu na trénovacej množine $\mathcal{L}_{train}$ s príslušnými výstupnými hodnotami, pričom je množina $\mathcal{L}_{train}$ rekurzívne delená deliacimi uzlami. Bežne využívanou funkciou hodnotenia kvality rozdelenia množiny je funkcia *zisku informácií*[17]. Zisk informácií dosiahnutý rozdelením množiny $\mathcal{S}_j$ v uzle j je vypočítaný ako

$$I_j = H(\mathcal{S}_j) - \sum_{i\in\{L,R\}} \frac{\left|\mathcal{S}_j^i\right|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i) \tag{1.1}$$

kde entropia $H(\mathcal{S}_j)$ je vypočítaná ako

$$H(\mathcal{S}_j) = -\sum_{c\in\mathcal{Y}} p(c)\log(p(c)) \tag{1.2}$$

V trénovacej fáze sa vyberú parametre θ_j deliaceho uzlu j, ktoré maximalizujú funkciu zisku informácií I_j. Cieľom učenia je teda každému z deliacich uzlov j nájsť deliace parametre θ_j^* maximalizujúce funkciu I_j. Množina sa delí, čím

[16] Z anglického *Random Forest*

[17] Z anglického *information gain*, tiež nazývaná *cross entropy*

strom rastie, pokým sa nedosiahne kritérium zastavenia, definované pomocou maximálnej hĺbky stromu alebo minimálneho zisku informácií. Výsledné rozdelenie množiny je uložené v listoch stromu.

Pri trénovaní stromu sa dá spočítať, ako každý z náhodne vybratých príznakov prispieva k zvýšeniu miery váženého zisku informácií. Tieto zvýšenia sa dajú medzi stromami v lese spriemerovať, čím získame dôležitosti príznakov v rámci lesa.

V testovacej fáze je každému deliacemu uzlu priradená deliaca funkcia $h(\mathbf{x}_i; \theta_j^*)$, ktorá pre všetky vektory $\mathbf{x}_i$ z množiny $\mathcal{L}_{train}$ na základe nájdených deliacich parametrov θ_j^* vyberie hranu, po ktorej sa vektor dostane do ďalšieho uzlu.

Listy stromu sú teda predikčným modelom, ktorý pri klasifikačných problémoch hovorí o pravdepodobnosti $p(c|\mathbf{x}_i)$. V rámci lesa označujeme predikčný model listov stromu $t \in T$, kde T je počet stromov v súbore, ako $p_t(c|\mathbf{x}_i)$ a výsledný model súboru stromov je

$$p(c|\mathbf{x}_i) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|\mathbf{x}_i) \tag{1.3}$$

čiže výsledný odhad triedy pre vektor $\mathbf{x}_i$ je trieda s najvyššou pravdepodobnosťou.

Pri náhodných lesoch sa pri fáze učenia každému zo stromov poskytne podmnožina množiny $\mathcal{L}_{train}$ získaná náhodným výberom s opakovaním. Každý z deliacich uzlov pri určovaní rozdelenia pracuje len s náhodnou podmnožinou príznakov. Náhodnosť je využitá iba vo fáze učenia – fáza testovania je deterministický proces po tom, čo sú stromy pevne určené.

Tento postup funguje veľmi dobre v porovnaní s množstvom iných klasifikátorov, akými sú napríklad analýza diskriminantov, SVM[18] či neurónové siete, a je robustný voči preučeniu.[5][6]

1.5 SVM

Ďalším z populárnych algoritmov strojového učenia je SVM. Funguje na báze rozdelenia množiny dát pomocou nadroviny, ktorá sa tiež nazýva *hranica rozhodovania*. Priestor trénovacích dát delí na dve časti: časť pre negatívnu a časť pre pozitívnu triedu. Hranica rozhodovania je umiestnená tak, aby mala od najbližších inštancií trénovacích dát čo najväčšiu vzdialenosť. Najbližším inštanciám dát k hranici rozhodovania hovoríme *podporné vektory*[19]. SVM predvolene rieši binárny klasifikačný problém, no existujú rozšírenia umožňujúce riešenie problémov klasifikácie viacerých tried.

Nie každá množina trénovacích dát sa dá bezchybne rozdeliť lineárnou hranicou rozhodovania. Tento problém sa rieši zavedením penalizácie chyby

[18] *Support Vector Machine*
[19] Z anglického *support vector*

pomocou parametru C, čo umožňuje vybrať najmenej penalizované uloženie hranice rozhodovania.

SVM umožňuje projekciu pôvodnej množiny trénovacích dát z priestoru $\mathcal{X} \in \mathbb{R}^d$ do priestoru príznakov $\mathcal{F}$ s vyššou dimenziou pomocou funkcie jadra K. Výber rôznych funkcií jadra umožňuje projekcie z $\mathcal{X}$ do $\mathcal{F}$, pre ktoré nadroviny v $\mathcal{F}$ po spätnej projekcii do $\mathcal{X}$ zodpovedajú komplexnejším hraniciam rozhodovania, ktoré dokážu dáta lepšie rozdeliť.[7][8]

Podľa zdroja [7] je funkcia jadra K funkciou $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Princípom jadra je, že dokáže efektívne vypočítať skalárny súčin v priestore $\mathbb{R}^m$ s vyššou dimenziou, pričom zostane v $\mathbb{R}^n$. To môžeme zapísať nasledovne: Pre $x_i, x_j \in \mathbb{R}^n, K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_m$, kde $\langle \cdot, \cdot \rangle_m$ je skalárny súčin v $\mathbb{R}^m$, $m > n$, a $\phi(x)$ transformuje vektor x do $\mathbb{R}^m$ ($\phi : \mathbb{R}^n \to \mathbb{R}^m$).

V tejto práci používame okrem lineárneho SVM aj SVM s RBF jadrom. RBF jadro sa dá vyjadriť ako $K(x_i, x_j) = exp(-\gamma \cdot |x_i - x_j|^2)$, kde $\gamma > 0$.

Po trénovacej fáze, pri ktorej sa určí hranica rozhodovania, nasleduje testovacia fáza, kde sa stačí pozrieť na stranu hranice rozhodovania, na ktorej bod z testovacej množiny leží a podľa nej určiť výstupnú triedu.

Kapitola 2

Malvér a HTTPS

Protokol HTTP bol pôvodne na internete používaný v nešifrovanej podobe, no jeho nasadzovanie na komunikáciu kritických aplikácií si vyžadovalo zvýšenie bezpečnosti. Protokoly SSL a jeho následník TLS boli navrhnuté s úmyslom poskytnúť zabezpečenie na úrovni komunikačného kanálu.[9] Protokol HTTPS, tiež nazývaný HTTP over TLS úspešne využíva tieto protokoly na utajenie komunikácie klienta so serverom.

Keďže sa protokol TLS stal časom populárnym a jednoduchým na použitie, začal byť používaný malvérom na zabezpečenie jeho komunikácie. Pre tvorcov malvéru je pomerne jednoduché využiť existujúce knižnice TLS podľa ich potrieb bez veľkého zásahu do kódu. Jednoduchosť použitia je v tomto prípade znepokujujúca, lebo umožňuje malvéru vyhnúť sa detekcii a splynúť s bežnou komunikáciou na sieti.[10]

Podľa zdroja [10] malvér komunikujúci pomocou TLS tvorí približne 10 % zachytenej komunikácie malvéru. Zdroj [11] uvádza, že 38 % malvéru komunikujúceho pomocou TLS sa dá odlíšiť tým, že používa šifry považované za nedostatočne bezpečné, pričom tieto zastaralé šifry sú v bežnej komunikácii zastúpené s podielom menej ako 1 %. Táto skutočnosť poukazuje na možnú úspešnosť detekcie šifrovanej komunikácie malvéru.

Táto kapitola sa zaoberá analýzou príznakov šifrovanej komunikácie priamo dostupných z protokolu HTTPS.

2.1 Protokol HTTPS

Protokol HTTPS je konceptuálne jednoduchý, keďže ide o využitie protokolu HTTP cez TLS rovnako, ako sa používa HTTP cez TCP.[9] Preto sa budeme zaoberať najmä protokolom TLS.

2.1.1 Protokol TLS

Primárnym cieľom protokolu TLS je poskytnúť utajenie a integritu dát medzi dvoma komunikujúcimi aplikáciami. Aplikáciam typu klient/server umožňuje komunikovať spôsobom, ktorý znemožnuje odposluch, nechcenú manipuláciu a falšovanie správ. TLS sa pri prenose dát nachádza medzi aplikáciou a transportným protokolom, akým je napríklad TCP. Pre užívateľov je tento protokol transparentný, takže si užívateľ jeho využitie vôbec nemusí všimnúť. Tento protokol je tvorený dvoma vrstvami: TLS Record Protocol a TLS Hanshake Protocol. Na najnižšej úrovni, vrstvený na spoľahlivom protokole ako TCP, sa nachádza TLS Record Protocol. Má na starosti zabezpečenie komunikácie a podľa zdroja [12] má dve základné vlastnosti:

- Spojenie je utajené: Symetrická kryptografia je použitá na zašifrovanie dát. Pre každé spojenie je vygenerovaný unikátny kľúč založený na utajenom prvku dohodnutom iným protokolom, napríklad protokolom TLS Handshake Protocol.

- Spojenie je spoľahlivé: Prenos správy obsahuje kontrolu integrity správy pomocou MAC.

TLS Record Protocol slúži na zapuzdrenie protokolov vyššej vrstvy. Jedným z takro zapuzdrovaných protokolov je TLS Handshake Protocol. Umožňuje autentizáciu medzi klientom a serverom, vyjednanie šifrovacieho algoritmu a výmenu kryptografickych kľúčov pred tým, ako aplikačný protokol vyššej vrstvy prenesie alebo dostane jediný bajt dát. Bezpečnosť poskytovaná protokolom TLS Handshake Protocol má podľa zdroja [12] tri základné vlastnosti:

- Identita druhej strany môže byť overená použitím asymetrickej kryptografie.

- Vyjednanie zdieľaného tajného prvku je bezpečné: nedá sa zistiť ani v prípade odposluchu komunikácie.

- Vyjednanie je spoľahlivé: útočník nemôže upraviť komunikáciu pri vyjednávaní bez toho, aby bol odhalený účastníkmi komunikácie.

Z týchto vlastností vyplýva, že sledovanie dvoch strán komunikujúcich pomocou protokolu TLS nám neumožní získanie detailných vlastností komunikácie protokolu aplikačnej vrstvy, keďže tá je v celku šifrovaná. Zostáva nám len zozbierať čo najviac informácií poskytnutých protokolmi na nižších vrstvách a pomocou algoritmov strojového učenia sa pokúsiť odhaliť škodlivú komunikáciu.

2.1.2 Príznaky dostupné zo sledovania HTTPS komunikácie

Keďže dáta prenášané protokolom HTTPS sú šifrované protokolom TLS, zachytením komunikácie môžeme získať len dáta spojenia, ktoré sa nešifrujú, ako napríklad IP adresy klienta a servera, port spojenia alebo veľkosť prenesených dát. Poskytnuté vzorky obsahujú okrem iného dáta zachytené pomocou Netflow[20]. Nasledujúce príznaky sú priamo dostupné z protokolu HTTPS, alebo sledovaním jedného HTTPS spojenia medzi klientom a serverom:

Prvá skupina príznakov obsahuje príznaky so záznamom použitých portov pri komunikácii klienta so serverom:

FirstSeenPort – prvý použitý port

LargestPort – najvyšší použitý port

Druhá skupina popisuje objem prenesených dát počas komunikácie:

getNumPackets – počet paketov prenesených počas jedného spojenia

getNumBytes – koľko bajtov dát bolo prenesených počas jedného spojenia

Tretia skupina zahŕňa informácie dostupné z protokolu transportnej vrstvy:

getSynPackets – množstvo TCP SYN paketov v komunikácii. SYN pakety slúžia na synchronizáciu čísla sekvencie dát

getFinPackets – množstvo TCP FIN paketov v komunikácii. FIN paket značí, že tento paket je posledným, a jeho odosielateľ už nepošle viac paketov

getRstPackets – množstvo TCP RST paketov zaslaných klientom a serverom. RST paket je bežný TCP paket s nastaveným kontrolným bitom RST v TCP hlavičke. Tento kontrolný bit značí, že pripojenie má byť ukončené a má sa nadviazať nové pripojenie.

isPersistent – označuje, či ide o tzv. perzistentné spojenie – tiež nazývané HTTP keep-alive alebo HTTP connection reuse. Ide o použitie jedného TCP pripojenia pre viac HTTP požiadavok alebo odpovedí namiesto vytvárania nového pripojenia pre kažý požiadavok.[13]

Ďalšie dostupné príznaky pochádzajú z Netflow:

getClient – IP adresa klienta, alebo pôvodcu pripojenia

getServer – IP adresa servera, alebo cieľa pripojenia

getProtocol – protokol transportnej vrstvy použitý na prenos dát

[20] Netflow je protokol slúžiaci na zberanie informácií na sieti

getEndTime – časová značka konca spojenia

getElapsedTime – čas trvania spojenia

getServerAutonomousSystem – autonómny systém, v ktorom sa nachádza server

getServerCountry – krajina pôvodu IP adresy servera

Poskytnuté dáta naviac obsahujú príznaky, ktoré predstavujú globálnu znalosť ako je pravdepodobnosť pripojenia na server v danom štáte alebo v danom autonómnom systéme.

2.2 Analýza poskytnutých dát

Poskytnuté dáta pochádzajú z reálnych počítačov, ktoré boli zámerne nakazené.

Súbory sú rozdelené na súbory s komunikáciou, o ktorej vieme, že jej pôvodcom je užívateľ, alebo konkrétny počítač nakazený malvérom, a na súbory obsahujúce komunikáciu nenakazených užívateľov. Súbory s malvérom obsahujú vždy komunikáciu jedného typu malvéru zmiešanú z neškodnou komunikáciou užívateľa. Všetko čo vieme je, že daný užívateľ bol nakazený určitým typom malvéru, no nevieme, ktorý zo vzorkov komunikácie zodpovedá komunikácii škodlivého softvéru.

Vzorky dát predstavujú jedno spojenie klienta so vzdialeným serverom a stĺpce sú tvorené informáciami, ktoré boli pre dané spojenie zachytené. Týmto stĺpcom tiež hovoríme príznaky pre účely algoritmov strojového učenia. Pozostávajú z dát rôznych typov. Ide o číselné, slovné príznaky alebo IP adresy.

Nie vždy sú pri zachycovaní informácií o spojení dostupné dáta pre všetky príznaky. To spôsobí, že v zachytených dátach máme chýbajúce hodnoty. Niektoré algoritmy strojového učenia sú na to veľmi citlivé. Chýbajúce hodnoty môžu do procesu učenia zaniesť pomerne veľa šumu, či dokonca znemožniť fungovanie čisto numerických algoritmov. Algoritmy pracujúce s kategorickými hodnotami si s chýbajúcimi hodnotami poradia vytvorením samostatnej kategórie pre chýbajúce dáta. Ďalším problémom pre niektoré z algoritmov môžu byť rôzne rozsahy príznakov.

V procese učenia modelu je dôležité dať si pozor na príznaky, ako napríklad čas ukončenia spojenia. Hlavne pri malom počte vzoriek sa algoritmus naučí čas ukončenia spojenia pre všetky vzorky malvéru a dôjde k preučeniu modelu. Algoritmus dá tomuto príznaku veľkú dôležitosť, lebo jednoznačne identifikuje vzorky malvéru, pričom vieme, že čas ukončenia komunikácie nemôže určiť, či ide o komunikáciu škodlivého programu. Tým síce dosiahne výborný výsledok pri trénovaní, no zlyhá v detekcii na nových dátach, čo sa prejavilo pri pokusoch v ďalšej časti práce.

2.3 Predspracovanie dát

2.3.1 Úprava typu dát

Pre algoritmy strojového učenia budeme musieť vstupné dáta upraviť tak, aby tieto údaje boli číselné, pokiaľ to je možné. Príznaky s nominálnymi nečíselnými údajmi je možné prekódovať do číselnej reprezentácie nahradením unikátnych kategórií číslom. Príznaky textového charakteru na začiatok pre jednoduchosť vynecháme a zvážime ich použitie v závislosti na výsledkoch testovania.

2.3.2 Úprava rozsahu dát

Veľký rozptyl či rôzne rozsahy hodnôt sa dá vyriešiť normalizáciou na rovnakú strednú hodnotu a rozptyl jednotlivých príznakov, pričom dáta neprídu o svoju výpovednú hodnotu. Podľa zdroja [14] táto úprava zrýchli konvergenciu algoritmov, ktoré fungujú na princípe minimalizácie euklidovskej vzdialenosti. Na funkciu algoritmov založených na rozhodovacích stromoch nemá normalizácia žiaden vplyv a v tejto práci pracujeme hlavne s náhodnými lesami. Aj napriek tomu normalizáciu použijeme. Ak by sme chceli v budúcnosti skúsiť aj iné typy algoritmov, budeme môcť pracovať s rovnakými dátami.

2.3.3 Náhrada chýbajúcich hodnôt

Pre algoritmy strojového učenia sú chýbajúce hodnoty nežiadúcim javom. Samotná reprezentácia chýbajúcej hodnoty nám môže do dát zaniesť nechcenú skreslenosť, ktorá môže mať negatívny vplyv na presnosť odhadu algoritmu. Niektoré algoritmy si dokonca s chýbajúcimi hodnotami nevedia vôbec poradiť. Najjednoduchším riešením by bolo záznamy o komunikácii predstavované riadkami v dátových súboroch nepoužiť, no tým by sme vyradili veľké množstvo potenciálne užitočných informácií, keďže väčšine záznamov chýba údaj v jednom či vo dvoch príznakoch. Ďalšou možnosťou by bolo získať lepšie dáta, a to bez chýbajúcich hodnôt, no to v tomto prípade nie je možné.

Niekoľko príznakov vyskytujúcich sa v poskytnutých dátach obsahuje veľké množstvo chýbajúcich hodnôt. Príznaky, u ktorých chýba väčšina hodnôt vyradíme z použitia, keďže príznak, u ktorého je väčšina hodnôt umelo doplnená ma zanedbateľný, či až nechcený prínos pri učení algoritmu. Problém chýbajúcich hodnôt vyriešime nahradením chýbajúcej hodnoty najčastejšie sa vyskytujúcou hodnotou príznaku pre nominálne dáta. Pri číselných dátach použijeme náhradu za strednú hodnotu príznaku.

Možným vylepšením do budúcna by bolo využitie postupov dopĺňania chýbajúcich hodnôt opísaných zdrojom [15], pri ktorých sa na odhad chýbajúcej hodnoty používajú pokročilejšie štatistické metódy.

Kapitola 3

Detekcia pomocou udalostí

Veľkú časť komunikácie užívateľov na sieti tvoria nešifrované dáta. Zbieranie informácií o komunikácii, použitých protokoloch, type prenášaných dát a pod. poskytuje obrovský objem informácií. V sieťach inštitúcií, v ktorých všetka komunikácia prechádza cez jeden centrálny proxy server je takéto zbieranie informácií obzvlášť jednoduché, no zozbierané dáta majú obrovský objem.

Objem dát je to, čo znemožňuje ručnú analýzu komunikácie za účelom odhalenia komunikácie škodlivých programov. Zbierať môžeme dáta, ako adresa a doména servera, druh prenášaných dát a mnoho ďalších, čo poskytne množstvo informácií. Dáta je potrebné prepracovať odfiltrovaním irelevantných informácií a zo zozbieraných dát vytvoriť vektory príznakov.

V tejto kapitole sa budeme zaoberať analýzou reprezentácie vstupných veličín pomocou *udalostí*. Uskutočnenie týchto udalostí môžeme sledovať v rámci jednej IP adresy počas dňa, čo nám poskytne obraz o správaní sa jedného užívateľa na sieti.

3.1 Udalosti a slabé ukazovatele

Pojmom *udalosť* tejto práci označujeme informáciu o vykonaní vopred definovanej akcie počas komunikácie užívateľa v období jedného dňa. Udalosti definujeme nezávisle na protokole komunikácie. Udalostiam môžeme rozumieť ako typ slabého indikátoru, ktorý síce môže označovať prítomnosť vysoko podozrivej aktivity, no nemusí nutne značiť prítomnosť komunikácie škodlivého programu. Udalosti môžu byť zostrojené z nasledujúcich javov:

- HTTP požiadavok na server pomocou čistej IP adresy
- často sa opakujúci požiadavok
- komunikácia so serverom v neobvyklej časti sveta
- stiahnutie podozrivého binárneho súboru

- komunikácia s vygenerovanou alebo falošnou doménou
- viacero pokusov o prihlásenie

Vzniknutá reprezentácia je obdobou populárneho princípu *bag-of-word* používaného pri klasifikácii textu, kde každá udalosť predstavuje jedno slovo v slovníku a teda jednu binárnu hodnotu vo výstupnom vektore.

Zavedenie udalostí zníži dimenzionalitu dát, čo uľahčuje učenie modelov učenia pod dohľadom. Ďalším zjednodušením je, že udalosti sú reprezentované v binárnej podobe. Dôsledkom toho nemusíme riešiť chýbajúce hodnoty, nerovnomerné rozdelenie príznakov a ani úpravu príznakov do numerickej podoby.

3.2 Analýza poskytnutých dát

Poskytnuté dáta obsahujú udalosti v komunikácii užívateľov za obdobie mesiacov február a marec 2017 rozdelenú do dní. Vektory vzoriek sú binárne vektory, ktoré majú na jednotlivých pozíciach informáciu, či udalosť nastala alebo nenastala. Výstupné hodnoty sú tiež v podobe vektorov, ktorých prvky značia prítomnosť štyroch známych typov malvéru v komunikácii, ktoré sú obzvlášť nebezpečné. Konkrétne ide o nasledujúce typy malvéru:

info stealer – snaží sa získať a odoslať informácie o užívateľovi za účelom zneužitia.

banking trojan – trójsky kôň, ktorý sa od užívateľa snaží získať prístup k bankovému účtu.

click fraud – bez vedomia užívateľa kliká na reklamy, ktorými generuje zisk.

malware distribution – druh malvéru tiež nazývaný *dropper*. Jeho zámerom je nahrať do počítača ďalšie moduly malvéru ako info stealer, modul pre rozosielanie spamu a podobne.

Keďže povaha udalostí ako binárnych príznakov nedovoľuje výskyt chýbajúcich hodnôt, nemusíme riešiť ich dopĺňanie, škálovanie príznakov a ani rôzne dátové typy medzi príznakmi.

3.3 Predspracovanie dát

Kvôli množstvu dát a nevyváženosti výstupných tried je potrebné dáta upraviť. Pozitívna trieda je zastúpená pomerne málo, takže vyberieme všetky vzorky s pozitívnou triedou a náhodným výberom vyberieme vzorky z negatívnej triedy tak, aby výsledný pomer pozitívnej a negatívnej triedy bol 1:100. Tento pomer má simulovať skutočnosť, že podľa firmy Cisco škodlivá komunikácia tvorí približne 1 % celkovej komunikácie na internete.

Kapitola 4

Experimenty

V tejto kapitole sa budeme zaoberať učením algoritmov a hodnotením naučených modelov. Na implementáciu bol použití programovací jazyk *Python* verzie 3.4 s knižnicou *scikit-learn*, ktora obsahuje efektívne implementácie bežne používaných algoritmov strojového učenia spolu s nástrojmi na ich testovanie.

4.1 Experimenty s detekciou malvéru v HTTPS komunikácii

4.1.1 Vytvorenie učiacej množiny

Vstupné dáta pozostávajú z dvoch hlavných častí. Prvou časťou je zbierka súborov obsahujúcich komunikáciu užívateľov alebo počítačov s počtom vzoriek mierne presahujúcim číslo 5 800 000. Každý zo súborov obsahuje vzorky zachytenej komunikácie, o ktorej vieme, že pochádza od užívateľa, ktorý je určite nakazený jedným typom malvéru. Skutočnosťou je, že tieto vzorky obsahujú komunikáciu škodlivých, ako aj neškodných programov, no nevieme určiť škodlivosť alebo neškodnosť jednotlivých vzoriek. V závislosti na type malvéru je pomer vzoriek škodlivej komunikácie s neškodnou v rámci jedného súboru premenlivý. Niektoré typy malvéru komunikujú len zriedka a tým pádom je ich detekcia pomocou vzoriek komunikácie obzvlášť obtiažna.

Na učenie modelov budeme musieť zredukovať počet vzoriek, aby bolo možné model zostrojiť v rozumnom čase. Preto sme spomedzi všetkých typov malvéru vybrali tie, pre ktoré máme rádovo tisícky vzoriek. Niektoré z typov totižto obsahujú milióny vzoriek, a bolo by nutné vybrať ich podmnožinu, ktorá by mohla z veľkej časti pozostávať z neškodnej komunikácie. Tabuľka 1 obsahuje časť vybraných druhov, u ktorých sme použili všetku dostupnú komunikáciu na trénovanie modelu. Vo výsledku ide približne o 11 000 vzoriek pozitívnej triedy.

Druhá časť dát je tvorená dátami komunikácie pochádzajúcej od užívateľov, ktorí by podľa všetkého nemali byť infikovaní žiadnym typom malvéru.

Túto časť dát tvorí približne 1 300 000 vzoriek. Aby sme zaručili vyváženosť dát, vybrali sme z tejto časti 16 000 vzoriek a označili sme ich negatívnym štítkom.

4.1.2 Vytvorenie trénovacej a testovacej množiny

Naším cieľom je zistiť úspešnosť detekcie jednotlivých typov malvéru. Na to použijeme testovanie pomocou metódy *leave one out*. Táto metóda pozostáva z naučenia modelu na všetkých typoch malvéru okrem jedného z nich. Typ malvéru vynechaný z učenia použijeme po premiešaní so vzorkami neškodnej komunikácie s negatívnym štítkom na testovanie naučeného modelu.

Testovacia a trénovacia množina má teda rôzny obsah na základe toho, aký typ malvéru testujeme.

4.1.3 Vybudovanie modelu a ohodnotenie detekcie

Model sme vytvorili pomocou algoritmu náhodný les s miernou úpravou parametrov. Pomocou metódy *leave one out* sme zistili precíznosť a senzitivitu detekcie vybraných typov malvéru. Tabuľka 1 obsahuje výsledky testovania.

Tabuľka 1: Skóre klasifikácie malvéru pomocou náhodného lesa

Malvér	**Precíznosť**	**Senzitivita**	**PR AUC**
Urelas	0.400000	0.024390	0.185022
Bifrose	0.333333	0.018634	0.225896
NanoBot	0.500000	0.200000	0.256658
Sality	0.983051	0.267281	0.457478
Trojan-agent	0.428571	0.500000	0.517811
DomaIQ	0.621622	0.748197	0.713765
Matsnu	0.951220	0.270833	0.832786
Ramdo	0.921429	0.064500	0.975156
Mydoom.R	0.994061	0.701047	0.981700
caphaw	0.826087	0.974359	0.999084

4.1.4 Zhodnotenie

Výsledky experimentov poukazujú na nedostatočnosť príznakov priamo dostupných z protokolu HTTPS. Vstupné dáta obsahujú množstvo chýbajúcich hodnôt, kvôli čomu je veľká časť príznakov nepoužiteľná. Niektoré z druhov malvéru ako *caphaw* alebo *Mydoom.R* sa dajú detegovať s pomerne dobrou presnosťou, kedže podľa zdrojov [16] a [17] ich funkčnosť spočíva v pripojení sa na vzdialený server, čo znamená vyšší počet pripojení na podozrivý server. Aj napriek prítomnosti príznakov globálnej znalosti sa nepodarilo nájsť spoľahlivý spôsob detekcie väčšiny dobre známych druhov malvéru. Preto sme sa

rozhodli presunúť naše úsilie na sľubnejšie, na protokole nezávisle definované príznaky, založené na správaní užívateľa, ktoré sme nazvali udalosti. Touto témou sa zoberá nasledujúca sekcia.

4.2 Experimenty s detekciou malvéru v komunikácii pomocou udalostí

Poskytnuté vstupné dáta majú obrovský objem a vysoko nevyvážené výstupné triedy. Ak ich chceme použiť na vytvorenie modelu pomocou algoritmov ako napríklad náhodné lesy, musíme ich najprv upraviť. Keďže nie je jednoduché získať viac dát komunikácie nakazených užívateľov, je dôležité zachovať všetky vzorky takejto komunikácie. Veľkú časť tvorí komunikácia s negatívnou triedou, z ktorej časť vynecháme tak, aby sme výsledný pomer tried mali 1:100.

Cieľom tejto sekcie je ukázať, že je možné pomocou udalostí odlíšiť užívateľov nakazených malvérom od nenakazených užívateľov pomocou algoritmov učenia pod dohľadom a nájsť optimálne nastavenie algorimu za účelom zlepšenia úspešnosti detekcie.

Implementácie použitých algoritmov pochádzajú z knižnice *scikit-learn*[18] jazyka Python.

4.2.1 Hodnotenie modelov

V tejto časti sme sa rozhodli vstupné dáta rozdeliť do množín po siedmych dňoch. Tým rozdelíme dáta za jeden mesiac do štyroch týždňov. Ďalej tieto týždne rozdelíme tak, aby trénovacie dáta boli tvorené dvoma týždňami a testovacie jedným. Medzi trénovacími a testovacími dátami vynecháme jeden týždeň preto, aby sme dokázali lepšie ohodnotiť generalizáciu modelu v detekcii malvéru. Model by sa totižto mohol preučiť, čiže by sa presne naučil tých užívateľov, ktorých komunikácia obsahovala rovnaké podmnožiny udalostí a ich štítok bol pozitívny v priebehu mesiaca. Komunikácia užívateľov sa totižto zo dňa na deň príliš nemení. Model by teda detegoval rovnakého nakazeného užívateľa na základe rovnakých udalostí, no my chceme, aby detegoval malvér všeobecne, čiže chceme nájsť udalosti špecifikujúce malvér.

Modely budú učené najprv na zjednotení tried. Tým sa prevedie problém učenia na binárny klasifikačný problém, na ktorom budú ladené parametre algoritmov. Po získaní optimálnych parametrov bude problem otestovaný v klasifikácii jednotlivých typov malvéru.

Algoritmus postupne dostane trénovacie a testovacie dáta podľa rozdelenia po týždňoch, s odstupom jedného týždňa medzi trénovacími a testovacími dátami. Po naučení modelu bude model otestovaný a bude vypočítané testovacie skóre PR AUC. Tiež bude vykreslená PR krivka pri vizuálne ohodnotenie.

V nasledujúcich sekciách je uvedené klasifikačné skóre jednotlivých kombinácií týždňov a krivka pre model trénovaný na prvom a druhom týždni a testovaný na treťom, ak nie je spomenuté ináč.

4.2.2 Modely s predvolenými parametrami

Na začiatok sme sa rozhodli pre vybudovanie klasifikačných modelov pomocou bežne používaných algoritmov strojového učenia, akými sú náhodné lesy, lineárne SVM a SVM s RBF jadrom. Rozhodli sme sa najprv použiť predvolené nastavenia algoritmov poskytovaných knižnicou *scikit-learn*, aby sme si vytvorili prehľad o ich schopnosti klasifikovať naše dáta. Porovnávame ich presnosť pomocou PR AUC a PR krivky a tiež porovnávame dobu trénovacej fázy.

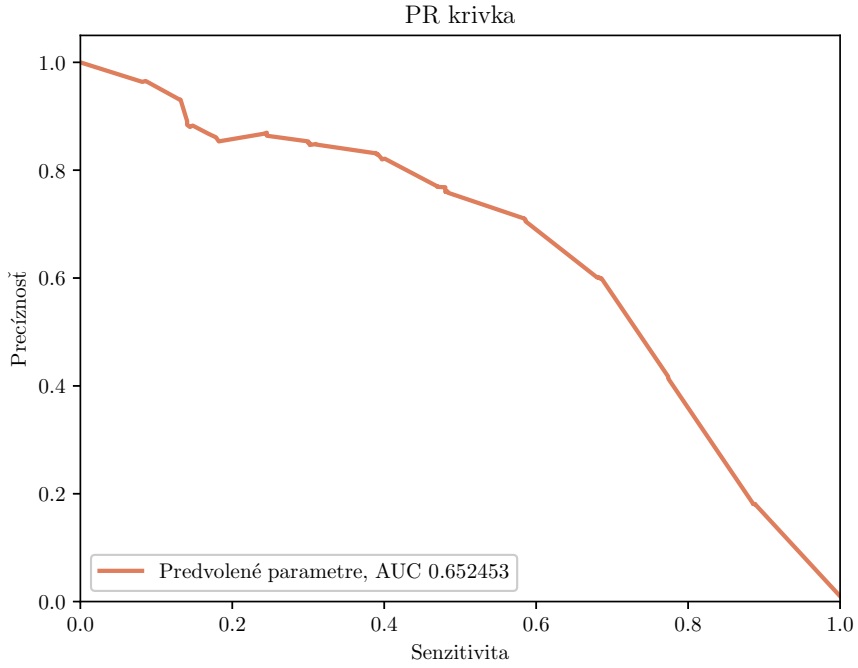
Náhodný les

Prvý experiment spočíva v natrénovaní binárneho klasifikátora pomocou náhodného lesa s predvolenými parametrami algoritmu. To poskytne dobrý štartovací bod, na ktorom je možné klasifikačné skóre ďalej zlepšovať. V predvolenom nastavení les pozostáva z desiatich stromov.

Tabuľka 2 ukazuje klasifikačné skóre pre jednotlivé kombinácie trénovacích a testovacích dát. Tabuľka 5 obsahuje dĺžku trénovania modelu a obrázku 2 môžeme vidieť PR krivku výsledného modelu. Na nej vidíme, že model klasifikuje pomerne dobre. V porovnaní s ostatnými algoritmami na obrázku 5, ktoré sme testovali, si náhodný les nepočína najlepšie, no ladením parametrov je možné skóre zlepšiť.

Tabuľka 2: Skóre klasifikácie náhodného lesa s predvolenými parametrami

Trénovacie týždne	Testovací týždeň	Skóre PR AUC
1,2	4	0.65524
3,4	1	0.65435
1,4	2	0.69769
1,4	3	0.68149



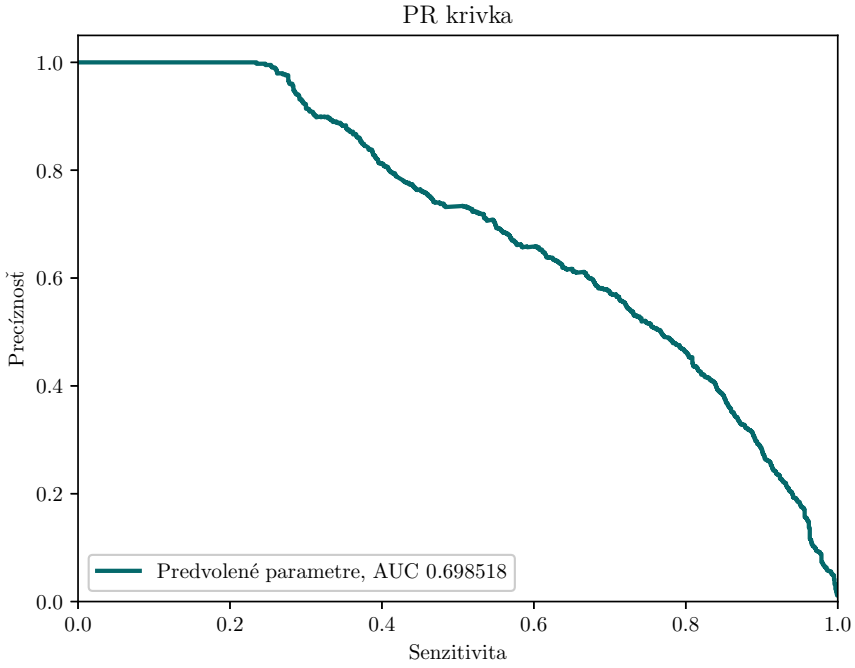
Obr. 2: PR krivka pre náhodný les s predvolenými parametrami

Lineárne SVM

Naše dáta pozostávajú z pomerne riedkych binárnych vektorov, čiže je možné, že lineárne SVM dokáže nájsť dobrú rozhodovaciu hranicu. Model vybudovaný s predvolenými parametrami dosahuje o trochu vyššie skóre ako náhodný les. Tabuľka 3 obsahuje skóre pre kombinácie týždňov a obrázok 3 PR krivku pre prvú kombináciu. Na obrázku 5 môžeme vidieť porovnanie PR kriviek, kde je viditeľné, že lineárne SVM si počína najlepšie, ak berieme do úvahy predvolené nastavenia algoritmov.

Tabuľka 3: Skóre klasifikácie SVM s lineárnym jadrom s predvolenými parametrami.

Trénovacie týždne	Testovací týždeň	Skóre PR AUC
1,2	4	0.69851
3,4	1	0.70806
1,4	2	0.71364
1,4	3	0.70215



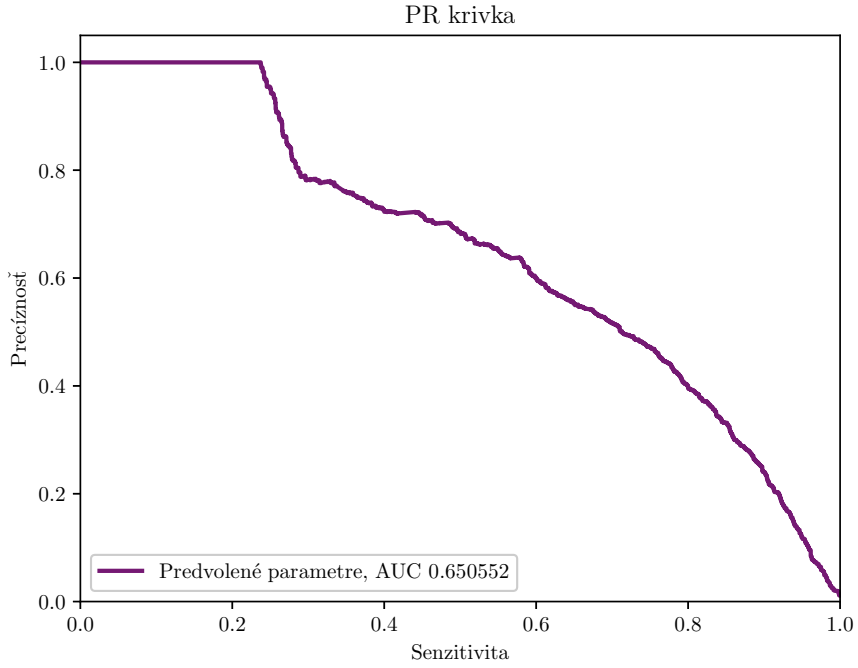
Obr. 3: PR krivka pre lineárne SVM predvolenými parametrami

RBF SVM

Algoritmus RBF SVM sa pokúša pomocou funkcie jadra RBF nájsť čo najlepšiu rozhodovacú hranicu, ktorá môže byť v priestore príznakov trénovacích dát nelineárna. Pri predvolených nastaveniach dosahuje približne rovnaké skóre ako náhodný les, čo môžeme vidieť na obrázku 5. Nevýhodou oproti náhodnému lesu je, že RBF SVM sa trénuje dlhú dobu v porovnaní s ostatnými v tabuľke 5.

Tabuľka 4: Skóre klasifikácie SVM s RBF jadrom s predvolenými parametrami

Trénovacie týždne	Testovací týždeň	Skóre PR AUC
1,2	4	0.65055
3,4	1	0.67197
1,4	2	0.64841
1,4	3	0.63615



Obr. 4: PR krivka pre SVM s RBF jadrom s predvolenými parametrami

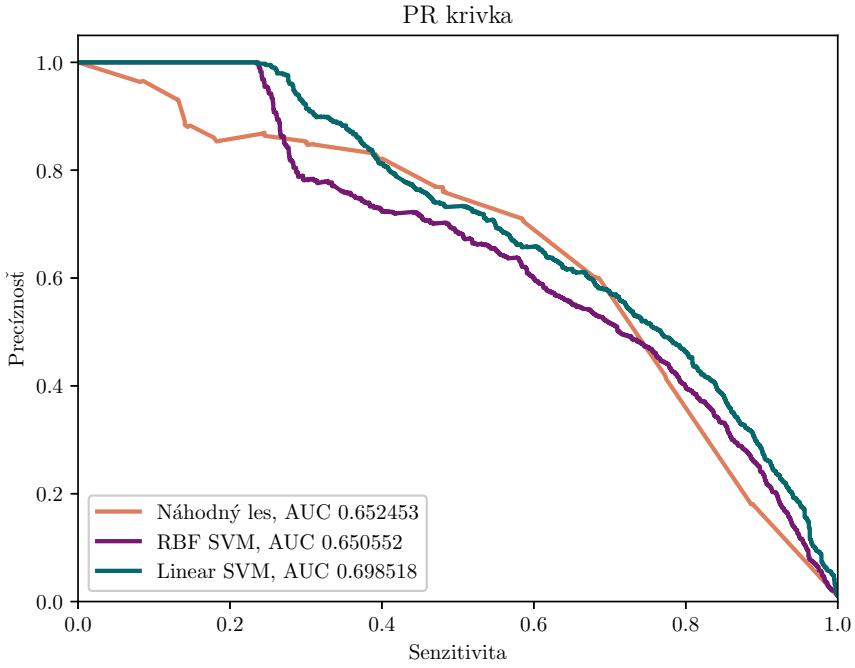
Tabuľka 5: Doba trénovania modelov podľa použitých algoritmov v sekundách

	Náhodný les	RBF SVM	Lineárne SVM
Doba trénovania (s)	7.23	1505.91	9.98

Zhodnotenie

Napriek tomu, že model naučený pomocou SVM s lineárnym jadrom dosiahol najlepších výsledkov, rozhodli sme sa ďalej vylepšiť skóre aj pre náhodný les. Jedným z dôvodov je poskytnutie miery dôležitosti príznakov, čo využijeme pri výbere relevantných príznakov.

RBF SVM sme sa rozhodli ďalej nepoužívať, kedže dosiahol najhoršieho výsledku a jeho trénovanie trvalo najdlhšiu dobu. Pomocou výberu príznakov a ladením parametrov sa pokúsime zlepšiť skóre pre lineárne SVM a náhodné lesy.



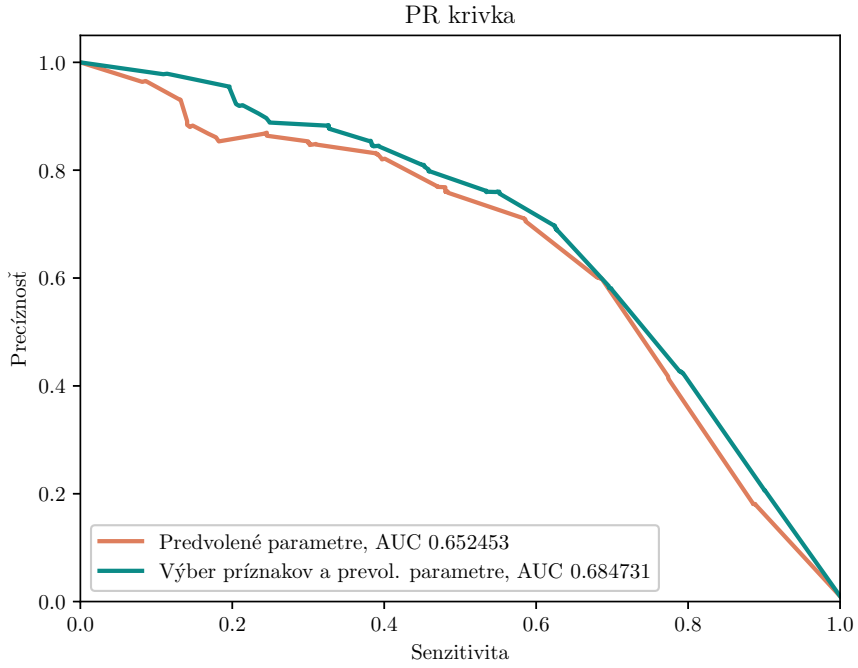
Obr. 5: PR krivka pre všetky testované algoritmy

4.2.3 Náhodný les s výberom príznakov

Vyskúšali sme využiť miery dôležitosti príznakov v náhodnom lese. Táto miera nám ukazuje, ako veľmi príznaky prispievajú ku správnemu odhadu triedy. Príznaky s nízkou dôležitosťou prispievajú k správnemu odhadu málo, čo môže znamenať, že nemajú žiaden vzťah s výstupnou triedou a ich použitie zanáša do modelu nechcený šum. Vybrali sme teda príznaky, ktorých dôležitosť bola nižšia ako priemer dôležitostí všetkých príznakov. Vyradili sme ich z trénovacej množiny a na upravenej trénovacej množine sme naučili náhodný les. Výber príznakov znížil počet príznakov v trénovacej množine z pôvodných 399 na 94, čo prinieslo skrátenie doby trénovania modelu.

Zníženie počtu príznakov na jednu štvrtinu pôvodného počtu ukazuje, že množstvo udalostí nemá vplyv na určenie prítomnosti malvéru v komunikácii užívateľa.

Na obrázku 6 môžeme vidieť mierne zlepšenie PR AUC vplyvom výberu príznakov.



Obr. 6: PR krivka pre náhodný les s výberom a bez výberu príznakov

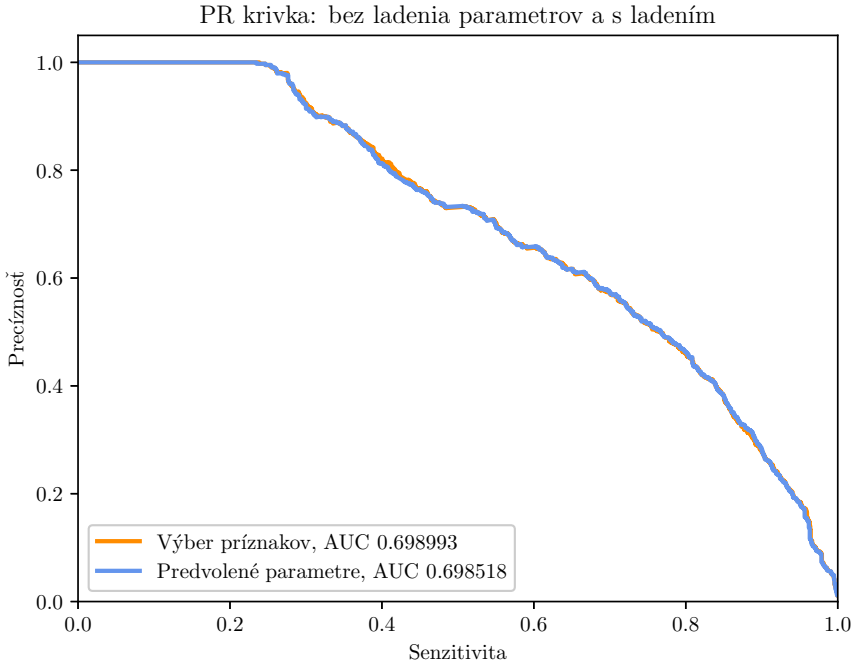
4.2.4 Lineárne SVM s výberom príznakov a ladením parametru C

Aplikácia výberu príznakov sa pri lineárnom SVM ukázala ako neúspešná v ohľade zlepšenia PR AUC. Počet príznakov sa síce výrazne znížil, no PR AUC sa zvýšila len zanedbateľne, čo môžeme vidieť na obrázku 7, na ktorom jednotilvé PR krivky takmer úplne splývajú. Rovnaký vplyv malo ladenie parametru C, pri ktorom sa skóre zlepšilo len pár desaťtisícin.

4.2.5 Náhodný les s výberom príznakov a ladením parametrov

Vyskúšali sme aj ladenie parametrov pomocou funkcie *randomized search*[21]. Táto funkcia skúša rôzne hodnoty parametrov algoritmu zo zadaného rozdelenia a vyberie kombináciu parametrov, ktorá maximalizuje PR AUC výsledného modelu.

[21] Na rozdiel od bežne používaneho *grid search*, ktorý skúša všetky kombinácie parametrov a trvá dlhú dobu, *randomized search* skúša n kombinácií parametrov definovaných pomocou rozsahov rozdelenia, čím šetrí čas a dosahuje podobne dobré výsledky.



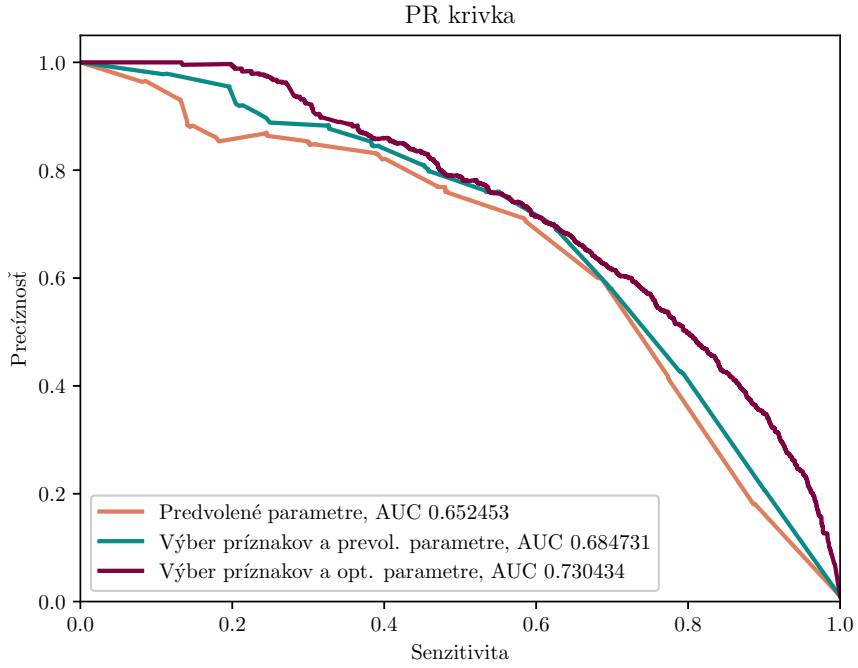
Obr. 7: PR krivka pre lineárne SVM výberom a bez výberu príznakov

Použitím nastavenia parametrov nájdeného pomocou *randomized search* a výberom najdôležitejších príznakov sme dosiahli ďalšie zlepšenie, ktoré je možné vidieť na obrázku 8. Ladili sme parametre pre rozhodovacie kritérium, počet príznakov použitých na rozdelenie, hĺbku jednotlivých stromov a minimálnu veľkosť listu.

4.2.6 Náhodný les učený na dátach za celý mesiac február

Rozhodli sme sa využiť výber príznakov a optimálne parametre náhodného lesa na vytvorenie modelu naučeného na dátach za mesiac február a testovaný na posledných dvoch marcových týždňoch. Výsledky tohoto experimentu môžeme vidieť na obrázku 9. Týmto experimentom sme chceli ukazať schopnosť modelu správne označiť vzorky komunikácie malvéru, ktoré pochádzajú výhradne z neskoršieho dátumu, ako trénovacie vzorky.

Možnou príčinou zníženia presnosti klasifikácie oproti predchádzajúcim experimentom je zmena sledovaných udalostí v dátach pre marec, pri čom boli odobrané udalosti, ktoré boli považované za nedostatočne informatívne a boli pridané nové udalosti.

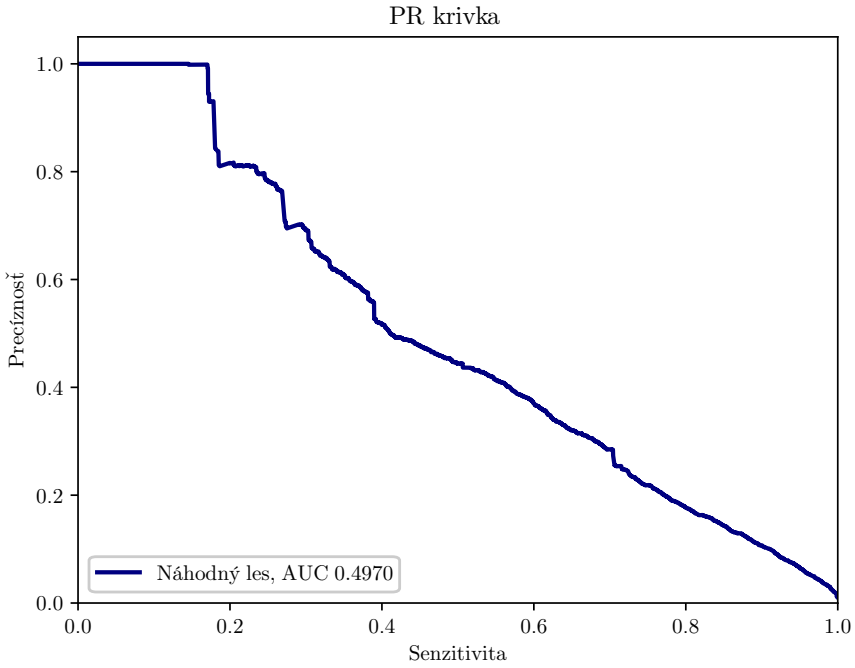


Obr. 8: PR krivka pre náhodný les pred a po aplikácii výberu príznakov a optimalizácii parametrov

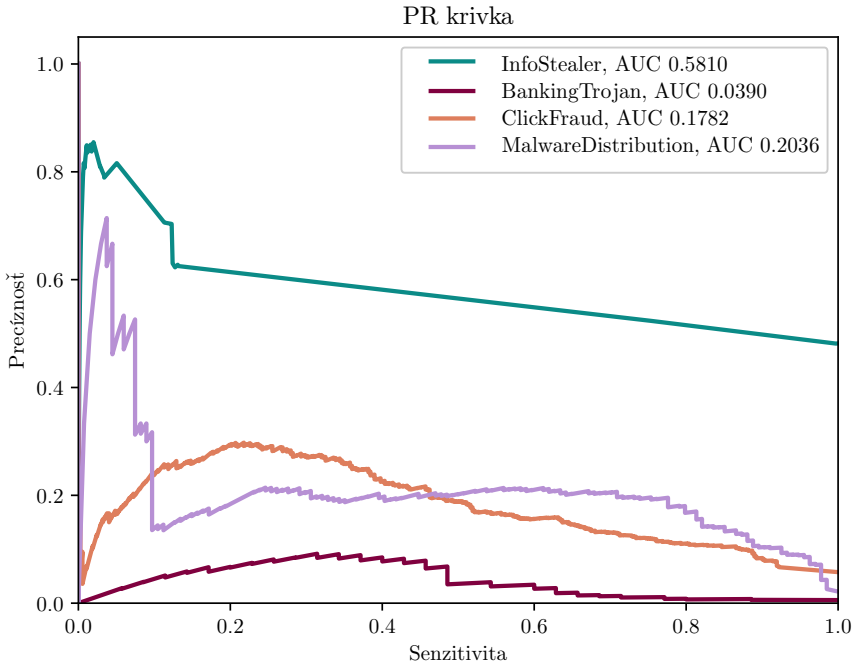
4.2.7 Náhodný les pre jednotlivé typy malvéru

Aby sme mohli porovnať výsledky experimentov tejto kapitoly s predchádzajúcou kapitolou, kde sme sa zaoberali detekciou malvéru v šifrovanej kamunikácii, otestujeme schopnosť odhaliť jednotlivé typy malvéru v dátach pozostávajúcich z udalostí, s ktorými pracujeme v tejto kapitole. Z dát za mesiac február sme vybrali sme približne 6 000 náhodných vzoriek z negatívnej triedy, ktoré sme použili pri testovaní jednotlivých druhov malvéru. K týmto vzorkám sme postupne pridali všetky pozitívne vzorky jedného z typov malvéru za február a použili sme ich na testovanie modelu naučenom na ostatných dátach. Týmto sme chceli zistiť ako dobre dokáže náš model detegovať typy malvéru, ktoré ešte nevidel.

Prvým testovaným typom bol *info stealer*, ktorého vzorky v testovacích dátach predstavovali približne polovicu. Výsledok testovania môžeme vidieť na obrázku 10. PR krivka ukazuje, že model dokáže odhaliť tento typ malvéru, no s pomerne malou presnosťou pre väčšinu z prahov. Podobný výsledok sme dosiahli aj pri ostatných typoch malvéru, čiže môžeme povedať, že jednotlivé typy sú z veľkej časti definované odlišnými množinami udalostí.



Obr. 9: PR krivka pre náhodný les pred a po aplikácii výberu príznakov a optimalizácii parametrov pri použití dát za február na trénovanie a posledných dvoch marcových týždňov na testovanie



Obr. 10: PR krivka pre náhodný les na detekciu jednotlivých typov malvéru

Záver

V rámci tejto práce sme testovali modely detekcie nakazených užívateľov v záznamoch sieťovej komunikácie. Tieto modely sme vytvárali pomocou algoritmov strojového učenia, pri ktorých sme popísali princípy ich fungovania. Algoritmy sme používali na učenie na dátach získaných zo šifrovanej komunikácie užívateľov a aj na dátach zostavených zo slabých indikátorov správania užívateľov.

Detekcia malvéru pomocou príznakov vytvorených z informácií, ktoré sa dajú zachytiť sledovaním šifrovanej komunikácie, sa ukázala pre množstvo typov malvéru ako neúspešná. Príčinou je to, že dáta, ktoré pochádzali od nakazených užívateľov boli zmesou bežnej komunikácie s komunikáciou malvéru, a pre potreby algoritmov učenia pod dohľadom sme ich museli všetky zaradiť do pozitívnej triedy. Ďalšou nevýhodou je fakt, že niektoré typy malvéru komunikujú po sieti len veľmi málo, takže ich detekcia pomocou záznamov komunikácie je veľmi nepresná. Napriek tomu sa nám podarilo v záznamoch šifrovanej komunikácie detegovať s pomerne vysokou úspešnosťou typy malvéru, ktoré sa vyznačujú tým, že sa pripájajú na určité servery s úmyslom odoslať ukradnuté dáta. Správanie týchto typov, ako časté pripájanie sa na podozrivý server a prenos veľkého objemu dát umožňuje dobré odlíšenie od neškodnej komunikácie.

Pri experimentoch na dátach, ktoré sú zostavené zo slabých príznakov správania užívateľov nazývaných udalosti sme prišli na niekoľko pozorovaní. Pri kombinácii všetkých tried, čiže všetkých typov malvéru do jednej triedy, sme dosiahli dobrú presnosť detekcie. Na týchto dátach sme porovnali presnosť detekcie malvéru pomocou algoritmov náhodný les, lineárne SVM a RBF SVM s predvolenými nastaveniami parametrov. Z toho síce vyšiel s najlepší PR AUC skóre algoritmus leneárne SVM, no pomocou výberu príznakov a ladením parametrov sme dokázali vylepšiť skóre náhodného lesa, ktorý sa nakoniec ukázal ako najpresnejší v detekcii. Otestovali sme aj schopnosť detekovať jednotlivé typy malvéru osobitne pomocou metódy *leave one out*, pri čom sa ukázalo, že jednotlivé typy malvéru sú špecifikované zväčša odlišnými množinami uda-

lostí, a teda detekcia nových typov malvéru pomocou ostatných typov nebola dostatočne úspešná.

Vzhľadom k povahe dát by do budúcna bolo dobré vyskúšať algoritmy učenia bez dohľadu, ako napríklad *isolation forest* alebo *one class SVM*, a otestovať ich úspešnosť v detekcii na našich dátach. Výsledky experimentov ukazujú, že dáta zostavené z udalostí sú v ohľade spracovania strojovým učením sľubné, a existuje množstvo postupov a rôznych algoritmov, ktorými by sa dalo klasifikačné skóre ďalej vylepšovať.

Literatúra

[1] Louppe, G.: *Understanding random forests: From theory to practice*. online, University of Liège, Faculty of Applied Sciences, Department of Electrical Engineering and Computer Science, 2014. Dostupné z: `https://arxiv.org/pdf/1407.7502v3.pdf`

[2] Boyd, K.; Eng, K. H.; Page, C. D.: Area under the precision-recall curve: Point estimates and confidence intervals [online]. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2013, s. 451–466, doi:10.1007/978-3-642-40994-3_29.

[3] Huang, B. F.; Boutros, P. C.: The parameter sensitivity of random forests. *BMC bioinformatics [online]*, ročník 17, č. 1, 2016: str. 331, ISSN 1471-2105.

[4] Criminisi, A.; Shotton, J.; Konukoglu, E.: *Decision Forests for Classication, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning* [online]. 2012, [cit. 2017-05-10]. Dostupné z: `http://www.ehu.eus/ccwintco/uploads/6/66/Random_Forest_Crim.pdf`

[5] Liaw, A.; Wiener, M.: Classification and Regression by randomForest. *R News*, ročník 2/3, 2002: s. 18–19, ISSN 1609-3631.

[6] Breiman, L.: Random Forests. *Machine Learning*, ročník 45, č. 1, 2001: s. 5–32, ISSN 1573-0565, doi:10.1023/A:1010933404324. Dostupné z: `http://dx.doi.org/10.1023/A:1010933404324`

[7] Kim, E.: *Everything You Wanted to Know about the Kernel Trick* [online]. 2013, [cit. 2017-05-10]. Dostupné z: `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

[8] Tong, S.; Koller, D.: Support vector machine active learning with applications to text classification. *Journal of machine learning research [online]*, ročník 2, č. Nov, 2001: s. 45–66.

[9] RTFM, Inc.: *HTTP Over TLS [online]*. 2000, [cit. 2017-04-22]. Dostupné z: https://tools.ietf.org/html/rfc2818

[10] Anderson, B.: Hiding in Plain Sight: Malware's Use of TLS and Encryption [online]. 2016, [cit. 2017-04-22]. Dostupné z: https://blogs.cisco.com/security/malwares-use-of-tls-and-encryption

[11] Anderson, B.; McGrew, D.; Kendler, A.: Classifying Encrypted Traffic with TLS-Aware Telemetry [online]. Cisco Systems, Inc., 2016, [cit. 2017-04-22]. Dostupné z: http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=449962

[12] RTFM, Inc.: *The Transport Layer Security (TLS) Protocol Version 1.2 [online]*. 2008, [cit. 2017-04-29]. Dostupné z: https://tools.ietf.org/html/rfc5246

[13] Oracle: *Persistent Connections [online]*. [cit. 2017-04-25]. Dostupné z: http://docs.oracle.com/javase/7/docs/technotes/guides/net/http-keepalive.html

[14] Raschka, S.: *About Feature Scaling and Normalization* [online]. 2014, [cit. 2017-05-08]. Dostupné z: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-standardization

[15] Saar-Tsechansky, M.; Provost, F.: Handling missing values when applying classification models. *Journal of machine learning research [online]*, ročník 8, č. Jul, 2007: s. 1623–1657.

[16] *Backdoor: Win32/Caphaw.A* [online]. [cit. 2017-05-11]. Dostupné z: https://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Backdoor%3aWin32%2fCaphaw.A

[17] *Worm: Win32/Mydoom.R@mm* [online]. [cit. 2017-05-11]. Dostupné z: https://www.microsoft.com/security/portal/threat/encyclopedia/Entry.aspx?Name=Worm:Win32/Mydoom.R@mm

[18] *Scikit-learn* 0.18.1 [software]. Dostupné z: http://scikit-learn.org/stable/

Dodatok **A**

Zoznam použitých skratiek

TP True Positive

FP False Positive

TN True Negative

FN False Negative

PR Precision-Recall

AUC Area Under Curve

TLS Transport Layer Security

SSL Secure Sockets Layer

MAC Message Authentication Code

SVM Support Vector Machine

RBF Radial Basis Function

Dodatok B

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `thesis.ps` ..... text práce vo formáte PS