PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené
ako otvorená cloudová služba s prvkami strojového učenia

V7 výsledok analýzy vhodných algoritmov

KPB3 – Návrh ML algoritmov

typ – dokumentácia

mesiac odovzdania – M10

Financované
Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD VLÁDY
SLOVENSKEJ REPUBLIKY

Obsah

1 Kategorizácia útokov v IP sieti 4
1.1 Útoky na sieťovú vrstvu (Network Layer Attacks) 4
1.2 Útoky na transportnú vrstvu (Transport Layer Attacks – OSI vrstva 4) 5
1.3 Útoky na aplikačnú vrstvu (Application Layer Attacks) 6
2 Analýza DoS a DDoS útokov 8
2.1 DoS (Denial of Service) 8
2.2 DDoS (Distributed Denial of Service) 9
3 Útočné taktiky IP útokov 10
3.1 Exploitácia 10
3.2 Laterálny pohyb 11
3.3 Eskalácia privilégií 12
4 Metódy a algoritmy strojového učenia 13
4.1 Rozdelenie algoritmov a metód na základe typu učenia 13
4.1.1 Supervised learning (učenie s dohľadom) 13
4.1.1.1 Klasifikácia 13
4.1.1.2 Regresia 14
4.1.1.3 K-Nearest Neighbor 16
4.1.1.4 Support Vector Machine 18
4.1.1.5 Decision Tree 19
4.1.2 Unsupervised learning (učenie bez dohľadu) 21
4.1.2.1 Klastrovanie (zhlukovanie) 21
4.1.2.2 K-means 22
4.1.2.3 Gaussian Mixture Model 23
4.1.2.4 DBSCAN 25
4.1.3 Redukcia dimenzionality 25
4.1.4 Detekcia anomálií 26
4.1.5 Association Rule-Mining 27
4.1.6 Semi-supervised learning (učenie s čiastočným dohľadom) 28
4.1.7 Reinforcement learning (učenie na základe odmeny) 29
4.2 Rozdelenie algoritmov a metód na základe typu útoku 30
4.2.1 Detekcia narušenia siete (Network Intrusion Detection (NIDS)) 30
4.2.2 Detekcia Malware 31
4.2.3 Detekcia Spam/Ham & Phishingu 31
4.2.4 Detekcia spoofingu 31
4.2.5 Detekcia botnetov 32
4.3 Rozdelenie algoritmov a metód podľa použitia neurónovej siete 32

4.3.1 Klasické algoritmy strojového učenia pre detekciu anomálií 33
4.3.1.1 Klasické algoritmy s učiteľom, bez učiteľa a s čiastočným učiteľom. 34
4.3.1.2 Vstupné údaje pre algoritmy – zdroje a príprava dát 34
4.3.1.3 Parametre algoritmov a tréningový proces 35
4.3.1.4 Odporúčania klasických ML metód na rozpoznanie rôznych útokov 35
4.3.1.5 Kritériá na porovnanie algoritmov 36
4.3.1.6 Odporúčaný algoritmus 36
4.3.2 Algoritmy strojového učenia používajúce neurónové siete 36
4.3.2.1 Princíp algoritmov hlbokého učenia 37
4.3.2.2 Princíp algoritmov hlbokého učenia 37
4.3.2.3 Vstupné údaje a ich príprava 37
4.3.2.4 Parametre algoritmov a trénovanie 37
4.3.2.5 Odporúčania metód s neurónovými sieťami na detekciu útokov 38
4.3.2.6 Kritériá pre porovnanie algoritmov 38
4.3.2.7 Odporúčaný algoritmus 38
5 Analýza štatistických parametrov na vybraných DDoS útokoch 39
5.1 Popis použitých záznamov DDoS útokov pre testovanie reakcie štatistických koeficientov 39
5.2 Použitie štatistických koeficientov 41
5.2.1 Variabilnosť, šikmosť a špicatosť 41
5.2.2 Detekčné hodnoty pre Autoregresiu a Koreláciu 44
5.2.3 Reakcia Divergencie na DDoS útok 46
5.3 Využitie 3σ-Tunela pre rozpoznanie DDOS útokov 48
5.4 Detekčné funkcie pre strojové rozpoznanie DDoS útokov 50
5.5 Záver 52
Prílohy 53

1 KATEGORIZÁCIA ÚTOKOV V IP SIETI

Jednotlivé typy hackerských útokov môžem kategorizovať podľa rôznych kritérií. Napr. podľa spôsobu útoku na **pasívne útoky**, ktoré sa snažia získať informácie bez zásahu do komunikácie (napr. odpočúvanie (sniffing), analýza prevádzky, atď.) a **aktívne útoky**, ktoré zasahujú do sieťového prenosu alebo modifikujú dáta (napr. MITM, DoS, spoofing, atď.). Sieťové útoky môžeme ďalej deliť podľa rôznych cieľov:

- **DoS/DDoS útoky –** Cieľom je vyradiť službu z prevádzky zahltením požiadavkami.
- **Sniffing (odpočúvanie)** – Zber paketov zo siete s cieľom získať citlivé informácie.
- **MITM (Man-in-the-Middle)** – Útočník sa postaví medzi dve komunikujúce strany a odpočúva/modifikuje komunikáciu.
- **Port scanning a reconnaissance útoky** – Skenovanie siete na vyhľadanie otvorených portov a zraniteľností.
- **Útoky na DNS** – Napr. DNS cache poisoning, DNS spoofing.

Jedným z najvýznamnejších delení sieťových útokov je podľa časti sieťovej infraštruktúry, v ktorej k útoku došlo. Nasledujúcom prehľade jednotlivé typy útokov len stručne popíšeme, niektorými vybranými závažnými útokmi sa budeme venovať neskôr v nasledujúcich kapitolách.

1.1 Útoky na sieťovú vrstvu (Network Layer Attacks)

Útoky na sieťovú vrstvu (Layer 3 v OSI modeli) sa zameriavajú na sieťovú infraštruktúru a protokoly ako IP, ICMP, alebo samotnú adresáciu a smerovanie. Ciele útokov na sieťovej vrstve je narušiť dostupnosť služby, obísť bezpečnostné opatrenia ako sú firewally, získať prístup k dátam cez manipuláciu so smerovaním, zakrývanie identity útočníka.

IP Spoofing

Útočník falšuje (spoof-uje) IP adresu odosielateľa. Cieľom je skryť identitu alebo sa tváriť ako dôveryhodný zdroj. Využíva sa napríklad pri **DoS/DDoS** útokoch alebo **Man-in-the-Middle** útokoch.

ICMP Flood/Ping Flood

Útočník zaplavuje cieľ veľkým množstvom ICMP požiadaviek (ping). Dochádza k preťaženiu systému alebo siete.

Smurf Attack

Špecifický typ ICMP útoku. Útočník pošle ICMP echo request na broadcast-ovú adresu siete s falošnou IP adresou obete. Všetky odpovede smerujú späť na obeť, čo ju preťaží.

Fragmentation Attacks

Manipulácia s fragmentáciou IP paketov. Útočník pošle neštandardne fragmentované pakety, ktoré cieľový systém nedokáže správne poskladať. Môže viesť k preťaženiu systému alebo vynechaniu detekcie firewallom.

Routing Table Poisoning

Zneužitie smerovacích protokolov (napr. RIP, BGP) na vloženie falošných informácií. Dôsledok: nesprávne smerovanie paketov, presmerovanie cez zariadenie útočníka (MITM), alebo znefunkčnenie siete.

TTL Expired Attacks

Využívajú nastavenie TTL (Time To Live) hlavičky IP paketu. Môžu sa použiť na **trasovanie siete** (napr. traceroute), ale aj ako súčasť zložitejších útokov na infraštruktúru.

1.2 Útoky na transportnú vrstvu (Transport Layer Attacks – OSI vrstva 4)

Transportná vrstva (Layer 4 v OSI modeli) zabezpečuje spoľahlivé alebo nespoľahlivé doručovanie dát medzi koncovými bodmi komunikácie najmä pomocou protokolov TCP (Transmission Control Protocol – spoľahlivý prenos) a UDP (User Datagram Protocol – nespoľahlivý prenos). Útoky na tejto vrstve sa zameriavajú na zneužitie spôsobu, akým tieto protokoly fungujú. Cieľom útokov je znefunkčnenie služieb alebo preťaženie systémov (DoS, DDoS), získanie kontroly nad reláciou (session hijacking), obídenie bezpečnostných mechanizmov a získanie informácií o otvorených portoch a službách

TCP SYN Flood

Najznámejší útok na TCP. Útočník posiela veľké množstvo požiadaviek na nadviazanie TCP spojenia (SYN pakety), ale nikdy ho nedokončí. Server drží otvorené „polospojenia“ a nakoniec vyčerpá pamäť alebo počet dostupných pripojení. Server prestane odpovedať legitímnym klientom.

TCP RST Attack (Reset Attack)

Útočník posiela falošné TCP RST pakety, čím násilne ukončuje spojenia medzi dvoma stranami. Môže sa použiť na narušenie komunikácie medzi dvoma systémami (napr. medzi klientom a serverom).

TCP Session Hijacking

Útočník prevezme už nadviazané TCP spojenie medzi dvoma stranami. Ak vie odhadnúť sekvenčné čísla TCP, môže sa do komunikácie „vložiť“ a prevziať kontrolu nad reláciou. Využíva sa na Man-in-the-Middle alebo injekciu dát.

UDP Flood

Útočník posiela veľké množstvo UDP paketov na náhodné porty cieľa. Cieľová služba zvyčajne odpovedá ICMP „Destination Unreachable“. Nadmerná prevádzka zaťažuje CPU a sieť.

UDP-based Amplification Attacks

Útočník posiela malé požiadavky na otvorené služby (napr. DNS, NTP), pričom ako odosielateľa nastaví IP obete. Služba odpovie obeti násobne väčším paketom – vzniká tzv. amplifikácia (zosilnenie). Používa sa pri DDoS útokoch.

Port Scanning

Aj keď nie je priamo útok, je to forma prieskumu (reconnaissance). Útočník skenuje porty pomocou nástrojov ako Nmap, aby zistil, aké služby bežia. Často predchádza ďalším útokom.

1.3 Útoky na aplikačnú vrstvu (Application Layer Attacks)

Aplikačná vrstva (Layer 7 v OSI modeli) je najbližšie k používateľovi a zahŕňa rôzne služby ako webové aplikácie, e-mail, DNS, FTP, atď. Útoky na tejto vrstve sa zameriavajú priamo na zraniteľnosti konkrétnych aplikácií a služieb. Často sú sofistikované, cielené a ťažko detekovateľné, pretože pripomínajú bežnú používateľskú aktivitu. Ciele útokov na aplikačnej vrstve je kompromitácia dát (krádež, zmena, mazanie), prevzatie kontroly nad systémom, získanie prístupových údajov, obídenie autentifikácie resp. autorizácie, znefunkčnenie služby (cez aplikačný DDoS)

HTTP Flood

Útok je prevádzaný formou DoS/DDoS. Útočník generuje veľké množstvo legitímnych HTTP požiadaviek (napr. GET/POST). Zámerom je zahltiť webový server výpočtovo náročnými požiadavkami. Na rozdiel od sieťových útokov je ťažšie rozpoznať, že ide o útok, pretože požiadavky sú platné.

SQL Injection

Útočník vkladá škodlivý SQL kód do vstupných polí (napr. prihlasovacie meno). Ak aplikácia nefiltruje vstup, útočník môže: čítať, meniť alebo mazať dáta z databázy, získať administrátorské práva.

Cross-Site Scripting (XSS)

Útočník vkladá škodlivý skript (často JavaScript) do webovej stránky, ktorý sa vykoná v prehliadači obete

Cross-Site Request Forgery (CSRF)

Útočník zmanipuluje používateľa, aby vykonal akciu, ktorú nechcel (napr. prevod peňazí), keď je prihlásený v inej aplikácii. Príklad: škodlivý odkaz, ktorý obeti „potichu“ pošle požiadavku na zmenu hesla.

Command Injection

Vloženie systémového príkazu do vstupného poľa aplikácie. Ak aplikácia spúšťa shell príkazy s neoverenými vstupmi, útočník môže vykonať príkazy v systéme.

Directory Traversal (Path Traversal)

Útočník sa snaží pristúpiť k súborom mimo povoleného adresára.

DNS Amplification / DNS Hijacking

Útoky na aplikačné služby ako DNS:

- Amplification – zneužitie DNS servera na DDoS.
- Hijacking – manipulácia s DNS záznamami na presmerovanie obetí.

Email-based útoky (Phishing, Spoofing)

Útočník zasiela e-maily, ktoré sa vydávajú za dôveryhodné napríklad s cieľom získať prihlasovacie údaje, šíriť malvér, získať prístup k firemným systémom.

Typ útoku	Popis	Príklady útokov
1. DoS/DDoS útoky	Preťaženie služby s cieľom znemožniť prístup legitímnym používateľom.	SYN Flood, UDP Flood, HTTP Flood
2. Sniffing (odpočúvanie)	Neoprávnené zachytávanie sieťovej komunikácie.	Packet sniffing (napr. Wireshark)
3. Spoofing	Falošná identita v sieťovej komunikácii (IP, MAC, DNS, e-mail).	IP Spoofing, ARP Spoofing, DNS Spoofing
4. Exploitácia	Zneužitie zraniteľnosti v systéme alebo aplikácii.	Buffer Overflow, SQL Injection, RCE
5. MITM (Man-in-the-Middle)	Útočník sa vloží medzi obe strany komunikácie.	SSL Stripping, Session Hijacking
6. Malware útoky	Infikovanie systému škodlivým softvérom.	Trójsky kôň, ransomware, spyware
7. Port scanning	Skenovanie portov na zisťovanie otvorených služieb.	Nmap, Zenmap
8. Social engineering	Manipulácia používateľa na získanie prístupu alebo informácií.	Phishing, pretexting, baiting
9. ARP/DNS útoky	Zneužitie slabín v ARP alebo DNS protokole.	ARP Poisoning, DNS Cache Poisoning
10. Privilege escalation	Získanie vyšších oprávnení v systéme.	Exploit kernelu, zlé nastavenia práv

Tabuľka 1: Typy IP útokov

2 ANALÝZA DOS A DDOS ÚTOKOV

Techniky DoS (angl. *Denial of Service*) a DDoS (angl. *Distributed Denial of Service*) sa vyskytujú pri rôznych typoch napadnutia bezpečnosti v IP sieti a pri incidentoch na rôznych úrovniach siete (vid predchádzajúca kapitola).

2.1 DoS (Denial of Service)

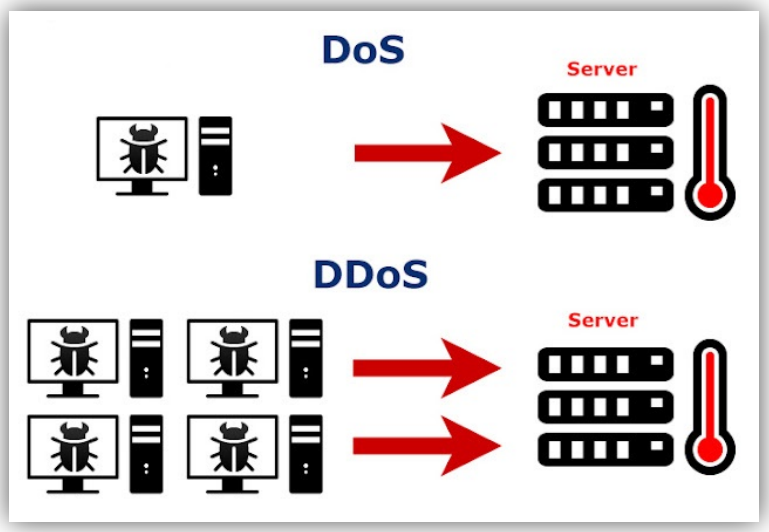
Skratka DoS v preklade znamená útok odmietnutia nejakej služby. Tento typ útoku sa používa na zahltenie servera. V tomto prípade útočník využíva zariadenie, ktorým sa pokúša narušiť normálnu prevádzku cieľového servera, služby alebo siete. Veľmi rýchlo sa stali hrozbou kybernetickej bezpečnosti. Sú jednoduché avšak ich dopad môže byť pre jedincov ale aj spoločnosti devastujúci.

DoS útoky sa často používajú pre vypnutie zariadení a sietí a existuje veľa rôznych spôsobov využitia spomínaných DoS útokov. Patria sem napr.:

- **Pretečenie vyrovnávacej pamäte** (angl. *buffer overflow*)*:* Vyrovnávacia pamäť je dočasné úložisko pre ukladanie dát. Ak program alebo systémový proces umiestni viac údajov ako bolo vopred určené, ďalšie údaje potom pretečú a môže sa stať, že niektoré z nich potom uniknú do iných vyrovnávacích pamätí. To má potom za následok poškodenie alebo prepísanie ďalších údajov. Útočník dokáže na základe tohto princípu poškodiť súbory, zmeniť alebo odhaliť rôzne informácie.
- **ICMP útok** (angl. *ICMP attack* alebo *Ping of Death*): Tento typ útoku využíva ICMP protokol pričom sa využívajú ICMP echo-requesty tzv. pingy. Všeobecne sa echo-requesty a echo-reply používajú na ping sieťového zariadenia s cieľom overiť konektivitu medzi zariadením a odosielateľom. Ak by útočník zahltil cieľ paketmi požiadaviek, sieť musí odpovedať rovnakým počtom paketov, čo spôsobí nedostupnosť zariadenia pre iné prichádzajúce požiadavky.
- **UDP útok** (angl. *UDP attack*): Útok je založený na posielaní UDP paketov so zdrojovou IP adresou, ktorú môže útočník sfalšovať a snahou je zahltiť náhodné porty na cieľovom servery. Výhodou UDP protokolu je, že nepotrebuje nadviazať spojenie, ako to je v prípade protokolu TCP. V prípade, že cieľový server príjme UDP paket na porte, ktorý nie je využívaný žiadnou aplikáciou, informuje o tom odosielateľa. Pokiaľ útočník odošle dostatočné množstvo paketov na takýto port, môže byť preťažený.
- **SYN útok** (angl. SYN flood): Toto je útok, ktorý vzniká pri vytváraní TCP spojenia medzi 2 zariadeniami. TCP využíva trojfázové potrasenie rúk (angl. *3-way handshake*). Nadviazanie spojenia začína odoslaním správy SYN. Ak je server aktívny odošle sa SYN/ACK správa späť ku klientovi, ktorí chcel nadviazať spojenie. Server sa snaží odpovedať dokým nevyprší čas na nadviazanie komunikácie, pričom si údaje uchováva v pamäti. Ak je klientom útočník s falošnou a nedostupnou zdrojovou adresou, zaslanie veľkého množstva takýchto paketov môže spôsobiť vyčerpanie pamäte, ktorá je určená pre TCP spojenia a všetky ďalšie pokusy o komunikáciu budú ďalej ignorované.
- **Útok slzy** (angl. *Teardrop attack*): Počas tohto útoku sa útočník snaží posielať fragmentované pakety a sieť sa potom snaží skompilovať tieto dáta do ich pôvodného stavu. Útočník môže využiť zraniteľnosť zastaraného operačného systému. Je bežné, že mnohé spoločnosti stále využívajú staršie verzie systémov z dôvodu opätovného spúšťania starej aplikácie a keďže niektoré verzie operačných systémov ako sú napr.: Linux či Windows obsahujú chybu pri spätnom zostavení fragmentovaného paketu, server môže byť práve kvôli tejto zraniteľnosti napadnutý.

2.2 DDoS (Distributed Denial of Service)

Na rozdiel od jednoduchého DoS útoku, DDoS útok (*Distributed Denial of Service*) naberá na sile, pretože útočník využíva viacero systémov naraz a zvyšuje tak závažnosť cielených útokov. Útočník môže použiť malware, ktorý odošle po sieti do zariadení so zámerom ich infikovať. Stanú sa tak súčasťou tzv. botnetu a podľa príkazov útočníka pomocou riadiaceho servera títo boti potom jednotne útočia na cielený server. DDoS útoky je veľmi ťažké odhaliť, pretože útočiaci boti môžu byť rozmiestnený na rôznych miestach po svete. Hlavný a kľúčový rozdiel medzi DoS a DDoS útokom je, že pri DDoS s použitím botov môže útočník posielať obrovské objemy prevádzky.



Obr. 1: Rozdiel medzi DoS a DDoS

DDoS útoky sa rozdeľujú do 3 hlavných kategórií:

1. **Objemový útok** (angl. *Volumetric attack*): Tento typ využíva zraniteľné služby ako sú napr.: NTP, DNS či SSDP. Útočník sa snaží zahltiť cieľ veľkým objemom paketov, na ktoré sa cieľ snaží odpovedať, až kým nedôjde ku kolapsu na strane cielenej sieťovej infraštruktúry. Zvyčajne to je širokopásmový útok, ktorý dokáže presiahnuť 100 GB či dokonca niekoľko TB za sekundu a je okamžite očividný pre poskytovateľov pripojenia. Na detekciu objemových útokov sa osvedčilo monitorovanie siete, analyzovanie toku a prietoková telemetrická analýza.
2. **Protokolový útok** (angl. *Protocol attack*): Tento typ útoku sa snaží vyčerpať zdroje servera alebo jeho systémov a spolieha sa na zraniteľnosti internetových komunikačných protokolov. Medzi protokolové útoky patria napr.: SYN útok, UDP útok, ICMP útok. Tieto útoky sa považujú za dostatočne ochromujúce na cielenú infraštruktúru vďaka frekvencii trvania. V roku 2018 útočníci využili BGP protokol na presmerovanie prevádzky, ktorá bola určená pre službu „MyEtherWallet“, čo však bola iba zásterka pre krádež kryptomenových peňaženiek. Trvalo to približne 2 hodiny.
3. **Útok na aplikačnú vrstvu** (angl. *Application layer attack*): Tento útok sa zameriava na konkrétnu aplikáciu a jej zraniteľnosti, aby nebola schopná spracovávať, odosielať či prijímať dáta. Útoky sú slabé a pomalé a sústreďujú sa na požiadavky GET a POST. Cieľom je potom ochromiť webový server a meria sa v požiadavkách za sekundu (angl. *Requests per second - Rps*). Sú nízke a slabé preto, aby boli v súlade s protokolom.

Existuje niekoľko znakov, ktoré nám dokážu pomôcť pri odhalení, či je naša sieť alebo aplikácia alebo celá infraštruktúra pod DDoS útokom. Môže to byť náhly nárast webovej aktivity na serveri, znížený výkon siete alebo náhla neschopnosť komunikovať s aplikáciou.

3 ÚTOČNÉ TAKTIKY IP ÚTOKOV

3.1 Exploitácia

Exploitácia je fázou útoku, počas ktorej útočník cielene využíva zraniteľnosti, chybné konfigurácie či slabé miesta vo softvéri (OS, aplikácie, sieťové služby) alebo v ľudskom faktore (sociálne inžinierstvo), aby získal neoprávnený prístup alebo spustil škodlivý kód. Stretávame sa napríklad s technikami, ktoré zahŕňajú exploitáciu vzdialených služieb, klientskych aplikácií, webových rozhraní alebo systému na obranu (sandbox, antivírus).

Exploitation of Remote Services (T1210)

Využitie zraniteľných sieťových služieb, ktoré bežia na vzdialených serveroch (napr. nezaplátované verzie SMB, RDP, HTTP, FTP, databázové služby). Útočník sa snaží získať prístup cez sieť tak, že zneužije verejne známe zraniteľnosti (CVE) alebo 0-day zraniteľnosti.

1. **Drive-by Compromise** (T1189)

Napadnutá alebo škodlivá webstránka obsahuje exploit kit, ktorý pri návšteve obete automaticky skúša využiť zraniteľnosti v prehliadači (Internet Explorer, Chrome, Firefox) alebo jeho pluginoch (Flash, Java, Silverlight). Zámerom je stiahnuť a spustiť malvér bez zásahu používateľa.

2. **Exploitation for Client Execution** (T1203)

Otvorenie dokumentu (Office, PDF), spustenie škodlivého prílohového skriptu alebo inštalácia makra, ktoré využije chybu v softvéri obete. Príkladom sú zneužitia makier vo Worde alebo chyby v PDF čítačkách.

3. **Supply Chain Compromise** (T1195)

Útočník sa zameriava na zraniteľnú časť reťazca dodávateľov (napr. vývojár, distribútor softvéru), aby infikoval legitímne aplikácie a tie následne využil u koncových používateľov. Klasickým prípadom je “trojanizovaný” inštalačný balík, ktorý obsahuje škodlivý kód s exploitom.

4. **Hardware/Firmware Exploits**

Menej časté, ale závažné techniky, kedy útočník zneužíva slabiny v implementácii firmvéru (BIOS, UEFI) alebo chyby v CPU (napr. Meltdown, Spectre). Umožňuje to prístup na veľmi nízkej úrovni (tzv. ring-0) a môže viesť k pretrvávajúcemu ovládnutiu systému bez povšimnutia antivírusov.

5. **Zero-day Exploits**

Využitie doposiaľ neznámych zraniteľností pre dodávateľa softvéru alebo bezpečnostnú komunitu. Pretože neexistuje záplata (patch), je detekcia a mitigácia takýchto exploitov extrémne náročná a často vyžaduje behaviorálne, heuristické alebo AI/ML techniky.

6. **Exploitation for Defense Evasion** (T1211)

Útočník využije zraniteľnosti v bezpečnostnom softvéri (napr. antivírus, EDR, sandbox) alebo v samotnom operačnom systéme, aby obišiel ochranné mechanizmy. Typickým príkladom je zneužitie chyby v ovládačoch podpísaných dôveryhodným certifikátom.

3.2 Laterálny pohyb

Laterálny pohyb (*Lateral Movement*) je taktika, pri ktorej sa útočník po úvodnom narušení bezpečnosti, napr. získaním prístupu na dané zariadenie, sa snaží presúvať v rámci siete a rozširovať ak svoj prístup a kontrolu nad ďalšími systémami. Cieľom je často eskalovať privilégia (napr. získať vyššie práva), spravovať kľúčové služby (napr. doménové radiče, SMB share) alebo pristupovať k citlivým dátam.

1. **Remote Services** (T1021) – rozšírený pohľad

Popri bežných RDP, SMB, SSH a WinRM sa v novších prípadoch sleduje aj zneužitie cloudových služieb či nástrojov na vzdialenú správu tretích strán (napr. VNC, TeamViewer).

Zdôrazňuje sa význam monitorovania netradičných časov a anomálnych geolokačných prístupov.

2. **Kerberoasting** (T1208)

Útočník získava Kerberos Service Tickets a následne sa pokúša prelomiť heslo offline (hashovanie pomocou RC4_HMAC_MD5).

Kombinácia s overpass-the-hash alebo pass-the-ticket ešte zjednodušuje šírenie v rámci Windows domén.

3. **Golden Ticket/Silver Ticket Attacks**

Sú špecifické pre Kerberos – Golden Ticket umožňuje útočníkovi vytvárať vlastné lístky TGT (Ticket-Granting Ticket), čím získa takmer neobmedzený prístup.

Silver Ticket je lístok TGS (Ticket Granting Service) špecifický pre určitú službu, čo takisto umožňuje neoprávnený prístup, avšak s užším rozsahom.

4. **Credential Stuffing a Overpass-the-Hash**

Credential Stuffing využíva hromadné prihlasovacie údaje z únikov dát a testuje ich platnosť v rámci cieľovej infraštruktúry.

Overpass-the-Hash funguje podobne ako pass-the-hash, avšak namiesto presunu NTLM hashu do LSASS sa generuje Kerberos ticket priamo z NTLM/RC4 hashu.

5. **Privileged Access Management (PAM) Bypass**

Postupy, ktoré obchádzajú systémy určené na správu privilegovaných účtov (cyberark, centrálne trezory hesiel a pod.).

Často ide o zneužitie nedostatkov v integračnom rozhraní, nedôsledné logovanie alebo prácu s temporárnymi tokenmi, ktoré PAM nepostrehne.

6. **Lateral Tool Transfer** (T1570)

Okrem tradičných nástrojov (PsExec, Mimikatz) sa objavujú prenosy vlastných exploit frameworkov (napr. Impacket, Cobalt Strike Beacon).

Dôležitá je segmentácia siete a kontrola integrácie EDR/AV sietí, aby sa odhalili pokusy o sťahovanie veľkých binárnych súborov či skriptov medzi segmentmi.

7. **Skripty a pamäťové techniky**

Zneužívanie PowerShell (T1059.001), WMI (T1047) či Dynamic Data Exchange (T1173) umožňuje spúšťať kód v pamäti bez uloženia na disk, čo sťažuje detekciu.

Najnovšie prístupy používajú kombinácie obfuskácie a legitímnych API volaní na zamaskovanie škodlivého obsahu.

3.3 Eskalácia privilégií

Eskalácia privilégií je taktikou, pri ktorej útočník po získaní prístupu k systému či sieti hľadá spôsoby, ako získať vyššiu úroveň oprávnení alebo administrátorské práva. Cieľom je rozšíriť možnosti manipulácie so systémom, obísť bezpečnostné obmedzenia, prípadne vykonávať zmeny v kľúčových častiach infraštruktúry (napr. doménový radič, databázové servery, firemné aplikácie).

1. **Exploitation for Privilege Escalation** (T1068)

Zneužitie zraniteľností v OS (Windows, Linux, macOS) alebo v ovládačoch, aby sa z procesu s nízkymi oprávneniami stal proces s vyšším oprávnením. Medzi bežné príklady patria zneužitia lokálnych eskalačných zraniteľností (napr. CVE v kerneloch, ovládačoch tlačiarní a pod.).

2. **Bypass User Account Control** (UAC) (T1548.002)

UAC vo Windows bráni bežnému používateľovi vykonávať administrátorské operácie bez dodatočného potvrdenia. Útočníci často používajú rôzne metódy obchádzania UAC, napr. cez falošné binárne súbory, úpravu registrov, zneužitie legitímnych Windows utilít (CMSTP, schtasks, Event Viewer a pod.).

3. **Access Token Manipulation** (T1134)

V prostredí Windows sa využívajú prístupové tokeny (access tokens) na identifikáciu a oprávnenia procesov. Útočník môže klonovať, vytvárať či upravovať token tak, aby predstieral iného používateľa (napr. administrátora) a získal prístup k ďalším zdrojom.

4. **Credential Dumping** (T1003)

Získavanie prihlasovacích údajov (hashov, hesiel, ticketov) z pamäte (LSASS vo Windows), registrov, SAM databázy, či iných úložísk. Nástroje ako Mimikatz, Impacket, alebo crackmapexec následne umožňujú eskalovať privilégiá či získať prístup k iným účtom v doméne.

5. **Scheduled Task/Job** (T1053)

Vytváranie či úprava plánovaných úloh (napr. cez Windows Task Scheduler alebo cron v Linuxe) s cieľom spustiť škodlivý kód s vyššími oprávneniami. Útočník využije legitímne funkcie systému, avšak úlohu naplánuje tak, aby bežala s administrátorskými právami.

6. **DLL Search Order Hijacking** (T1574.001)

Operačný systém Windows najprv vyhľadáva DLL knižnice v adresároch s vyššou prioritou. Ak útočník vie, kde a ako sa DLL načítajú, môže umiestniť škodlivú DLL s rovnakým názvom na miesto s vyššou prioritou v ceste, čím sa podarí spustiť kód s vyššími oprávneniami.

7. **SetUID a SetGID v Unix/Linux** (T1548.001)

Útočník môže zneužiť binárne súbory s nastavenými SUID bitmi (napr. /bin/su alebo iné) či zle nastavené skripty, aby získal root oprávnenia. Podobne môžu útočníci využiť chyby v konfigurácii sudo (napr. ak sú v sudoers definované nežiaduce výnimky).

8. **Abusing Application Shims** (T1138)

Vo Windows existuje funkcia Application Compatibility Shims, ktorá umožňuje aplikáciám bežať v kompatibilnom režime. Útočníci môžu zneužiť túto funkcionalitu na spustenie kódu s vyššími oprávneniami, ak je shim nesprávne nastavený.

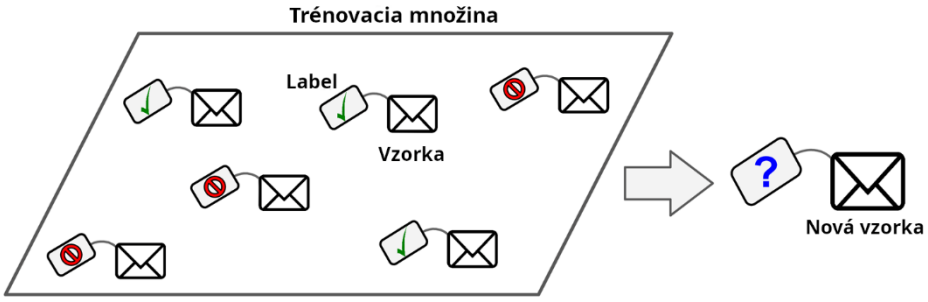
4 METÓDY A ALGORITMY STROJOVÉHO UČENIA

Predmetom tejto kapitoly je analýza metód strojového učenia – metód a algoritmov ML (Machine Learning), ktoré sa používajú pre detekciu anomálií v sieťovej prevádzke.

4.1 Rozdelenie algoritmov a metód na základe typu učenia

4.1.1 Supervised learning (učenie s dohľadom)

Metódy alebo algoritmy učenia s dohľadom zahŕňajú algoritmy učenia, ktoré prijímajú vzorky údajov (známe ako trénovacie údaje) a súvisiace výstupy (známe ako značka/label alebo odpoveď) s každou vzorkou údajov počas procesu trénovania modelu. Hlavným cieľom je naučiť sa mapovanie alebo asociáciu medzi vzorkami vstupných údajov x a ich zodpovedajúcimi výstupmi y na základe viacerých inštancií alebo vzoriek trénovacích údajov. Tieto naučené znalosti sa potom môžu v budúcnosti použiť na predpovedanie výstupu y pre akúkoľvek novú vzorku vstupných údajov x, ktorá bola predtým neznáma alebo nedostupná počas procesu trénovania modelu (obr.2).



Obr. 2: Príklad učenia s dohľadom – označkovaná trénovacia množina pre úlohu detekcie spamu.

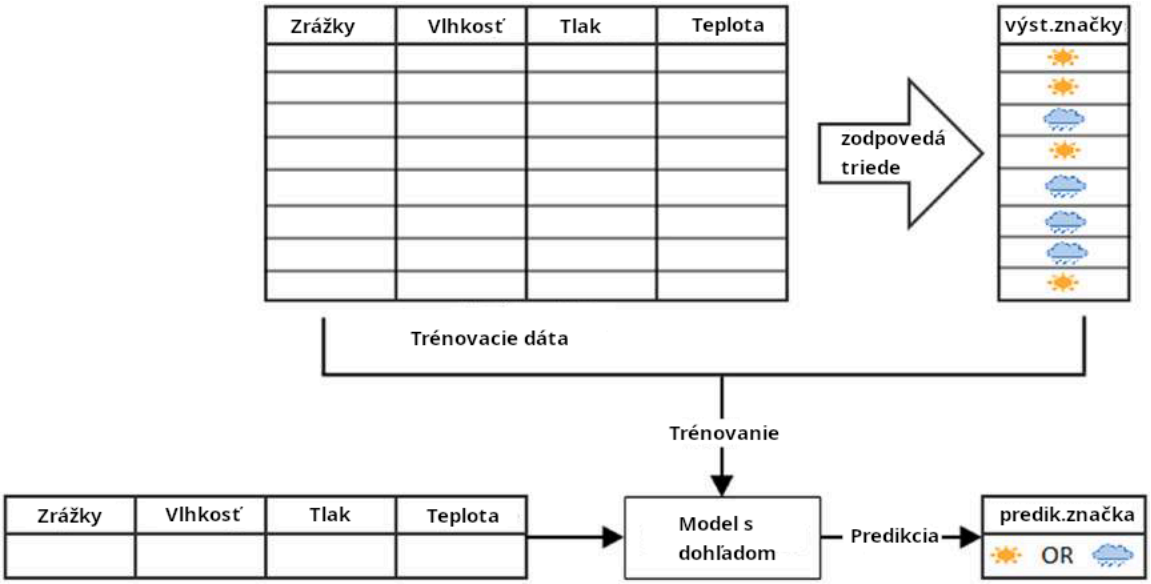
Tieto metódy učenia sa nazývajú učenie s dohľadom, pretože model sa učí na vzorkách údajov, kde sú požadované výstupné značky/labels známe už vopred vo fáze trénovania. Učenie s dohľadom sa v podstate snaží modelovať vzťah medzi vstupmi a ich zodpovedajúcimi výstupmi z trénovacích údajov tak, aby sme boli schopní predpovedať výstupné odpovede pre nové dátové vstupy na základe znalostí, ktoré získal predtým s ohľadom na vzťahy a mapovania medzi vstupmi a ich cieľovými výstupmi. To je presne dôvod, prečo sa metódy učenia s dohľadom vo veľkej miere používajú v prediktívnej analýze, kde je hlavným cieľom predpovedať určitú odozvu na niektoré vstupné údaje, ktoré sa zvyčajne vkladajú do trénovaného modelu ML s dohľadom. Metódy učenia s dohľadom sa delia na dve hlavné triedy podľa typu ML úloh, ktoré sa snažia riešiť:

- Klasifikácia
- Regresia

4.1.1.1 Klasifikácia

Úlohy založené na klasifikácii sú podoblasťou v rámci strojového učenia s dohľadom, kde je kľúčovým cieľom predpovedať výstupné značky alebo labels, ktoré sú kategorickej povahy pre vstupné údaje na základe toho, čo sa model naučil vo fáze trénovania. Výstupné značky sú tu známe aj ako triedy alebo označenia tried, pretože majú kategorickú povahu, čo znamená, že ide o neusporiadané a diskrétne hodnoty. Každá výstupná odozva teda patrí do konkrétnej diskrétnej triedy alebo kategórie. Predpokladajme, že si vezmeme predpovedania počasia ako príklad z reálneho sveta. Pre jednoduchosť si povedzme, že sa snažíme predpovedať, či je slnečné alebo daždivé počasie na základe viacerých vzoriek vstupných

údajov pozostávajúcich z atribútov alebo príznakov, ako je vlhkosť, teplota, tlak a zrážky. Keďže predpoveď môže byť slnečná alebo daždivá, celkovo existujú dve odlišné triedy; preto možno tento problém označiť aj ako problém binárnej klasifikácie. Obrázok 3 znázorňuje úlohu binárnej klasifikácie počasia predpovedajúcu počasie ako slnečné alebo daždivé na základe trénovania modelu s dohľadom na vzorkách vstupných údajov s vektormi príznakov (zrážky, vlhkosť, tlak a teplota) pre každú vzorku údajov/pozorovania a ich zodpovedajúce označenia tried ako slnečné alebo daždivé.

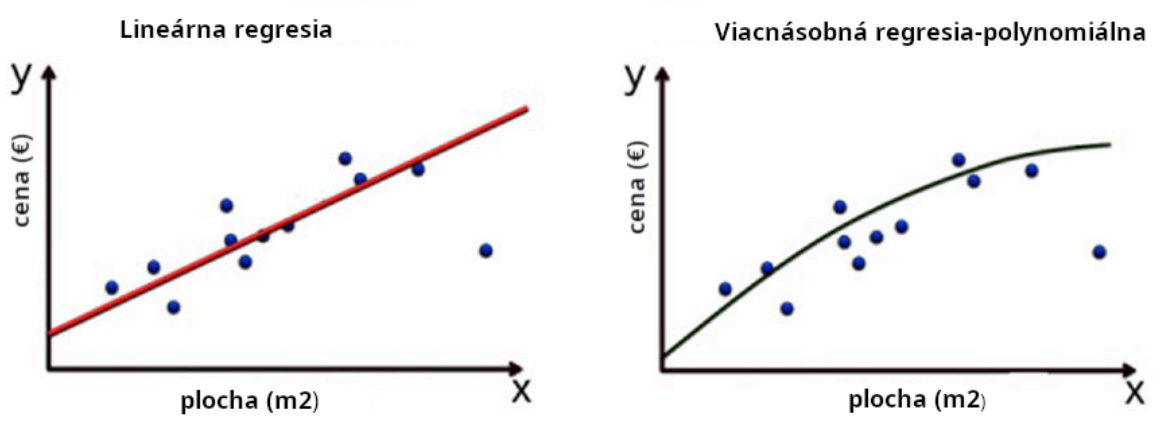


Obr.3: Binárna klasifikácia pre predpoveď počasia

Úloha, kde je celkový počet odlišných tried viac ako dve, sa stáva klasifikačným problémom viacerých tried, kde každá predikčná odpoveď môže byť ktorákoľvek z pravdepodobných tried z tejto množiny. Jednoduchým príkladom by bol pokus o predpovedanie číselných číslic z naskenovaných ručne písaných obrázkov. V tomto prípade sa stáva problémom klasifikácie 10 tried, pretože označenie výstupnej triedy pre akýkoľvek obrázok môže byť ľubovoľná číslica od 0 do 9. V oboch prípadoch je výstupná trieda skalárna hodnota poukazujúca na jednu konkrétnu triedu. Úlohy klasifikácie s viacerými značkami alebo labels sú také, že na základe akejkoľvek vzorky vstupných údajov je výstupná odpoveď zvyčajne vektor s jednou alebo viacerými značkami výstupnej triedy. Jednoduchým problémom v reálnom svete by bola snaha predpovedať kategóriu spravodajského článku, ktorý by mohol mať viacero výstupných tried, ako sú správy, financie, politika atď.

4.1.1.2 Regresia

Úlohy strojového učenia, ktorých hlavným cieľom je odhad hodnoty, možno nazvať regresnými úlohami. Metódy založené na regresii sa trénujú na vzorkách vstupných údajov, ktoré majú výstupné odpovede, ktoré sú spojitými číselnými hodnotami na rozdiel od klasifikácie, kde máme diskrétne kategórie alebo triedy. Regresné modely využívajú atribúty alebo príznaky vstupných údajov (nazývané aj vysvetľujúce alebo nezávislé premenné) a ich zodpovedajúce spojité číselné výstupné hodnoty (nazývané aj ako odpoveď, závislá alebo výsledná resp. výstupná premenná) na učenie sa špecifických vzťahov a asociácií medzi vstupmi a ich zodpovedajúcimi výstupmi. S týmito znalosťami dokáže model predpovedať výstupné odpovede pre nové, neznáme dátové inštancie podobné klasifikácii, ale so súvislými číselnými výstupmi. Jedným z najbežnejších príkladov regresie z reálneho svete je predikcia cien domov. Možno vytvoriť jednoduchý regresný model na predpovedanie cien domov na základe údajov týkajúcich sa plôch pozemkov v metroch štvorcových.



Obr. 4: Regresný model pre predikciu ceny domu

Obrázok 4 ukazuje dva možné regresné modely založené na rôznych metódach predikcie cien domov na základe plochy pozemku. Základnou myšlienkou je, že sa snažíme určiť, či existuje nejaký vzťah alebo asociácia medzi oblasťou grafu dátového prvku a výslednou premennou, ktorou je cena domu a tú chceme predpovedať. Akonáhle sa teda dozvieme tento trend alebo vzťah znázornený na obrázku 3, môžeme predpovedať ceny domov v budúcnosti pre akýkoľvek daný pozemok. Ak ste si tento obrázok všimli pozorne, zámerne sme zobrazili dva typy modelov, aby sme ukázali, že existuje viacero spôsobov, ako vytvoriť model na základe trénovacích údajov. Hlavným cieľom je minimalizovať chyby počas trénovania a validácie modelu tak, aby sa dobre zovšeobecnil, nepreháňal alebo nebol zaujatý iba trénovacími údajmi a dobre fungoval v budúcich predpovediach.

Jednoduché *lineárne regresné modely* sa pokúšajú modelovať vzťahy na údajoch s jedným príznakom alebo vysvetľujúcou premennou x a jednou premennou odozvy-výstupu y, kde cieľom je predpovedať y. Metódy ako sú metóda najmenších štvorcov (ordinary least squares OLS) sa zvyčajne používajú na dosiahnutie najlepšieho lineárneho prispôsobenia počas trénovania modelu. Viacnásobná regresia je tiež známa ako regresia s viacerými premennými. Tieto metódy sa snažia modelovať údaje, kde máme jednu výstupnú premennú odozvy y v každom pozorovaní, ale viacero vysvetľujúcich premenných vo forme vektora X namiesto jednej vysvetľujúcej premennej. Cieľom je predpovedať y na základe rôznych príznakov prítomných vo vektore X. Príkladom z reálneho sveta by mohlo byť rozšírenie nášho modelu predpovede domu o vytvorenie sofistikovanejšieho modelu, v ktorom predpovedáme cenu domu na základe viacerých príznakov namiesto iba plochy pozemku v každej vzorke údajov. Príznaky môžu byť reprezentované vo vektore, kde by sa nachádzala plocha pozemku, počet spální, počet kúpeľní, celkový počet poschodí, zariadený alebo nezariadený dom. Na základe všetkých týchto atribútov sa model snaží naučiť vzťah medzi jednotlivými vektormi príznakov a jeho zodpovedajúcou cenou domu, aby ich mohol predpovedať v budúcnosti.

Polynomiálna regresia je špeciálny prípad viacnásobnej regresie, kde je výstupná premenná y modelovaná ako polynóm n-tého stupňa vstupného príznaku x. V podstate ide o viacnásobnú regresiu, kde každý prvok vo vstupnom vektore funkcie je násobkom x. Model vpravo na obrázku 3 na predpovedanie cien domov je polynomiálny model stupňa 2. Nelineárne regresné metódy sa snažia modelovať vzťahy medzi vstupnými príznakmi a výstupmi na základe kombinácie nelineárnych príznakov aplikovaných na vstupné atribúty a potrebných parametrov modelu.

Lasso regresia je špeciálna forma regresie, ktorá vykonáva normálnu regresiu a dobre zovšeobecňuje model vykonaním regularizácie, ako aj výberu príznakov alebo premenných.

Laso je skratka pre operátor najmenšieho absolútneho zmršťovania a výberu. Norma L1 sa zvyčajne používa ako regularizačný termín v laso regresii.

Hrebeňová regresia je ďalšia špeciálna forma regresie, ktorá vykonáva normálnu regresiu a zovšeobecňuje model vykonaním regularizácie, aby sa zabránilo nadmernému prispôsobeniu modelu. Norma L2 sa zvyčajne používa ako regularizačný člen v regresii hrebeňa.

Zovšeobecnené lineárne modely predstavujú všeobecné rámce, ktoré možno použiť na modelovanie údajov predpovedajúcich rôzne typy výstupných odpovedí vrátane kontinuálnych, diskrétnych a ordinálnych údajov. Algoritmy ako logistická regresia sa používajú pre kategorické údaje a usporiadaná probitová regresia pre ordinálne údaje.

Medzi najdôležitejšie algoritmy učenia s dohľadom patria:

- Metóda K-najbližších susedov - K-Nearest Neighbors (KNN)
- Lineárna regresia – Linear Regression
- Logistická regresia – Logistic Regression
- Metóda podporných vektorov - Support Vector Machines (SVMs)
- Rozhodovacie stromy – Decision Tree (DT) a náhodné lesy – Random Forest (RF)
- Neurónové siete – Neural Networks (NN). Niektoré architektúry neurónových sietí, ako sú autoenkóder a obmedzené Boltzmannove stroje – restricted Boltzmann machines (RBM) predstavujú učenie bez dohľadu. Niektoré architektúry môžu predstavovať učenie s čiastočným dohľadom, ako napr. v prípade použitia Deep Belief Network s predtrénovaním na základe učenia bez dohľadu.

4.1.1.3 K-Nearest Neighbor

KNN možno definovať ako nelineárnu neparametrickú klasifikačnú metódu. Tento algoritmus je založený na veľmi jednoduchom princípe, že podobné dáta sa nachádzajú blízko pri sebe v prehľadávanom, resp. dátovom príznakovom priestore. Pre každý prvok testovaných dát algoritmus kNN vyhľadá na základe vzdialenosti určité okolie - susedstvo (neighborhood), obsahujúce k prvkov trénovacích, resp. referenčných dát a na základe určitého kritéria – najčastejšie pravidlo väčšiny - priradí danému prvku označenie triedy. Metóda kNN patrí do skupiny tzv. lenivých učiacich algoritmov (lazy learning techniques). Tieto algoritmy sú špecifické tým, že na rozdiel od už spomínaných modelovo založených metód sa z trénovacích dát nevytvára zovšeobecnený model, ktorý popisuje trénovacie dáta a ďalej sa využíva pri samotnej klasifikácii. Proces klasifikácie teda nie je tvorený 2 fázami – trénovanie modelov, samotná klasifikácia testovacích dát – ale prebieha v jednom kroku. Algoritmus kNN má 2 kľúčové elementy:

- dištančná metrika
- hodnota k susedov

Je zrejmé, že v prípade algoritmu kNN vyjadruje vzdialenosť mieru podobnosti dát – čím bližšie v príznakovom priestore sú si 2 prvky, tým sú podobnejšie. Z tohto hľadiska voľba vhodnej metriky vzdialenosti zohráva dôležitú úlohu a závisí vo veľkej miere na danom klasifikačnom probléme. Azda najznámejšou mierou je Euklidova vzdialenosť, definovaná nasledujúcim vzťahom:

$$d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k)^2},$$

resp. štandardizovaná Euklidova vzdialenosť

$$d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k).V^{-1}.(x_k-y_k)}$$

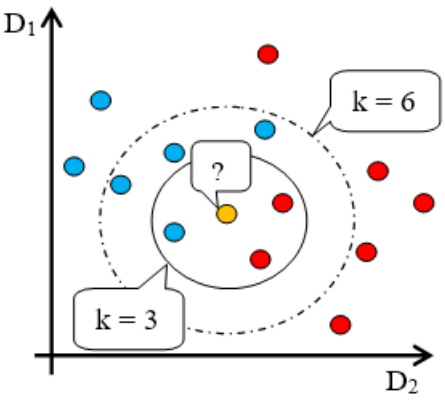
kde *x* je vektor testovacích dát, *y* je prvok testovacích dát, ***V*** je diagonálna matica, ktorej j-ty diagonálny prvok je $S(j)^2$, pričom *S* je vektor štandardných odchýlok. Z ďalších metrík možno spomenúť Mahalanobisovu vzdialenosť

$$d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k).C^{-1}.(x_k-y_k)'}\ ,$$

kde $\boldsymbol{C}^{-1}$ je inverzná kovariančná matica či Manhattan metriku

$$d(x,y)=\sqrt{\sum_{k=1}^{n}|x_k-y_k|}\ .$$

Hodnota *k* určuje „susedstvo“ alebo okolie, to znamená množstvo prvkov trénovacích dát, ktoré sa berú do úvahy pri určovaní príslušnosti určitého prvku testovacích dát k danej triede. Ak je hodnota k príliš malá, výsledok klasifikácie môže byť citlivý na tzv. šumové prvky (noisy points). Na druhej strane, ak je hodnota k príliš veľká, „susedstvo“ môže obsahovať príliš veľa prvkov z iných tried. Príklad ilustruje nasledujúci obrázok.



Obr. 5: Demonštrácia vplyvu hodnoty k na presnosť algoritmu kNN

Úlohou je určiť, do ktorej triedy patrí oranžový krúžok. V prvom prípade pre *k* = 3 bude patriť daný prvok do triedy červených krúžkov, v druhom prípade pre *k* = 6 bude patriť do triedy modrých krúžkov. K sa zvyčajne volí nepárne číslo.

Finálne určenie triedy, do ktorej by mal daný prvok testovacích dát patriť, z množiny možných tried určenej „susedstvom“, zvyčajne podlieha pravidlu väčšiny (majority voting) popísaným nasledovným vzťahom:

$$l=\arg\max_{v}\sum_{(x_i,y_i)\in D_Z}I(v=y_i)\ ,$$

kde v je označenie triedy, y_i je označenie triedy i-tého najbližšieho suseda a *I*(.) je funkcia, ktorá vracia hodnotu 1, pokiaľ je argument pravdivý. Iným príkladom je výber triedy na základe váhovania vzdialeností

$$l = \arg\max_{v} \sum_{(x_i, y_i) \in D_Z} w_i * I(v = y_i) \ ,$$

kde váhovací faktor w_i je prevrátená hodnota vzdialenosti testovacieho a trénovacieho , resp. referenčného prvku

$$wi = \frac{1}{d(x_i, y_i)^2} \ .$$

Pokiaľ sa použije toto rozhodovacie pravidlo, algoritmus kNN nie je až tak citlivý na veľkosť hodnoty *k*.

Metóda kNN je napriek svojej jednoduchosti veľmi úspešná v oblasti vyhľadávania dát a patrí k desiatim najlepším algoritmom v danej oblasti. V niektorých prípadoch dokáže dokonca predčiť aj omnoho sofistikovanejšie a komplexnejšie klasifikačné metódy. Vďaka svojej jednoduchosti je táto metóda veľmi ľahko implementovateľná, je vhodná pre multimodálne triedy a je veľmi flexibilná. Oproti vyššie spomenutým klasifikačným metódam odpadá časovo veľmi náročná (obzvlášť pri SVM) fáza vytvárania modelu, hoci samotná klasifikácia zvyčajne vyžaduje väčší časový interval v porovnaní s danými metódami. Nevýhodou je teda rastúca výpočtová náročnosť pri veľkých databázach. Niekoľko rôznych neskorších úprav a variácií pôvodného algoritmu bolo publikovaných v priebehu času (fuzzy kNN).

4.1.1.4 Support Vector Machine

Metóda podporných vektorov je klasifikačnou a regresnou metódou, ktorú navrhol v 90-tych rokoch V. Vapnik a ktorá v podstate určuje hraničnú krivku oddeľujúcu 2 triedy. Hlavnou myšlienkou je nájdenie nadroviny, ktorá by optimálne rozdelila v príznakovom priestore vstupné dáta. Pre danú množinu trénovacích dát patriacich do 2 tried (x_1, y_1), ...,(x_l, y_l), kde $x_i \in \boldsymbol{R}_n$ a $yi \in \{-1, 1\}$, sa teda snažíme nájsť nadrovinu $wx + b = 0$, aby sme mohli separovať dáta. Zo všetkých nadrovín určených *w* a *b* existuje len jedna taká, ktorá maximalizuje vzdialenosť medzi najbližšími bodmi z týchto tried a nadrovinou - optimálna nadrovina a platí:

$$d(w,b) = \min_{\{x_i | y_i = 1\}} \frac{wx_i + b}{|w|} - \max_{\{x_i | y_i = -1\}} \frac{wx_i + b}{|w|}$$

a zodpovedajúca minimálna a maximálna hodnota je -1, resp. +1

$$d(w,b) = \left| \frac{1}{\mathbf{w}} - \frac{-1}{\mathbf{w}} \right| = \frac{2}{|\mathbf{w}|} \ .$$

Minimalizáciou výrazu $|w|^2 / 2$ v závislosti na podmienke $y_i (wx_i + b) \geq 1$ dostaneme pre daný pár nadrovín maximálnu vzdialenosť 2 tried. Riešenie optimalizačného problému SVM je dané sedlovým bodom (saddle point) Lagrangeovej funkcie:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(wx_i + b \right) - 1 \right] ,$$

kde α_i je Lagrangeov multiplikátor. Zatiaľ čo $L(w, b, \alpha)$ sa zmenšuje s *w* a *b*, derivácie $L(w, b, \alpha)$ sú pre všetky $\alpha_i \geq 0$ rovné nule. Riešenie je dané

$$\overline{w} = \sum_{i=1}^{l} \overline{\alpha}_i y_i x_i \qquad \overline{b} = -\frac{1}{2} \overline{w} \left[x_r + x_s \right] ,$$

kde x_r a x_s sú podporné vektory, ktoré patria do triedy $y_r = 1$, resp. $y_s = -1$. Dáta v pôvodnom príznakovom priestore nie sú separovateľné žiadnou lineárnou nadrovinou, ale môžu byť lineárne separovateľné v nelineárne mapovanom priestore príznakov, ktorý je odlišný od toho pôvodného a zvyčajne má vyššiu dimenziu. Toto nelineárne mapovanie *K*(.,.) sa uskutočňuje pomocou jadrových alebo kernel funkcií (kernel functions):

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \overline{\alpha}_i y_i K(x_i, x) + b\right) \sum_{i=1}^{N} \alpha_i t_i K(x, x_i) + b,$$

kde y_i sú cieľové hodnoty, $\sum_{i=1}^{l} \overline{\alpha}_i y_i = 0$ a $\overline{\alpha}_i > 0$. Jadro *K*(.,.) sa vytvára tak, aby spĺňalo špecifické vlastnosti v závislosti na danom probléme a možno napísať:

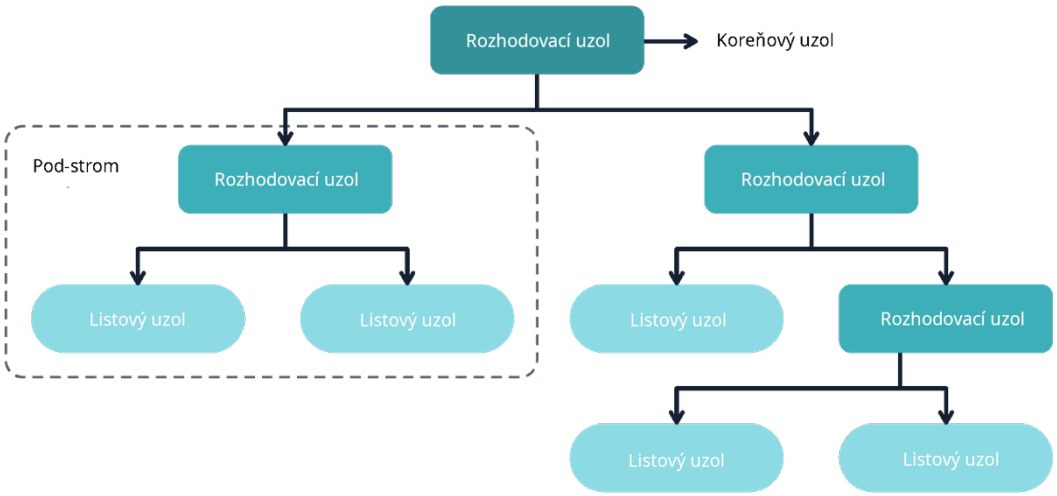
$$K(x, y) = \Phi(x)\Phi(y) \ ,$$

kde Φ(x) je mapovanie vstupného príznakového priestoru do nového priestoru s potenciálne nekonečnou dimenziou. Existujú 3 jadrové funkcie zabezpečujúce nelineárne mapovanie:

- *polynomiálna K*(*x*, *y*) = (*xy* + 1) z – kde z predstavuje stupeň polynómu
- *RBF Gaussove radiálne bázové funkcie K*(*x*, *y*) = exp {-[(x – y)^2 / 2σ^2]} – kde σ je štandardná odchýlka gaussovskej funkcie
- *MLP funkcia K*(*x*, *y*) = tanh (scale(*xy*) - offset) – scale a offset predstavujú známe a dané parametre.

4.1.1.5 Decision Tree

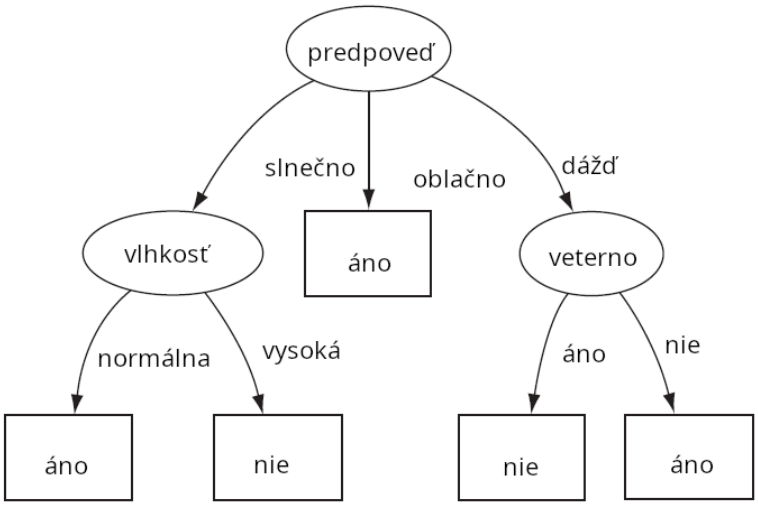
Rozhodovací strom je neparametrický algoritmus, ktorý sa používa na klasifikačné aj regresné úlohy. Má hierarchickú stromovú štruktúru, ktorá pozostáva z koreňového uzla, vetiev, vnútorných uzlov a listových uzlov. Ako možno vidieť z nižšie uvedeného diagramu, rozhodovací strom začína koreňovým uzlom, ktorý nemá žiadne prichádzajúce vetvy.



Obr. 6: Rozhodovací strom

Odchádzajúce vetvy z koreňového uzla sa potom privádzajú do interných uzlov, známych aj ako rozhodovacie uzly. Na základe dostupných funkcií oba typy uzlov vykonávajú

hodnotenia s cieľom vytvoriť homogénne podmnožiny, ktoré sú označené listovými uzlami alebo koncovými uzlami. Listové uzly predstavujú všetky možné výsledky v rámci súboru údajov. Ako príklad si možno predstaviť, že ste sa snažíme posúdiť, či by sme mali alebo nemali ísť hrať golf, môžeme na výber použiť nasledujúce pravidlá rozhodovania:



Obr. 7: Rozhodovací strom pre dataset Golf

Tento typ štruktúry vývojového diagramu tiež vytvára ľahko stráviteľnú reprezentáciu rozhodovania, čo umožňuje rôznym skupinám v celej organizácii lepšie pochopiť, prečo bolo rozhodnutie prijaté. Učenie rozhodovacieho stromu využíva stratégiu rozdeľuj a panuj tým, že vykonáva chamtivé vyhľadávanie s cieľom identifikovať optimálne body rozdelenia v strome. Tento proces rozdelenia sa potom opakuje zhora nadol, rekurzívnym spôsobom, kým všetky alebo väčšina záznamov nie sú klasifikované pod špecifickými štítkami tried. To, či sú všetky dátové body klasifikované ako homogénne množiny, do značnej miery závisí od zložitosti rozhodovacieho stromu. Menšie stromy sú ľahšie schopné dosiahnuť čisté listové uzly – t. j. dátové body v jednej triede. Ako však strom rastie, je čoraz ťažšie udržať túto čistotu a zvyčajne to vedie k tomu, že do daného pod-stromu spadá príliš málo údajov. Keď k tomu dôjde, je to známe ako fragmentácia údajov a často to môže viesť k nadmernému prispôsobeniu. Výsledkom je, že rozhodovacie stromy uprednostňujú malé stromy, čo je v súlade so zásadou úspornosti v Occamovej britve; to znamená, že "entity by sa nemali množiť nad nevyhnutnosť". Inak povedané, rozhodovacie stromy by mali zvyšovať zložitosť iba v prípade potreby, pretože najjednoduchšie vysvetlenie je často najlepšie. Aby sa znížila zložitosť a zabránilo sa nadmernému prispôsobeniu, zvyčajne sa používa prerezávanie; Toto je proces, ktorý odstraňuje vetvy, ktoré sa rozdeľujú na prvkoch s nízkou dôležitosťou. Prispôsobenie modelu sa potom môže vyhodnotiť prostredníctvom procesu krížovej validácie.

Ďalším spôsobom, ako si môžu rozhodovacie stromy zachovať svoju presnosť, je vytvorenie súboru pomocou algoritmu náhodného lesa; Tento klasifikátor predpovedá presnejšie výsledky, najmä ak jednotlivé stromy navzájom nekorelujú. Spomedzi viacerých variant možno uviesť nasledujúce najčastejšie používané:

- ID3
- C4.5
- CART
-

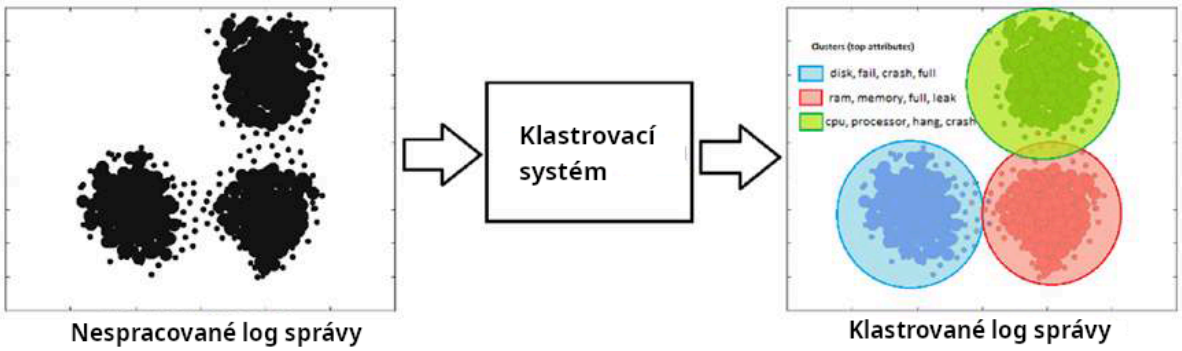
4.1.2 Unsupervised learning (učenie bez dohľadu)

Metódy učenia s dohľadom zvyčajne vyžadujú určité tréningové údaje, kde výsledky, ktoré sa snažíme predpovedať, sú už dostupné vo forme diskrétnych značiek/labels alebo spojitých hodnôt. V praxi však často však nemáme k dispozícii vopred označené tréningové údaje a stále chceme z našich údajov extrahovať užitočné znalosti alebo vzory. V tomto scenári sú metódy učenia bez dohľadu mimoriadne účinné. Tieto metódy sa nazývajú bez dohľadu, pretože model alebo algoritmus sa snaží naučiť inherentné latentné štruktúry, vzory a vzťahy z daných údajov bez akejkoľvek pomoci alebo dohľadu, ako je poskytovanie anotácií vo forme označených výstupov alebo výsledkov. Učenie bez dohľadu sa viac zaoberá snahou extrahovať zmysluplné poznatky alebo informácie z údajov, než sa snažiť predpovedať nejaký výsledok na základe predtým dostupných tréningových údajov s dohľadom. Vo výsledkoch učenia bez dohľadu je väčšia neistota, ale z týchto modelov možno získať aj veľa informácií, ktoré predtým pri pohľade na nespracované údaje neboli k dispozícii. Učenie bez dohľadu môže byť často jednou z úloh spojených s budovaním obrovského spravodajského systému. Mohli by sme napríklad použiť učenie bez dohľadu na získanie možných označení výsledkov pre nálady tweetov pomocou znalosti anglickej slovnej zásoby a potom trénovať model s dohľadom na podobných dátových bodoch a ich výsledkoch, ktoré sme predtým získali prostredníctvom učenia bez dohľadu. Neexistuje žiadne pevné pravidlo, pokiaľ ide o používanie len jednej konkrétnej techniky. Vždy možno kombinovať viacero metód, pokiaľ sú relevantné pri riešení problému. Metódy učenia bez dohľadu možno kategorizovať do nasledujúcich širokých oblastí úloh ML relevantných pre učenie bez dohľadu.

- klastrovanie – clustering
- redukcia dimenzionality – dimensionality reduction
- detekcia anomálií – anomaly detection
- asociatívne dolovanie na základe pravidiel – association rule-mining

4.1.2.1 Klastrovanie (zhlukovanie)

Metódy zhlukovania sú metódy strojového učenia, ktoré sa snažia nájsť vzory podobnosti a vzťahov medzi vzorkami údajov v našom súbore údajov a potom tieto vzorky zoskupiť do rôznych skupín, takže každá skupina alebo zhluk vzoriek údajov má určitú podobnosť na základe inherentných atribútov alebo príznakov. Tieto metódy sú úplne bez dohľadu, pretože sa pokúšajú zhlukovať údaje tak, že sa pozerajú na funkcie údajov bez predchádzajúceho trénovania, dohľadu alebo znalostí o atribútoch údajov, asociáciách a vzťahoch. Ako príklad z reálneho sveta možno uviesť problém spustenia viacerých serverov v dátovom centre a pokusu o analýzu protokolov pre typické problémy alebo chyby. Našou hlavnou úlohou je určiť rôzne druhy správ denníka, ktoré sa zvyčajne vyskytujú často každý týždeň. Jednoducho povedané, chceme zoskupiť správy denníka do rôznych klastrov alebo zhlukov na základe niektorých inherentných charakteristík. Jednoduchým prístupom by bolo extrahovať príznaky zo správ denníka, ktoré by boli v textovom formáte a aplikovať zhlukovanie na to isté a zoskupiť podobné správy denníka na základe podobnosti obsahu. Obrázok 4 ukazuje, ako by zhlukovanie vyriešilo tento problém. V podstate máme na začiatok nespracované správy denníka. Náš klastrovací systém by využíval extrakciu príznakov na extrahovanie príznakov z textu, ako sú výskyty slov, výskyty fráz atď. Nakoniec by sa na zoskupenie alebo zhlukovanie správ použil zhlukovací algoritmus, ako je K-means alebo hierarchické zhlukovanie, na zoskupenie alebo zhlukovanie správ na základe podobnosti ich inherentných vlastností.



Obr. 8: Klastrovanie

Z obrázka 7 je celkom zrejmé, že náš systém má tri odlišné zhluky správ denníka, kde prvý klaster zobrazuje problémy s diskom, druhý klaster je o problémoch s pamäťou a tretí klaster je o problémoch s procesorom. Na obrázku sú znázornené aj hlavné slová, ktoré pomohli pri rozlíšení zhlukov a zoskupení podobných vzoriek údajov (protokolov). Samozrejme, niekedy môžu byť niektoré príznaky prítomné vo viacerých vzorkách údajov, a preto môže dôjsť aj k miernemu prekrývaniu zhlukov, pretože ide o učenie bez dohľadu. Hlavným cieľom je však vždy vytvoriť zhluky tak, aby prvky každého zhluku boli blízko seba a ďaleko od prvkov iných klastrov. Existujú rôzne typy metód zhlukovania, ktoré možno klasifikovať podľa nasledujúcich hlavných prístupov.

- Metódy založené na ťažiskách – K-means, K-medoids
- Hierarchické metódy zhlukovania – Hierarchical Cluster Analysis (HCA)
- Metódy klastrovania založené na distribúcii hodnôt – Gaussian Mixture Model (GMM)
- Metódy založené na hustote – DBSCAN, OPTICS

4.1.2.2 K-means

K-means alebo metóda K stredných hodnôt je jednou z najpopulárnejších metód zhlukovania. Pseudo-kód uvedený nižšie zobrazuje postup zhlukovania K-priemerov.

vstup: K - počet klastrov, D - dátová sada N bodov

výstup: množina K-klastrov/zhlukov

- inicializácia
- **repeat**

  for každý bod p v D **do**

  nájdi najbližší centroid a priraď p k zodpovedajúcemu klastru

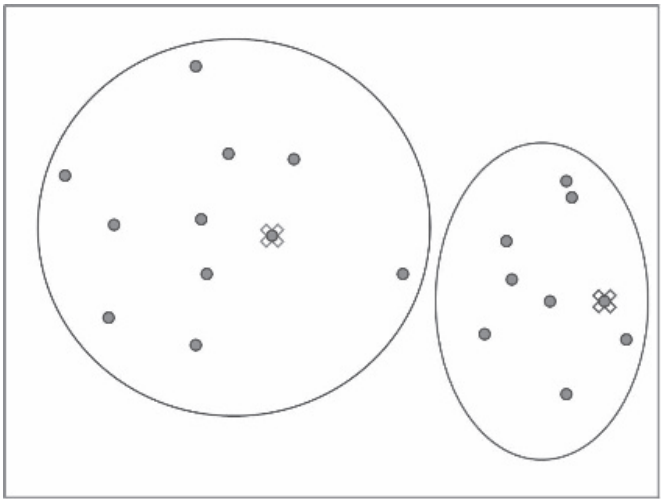
  end for

  aktualizuj klastre výpočtom nových centier pomocou priemeru členov

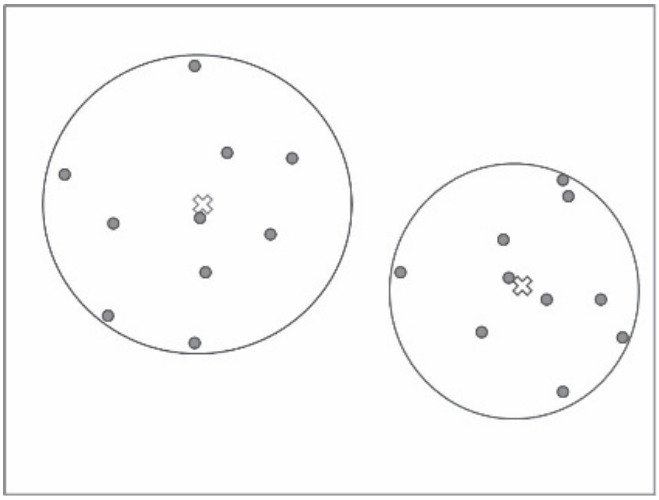
- **until** nie sú splnené stop-kritériá
- **return** výsledok klastrovania.

Základná myšlienka znie: Vzhľadom na počiatočné, ale nie optimálne zhlukovanie, premiestnite každý bod do jeho nového najbližšieho stredu, aktualizujte stredy zhlukovania výpočtom priemeru členských bodov a opakujte proces premiestňovania a aktualizácie, kým nie sú splnené kritériá konvergencie (ako je preddefinovaný počet iterácií, rozdiel v hodnote kriteriálnej funkcie). Úlohou inicializácie je vytvoriť počiatočné zhluky K. Bolo navrhnutých mnoho inicializačných techník, od jednoduchých metód, ako je výber prvých K dátových bodov,

Forgy inicializácia (náhodný výber K dátových bodov v datasete) a *Random partitions* (náhodné rozdelenie dátových bodov na K podmnožín), až po sofistikovanejšie metódy, ako je inicializácia založená na hustote, *Intelligent initialization*, *Furthest First initialization* inicializácia (FF skrátene funguje tak, že náhodne vyberie prvý stredový bod), potom pridanie ďalších stredových bodov, ktoré sú najďalej od existujúcich) a inicializácia *subset furthest-first* (SFF). Obrázky nižšie ukazujú príklad zhlukovania K-priemerov na množine bodov, pričom K = 2. Zhluky sa inicializujú náhodným výberom dvoch bodov ako stredov.



Obr.9-1: Inicializácia k-mean



Obr. 9-2: Aktualizácia k-mean

4.1.2.3 Gaussian Mixture Model

Metóda GMM je založená na štatistických vlastnostiach signálov a predpokladá, že pozorovaný proces a jeho rozloženie možno modelovať pomocou súčtu určitej skupiny komponentov a ich hustôt rozdelenia pravdepodobnosti:

$$P(x|\lambda) = \sum_{m=1}^{M} c_m b_m(x),$$

kde x je náhodný vektor rozmeru d, M je počet komponentov modelu, c_m je vektor váh a $b_m(x)$ je Gaussovo rozdelenie definované vektorom stredných hodnôt μ_m a kovariančnou maticou Σ_m, pričom platí:

$$b_m(x) = \frac{1}{(2\pi)^{d/2}\left|\Sigma_m\right|^{1/2}} \exp\left\{-\frac{1}{2}\left[(x-\mu_m)\Sigma_m^{-1}(x-\mu_m)^T\right]\right\}.$$

GMM je parametrický model a jeho parametre sa zvyknú označovať $\lambda = \{c_m, \mu_m, \Sigma_m\}$ pre $m = 1,2,\ldots M$. Tieto parametre sa odhadujú tak, aby pravdepodobnosť $P(x \mid \lambda)$ vstupných dát pre všetky triedy bola maximálna. Jedným z užitočných algoritmov je iteračný algoritmus *Expectation Maximization* (EM), ktorý upravuje parametre modelu na základe nasledujúcich rovníc:

$$\mu_m = \frac{\sum_{i=1}^{n} x_i P(m|x_i,\lambda)}{\sum_{i=1}^{n} P(m|x_i,\lambda)}$$

$$\Sigma_m = \frac{\sum_{i=1}^{n} P(m|x_i,\lambda)(x_i-\mu_m)(x_i-\mu_m)^T}{\sum_{i=1}^{n} P(m|x_i,\lambda)}$$

$$c_m = \frac{1}{n}\sum_{i=1}^{n} P(m|x_i,\lambda)$$

$$P(m|xi,\lambda) = \frac{c_m b_m(x_i)}{\sum_{j=1}^{M} c_j b_j(x_i)}$$

Pre množinu K tried zvukov reprezentovanými modelmi GMM λ_1 λ_2, ... , λ_K je cieľom rozpoznávania (klasifikácie) nájsť taký model, ktorý má maximálnu aposteriórnu pravdepodobnosť $P(\lambda \mid X)$ (MAP kritérium) pre danú postupnosť pozorovaní $X = x_1, x_2, \ldots, x_L$:

$$\hat{K} = \arg\max_{1\le k\le K} P(\lambda_k|X) = \arg\max_{1\le k\le K} \frac{P(X|\lambda_k)P(\lambda_k)}{P(X)},$$

pričom vzhľadom k tomu, že pravdepodobnosť $P(X)$ je rovnaká pre všetky modely a nemení výsledok maximalizácie, možno výraz v menovateli zanedbať, takže potom platí:

$$\hat{K} = \arg\max_{1\le k\le K} P(X|\lambda_k)P(\lambda_k).$$

Aplikovaním operácie logaritmu a uvažovaním štatistickej nezávislosti medzi jednotlivými pozorovaniami potom možno napísať:

$$\hat{K} = \arg\max_{1\le k\le K} \sum_{l=1}^{L} \log P(x_l|\lambda_k)P(\lambda_k),$$

a ak predpokladáme, že pravdepodobnosť $P(\lambda_k)$ je rovnaká pre všetky triedy dátovej sady, môžeme túto pravdepodobnosť z predchádzajúceho vzťahu vynechať.

4.1.2.4 DBSCAN

DBSCAN je algoritmus klastrovania založený na hustote, ktorý nevyžaduje vopred určený počet klastrov. DBSCAN závisí od dvoch parametrov: minimálny počet bodov potrebných na vytvorenie zhluku zo skupiny bodov a ξ (ktorý určuje minimálnu vzdialenosť medzi dvoma bodmi, aby sa mohli nazývať susedné body).

Interne algoritmus klasifikuje každý dátový bod ako jeden z nasledujúcich troch kategórií:

- stredové body – sú tie, ktoré majú aspoň minimálny počet bodov v okolí definovaný okruhom polomeru ξ
- hraničné body – sú také body, ktoré nie sú stredovými bodmi, ale nachádzajú sa v susednej oblasti alebo klastra stredového bodu opísaného vyššie
- body anomálie – sú také body, ktoré nie sú ani stredovým bodom, ani z neho nie sú dosiahnuteľné (teda ani hraničný bod)

Stručne zhrnuté, proces klastrovania funguje nasledovne:

1. nastavenie hodnoty parametrov, minimálny počet bodov a ξ.
2. náhodne vybratie počiatočného bodu (povedzme A) a nájdenie všetkých bodov vo vzdialenosti ξ alebo menšej od tohto bodu.
3. ak počet bodov spĺňa prahovú hodnotu pre minimálny počet bodov, potom je A stredovým bodom a okolo neho sa môže vytvoriť zhluk alebo klaster. Všetky body vo vzdialenosti ξ alebo menšej sú hraničné body a pridajú sa do zhluku so stredom v bode A.
4. opakovanie týchto krokov pre každý bod. Ak sa ukáže, že bod (povedzme B) pridaný do zhluku je tiež stredovým bodom, najprv vytvoríme jeho vlastný zhluk a potom ho zlúčime s pôvodným zhlukom okolo A.

DBSCAN je teda schopný identifikovať oblasti s vysokou hustotou v príznakovom priestore a zoskupiť body do klastrov. Bod, ktorý nespadá do zhluku, je určený ako anomálny alebo odľahlý.

DBSCAN ponúka dve silné výhody oproti K-Means zhlukovaniu:

- nie je nutné špecifikovať počet zhlukov. Pri analýze dát, najmä v podmienkach učenia bez učiteľa, si často nemusíme byť vedomí počtu tried v dátach.

Preto nie je známe, aký by mal byť počet klastrov pre detekciu anomálií. DBSCAN tento problém eliminuje a vytvára zhluky vhodne na základe hustoty.

DBSCAN je schopný nájsť zhluky, ktoré majú ľubovoľné tvary. Pretože je založený na hustote oblastí okolo základných bodov, tvar zhlukov nemusí byť kruhový. Na druhej strane K-Means zhlukovanie nedokáže odhaliť prekrývajúce sa alebo ľubovoľne tvarované zhluky.

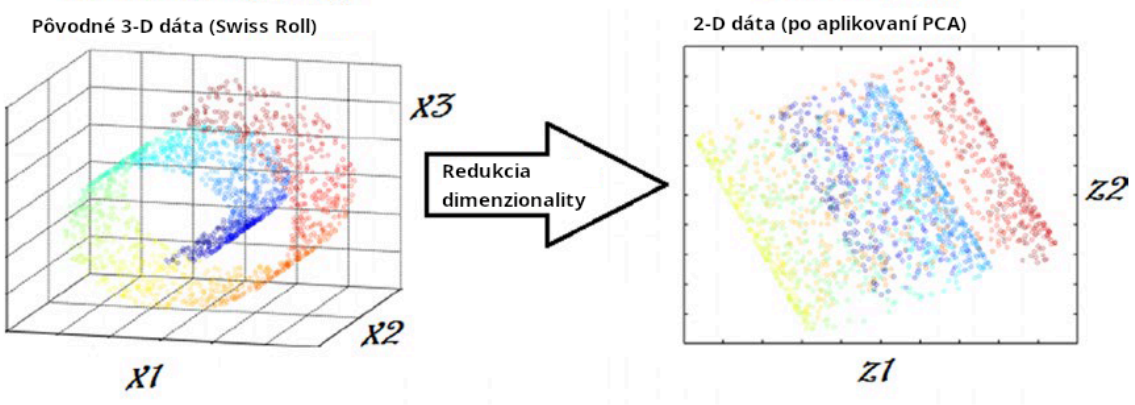
4.1.3 Redukcia dimenzionality

Akonáhle začneme extrahovať atribúty alebo príznaky zo vzoriek nespracovaných údajov, niekedy sa dokáže príznakový priestor nabaliť obrovským počtom príznakov. To predstavuje viacero výziev vrátane analýzy a vizualizácie údajov s tisíckami alebo miliónmi príznakov, čo robí tento príznakový priestor mimoriadne zložitým a predstavuje problémy s ohľadom na trénovacie modely, pamäť a priestorové obmedzenia. V skutočnosti sa toto

označuje ako "prekliatie dimenzionality". V týchto scenároch je možné použiť aj učiace metódy bez dohľadu, v ktorých znižujeme počet príznakov alebo atribútov pre každú vzorku údajov. Tieto metódy znižujú počet premenných príznakov extrahovaním alebo výberom množiny hlavných alebo reprezentatívnych príznakov. Na redukciu dimenzionality je k dispozícii viacero populárnych algoritmov:

- analýza hlavných komponentov PCA (Principal Component Analysis PCA)
- analýza nezávislých komponentov ICA (Independent Component Analysis ICA)
- diskriminačná analýza (Discriminant Analysis)

Obrázok 10 ukazuje výstup typického procesu redukcie znakov aplikovaného na 3D štruktúru Swiss Roll, ktorá má tri rozmery na získanie dvojrozmerného priestoru prvkov pre každú vzorku údajov pomocou PCA.



Obr. 10: Redukcia dimenzionality pomocou PCA

Z obrázka 10 je celkom jasné, že každá vzorka údajov mala pôvodne tri vlastnosti alebo dimenzie, a to D(x1, x2, x3) a po aplikácii PCA zredukujeme každú vzorku údajov z nášho súboru údajov na dve dimenzie, a to D'(z1, z2). Techniky redukcie dimenzionality možno klasifikovať do dvoch hlavných prístupov nasledovne.

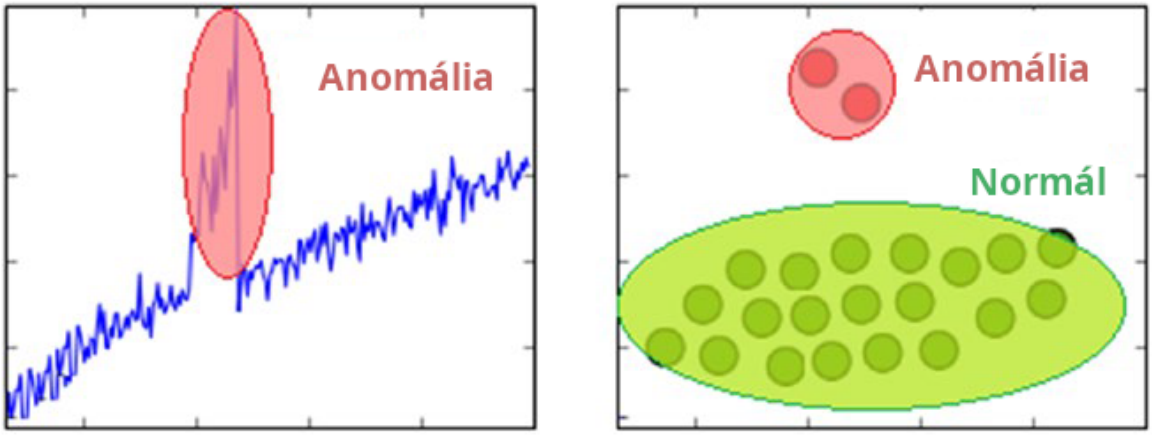
- Metódy **selekcie** alebo výberu príznakov: Špecifické príznaky sa vyberú pre každú vzorku údajov z pôvodného zoznamu príznakov a ostatné príznaky sa zahodia. V tomto procese sa negenerujú žiadne nové príznaky.
- Metódy **extrakcie** príznakov: Navrhujeme alebo extrahujeme nové príznaky z pôvodného zoznamu príznakov v údajoch. Zmenšená podmnožina príznakov teda bude obsahovať novo vygenerované príznaky, ktoré neboli súčasťou pôvodnej množiny príznakov. PCA patrí do tejto kategórie.

4.1.4 Detekcia anomálií

Proces detekcie anomálií sa nazýva aj detekcia odľahlých hodnôt, kde nás zaujíma zistenie výskytu zriedkavých udalostí alebo pozorovaní, ktoré sa zvyčajne bežne nevyskytujú na základe vzoriek historických údajov. Niekedy sa anomálie vyskytujú zriedkavo, a preto sú zriedkavé, a v iných prípadoch anomálie nemusia byť zriedkavé, ale môžu sa vyskytnúť vo veľmi krátkych dávkach v priebehu času, takže majú špecifické vzorce. Na detekciu anomálií možno použiť metódy učenia bez dohľadu, takže algoritmus trénujeme na trénovacom súbore údajov, ktorý má normálne, neanomálne vzorky údajov. Akonáhle sa naučí potrebné

reprezentácie údajov, vzory a vzťahy medzi atribútmi v normálnych vzorkách, pre akúkoľvek novú vzorku údajov by ju bol schopný identifikovať ako anomálny alebo normálny dátový bod pomocou svojich naučených znalostí. Obrázok 11 znázorňuje niektoré typické scenáre založené na detekcii anomálií. Medzi najčastejšie metódy patria:

- SVM s jednou triedou – One-class SVM
- Izolovaný les – Isolation Forrest

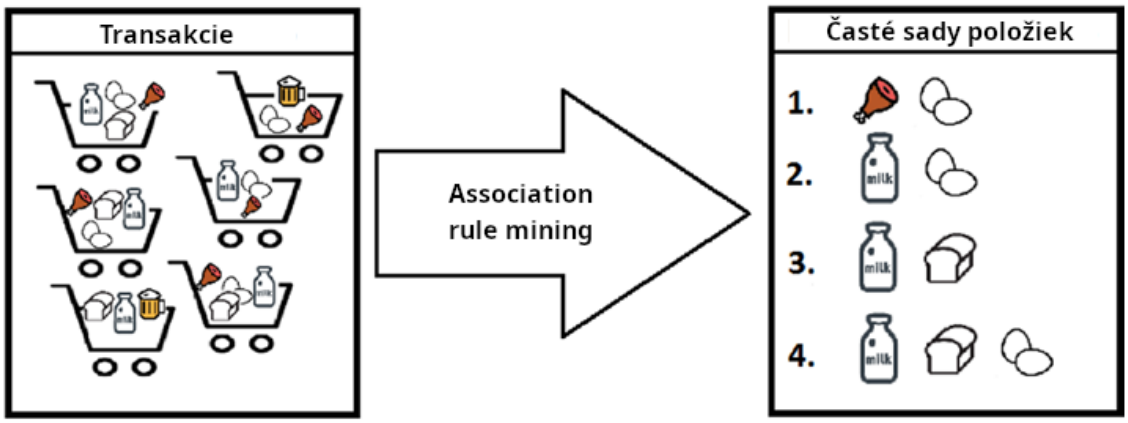


Obr. 11: Detekcia anomálií

Metódy založené na detekcii anomálií sú mimoriadne užitočné v rôznych prípadoch z reálneho sveta, ako je detekcia bezpečnostných útokov alebo narušení, podvody s kreditnými kartami, výrobné anomálie, problémy so sieťou a mnoho ďalších.

4.1.5 Association Rule-Mining

Dolovanie pravidiel asociácie je zvyčajne metóda získavania údajov používaná na skúmanie a analýzu veľkých transakčných súborov dát s cieľom nájsť vzory a pravidlá záujmu. Tieto vzorce predstavujú zaujímavé vzťahy a asociácie medzi rôznymi položkami naprieč transakciami. Získavanie pravidiel asociácie sa často nazýva aj analýza trhového koša, ktorá sa používa na analýzu nákupných vzorcov zákazníkov. Asociačné pravidlá pomáhajú pri zisťovaní a predpovedaní transakčných vzorcov na základe znalostí, ktoré sa získavajú z trénovacích transakcií. Pomocou tejto techniky môžeme odpovedať na otázky, aké položky majú ľudia tendenciu kupovať spoločne, čím naznačujeme časté sady položiek. Môžeme tiež spájať alebo korelovať produkty a predmety, t. j. poznatky, ako ľudia, ktorí kupujú pivo, majú tendenciu kupovať kuracie krídelká v reštaurácii. Obrázok 12 ukazuje, ako by mala typická metóda dolovania pravidiel asociácie fungovať ideálne na transakčnom súbore údajov.



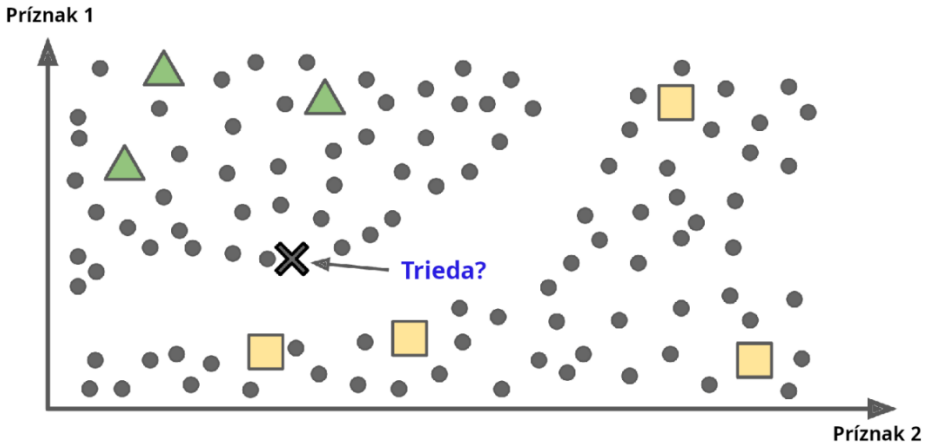
Obr. 12: Association rule mining

Z obrázka 12 možno jasne vidieť, že na základe rôznych transakcií zákazníkov za určité časové obdobie sme získali položky, ktoré sú úzko spojené a zákazníci majú tendenciu ich kupovať spoločne. Niektoré z týchto častých sád predmetov sú zobrazené ako {mäso, vajcia}, {mlieko, vajcia} a tak ďalej. Kritérium určovania kvalitných asociačných pravidiel alebo častých sád položiek sa zvyčajne vykonáva pomocou metrík, ako je podpora, dôvera a nárast. Ide o metódu bez dohľadu, pretože vopred netušíme, aké sú časté sady položiek alebo ktoré položky sú silnejšie spojené s ktorými položkami. Až po aplikovaní rozhodovacích algoritmov môžeme zistiť a predpovedať produkty alebo položky, ktoré sú navzájom úzko spojené, a nájsť podmienené pravdepodobnostné závislosti. Medzi najčastejšie používané rozhodovacie algoritmy patria:

- Apriori
- FP-growth
- Eclat

4.1.6 Semi-supervised learning (učenie s čiastočným dohľadom)

Keďže označovanie alebo značkovanie/labeling dát je zvyčajne časovo náročné a nákladné, často je k dispozícii veľa neoznačených inštancií alebo príkladov a málo označených inštancií. Niektoré algoritmy dokážu pracovať s dátami, ktoré sú čiastočne označené. Toto sa nazýva učenie s čiastočným dohľadom, ktorého príklad s dvoma triedami je znázornený na obrázku 13. Neoznačené príklady-sivé krúžky pomáhajú klasifikovať novú inštanciu-krížik do triedy „trojuholník“ a nie do triedy „štvorec“ aj keď je bližšie k označeným príkladom triedy „štvorec“.



Obr.13: Príklad učenia s čiastočným dohľadom

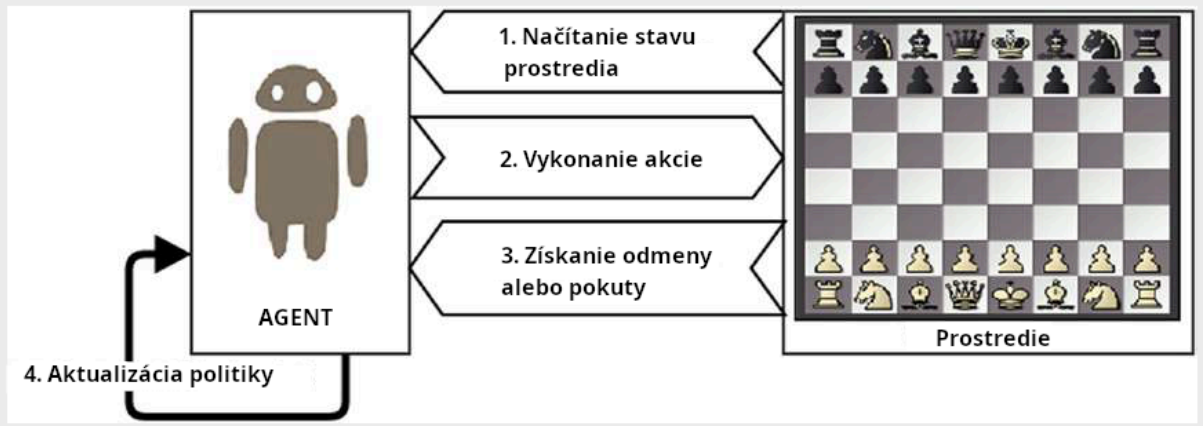
Niektoré služby hosťovania fotografií, ako napríklad Fotky Google, sú toho dobrým príkladom. Keď do služby nahráte všetky svoje rodinné fotografie, automaticky rozpozná, že na fotografiách 1, 5 a 11 sa zobrazuje tá istá osoba A, zatiaľ čo iná osoba B sa zobrazuje na fotografiách 2, 5 a 7. Toto je nekontrolovaná časť algoritmu (zhlukovanie). Teraz všetko, čo systém potrebuje, je, aby ste mu povedali, kto sú títo ľudia. Stačí pridať jednu značku na osobu 4 a dokáže pomenovať všetkých na každej fotografii, čo je užitočné pri vyhľadávaní fotografií. Väčšina algoritmov učenia s čiastočným dohľadom je kombináciou algoritmov bez dohľadu a s dohľadom. Napríklad siete *Deep Belief Networks* (DBN) sú založené na nekontrolovaných komponentoch nazývaných *Restricted Boltzmann machines* (RBM) naskladané na seba. RBM sa trénujú postupne bez dohľadu a potom sa celý systém dolaďuje pomocou techník učenia s dohľadom.

4.1.7 Reinforcement learning (učenie na základe odmeny)

Metódy učenia na základe odmeny sa trochu líšia od konvenčných metód s dohľadom alebo bez dohľadu. Učiaci sa systém, ktorý sa v tomto kontexte nazýva agent, môže pozorovať prostredie, vyberať a vykonávať akcie a na základe týchto rozhodnutí dostávať odmeny (alebo tresty vo forme negatívnych odmien, ako je znázornené na obrázku 14). Potom sa musí sám naučiť, aká je najlepšia stratégia, nazývaná politika/zásada, aby časom získal čo najväčšiu odmenu. Politika definuje, akú akciu by mal agent zvoliť, keď je v danej situácii. Hlavné kroky metódy učenia na základe odmeny sú uvedené nasledovne.

1. príprava agenta so súborom počiatočných politík a stratégie
2. sledovanie prostredia a aktuálny stav
3. výber optimálnej politiky a vykonanie akcie
4. získanie zodpovedajúcej odmeny/pokuty
5. aktualizácia politiky/zásady ak je to potrebné
6. iteratívne opakovanie krokov 2-5, kým sa agent nenaučí najoptimálnejšie politiky

Príkladom skutočného problému môže byť situácia, keď sa snažíte prinútiť robota alebo stroj, aby sa naučil hrať šach. V tomto prípade by agentom bol robot a prostredie a stavy by boli šachovnica a pozície šachových figúrok. Vhodná metodika učenia na základe odmeny je znázornená na obrázku



Obr. 14: Reinforcement learning – trénovanie agenta aby vedel hrať šach

Mnoho robotov napríklad implementuje algoritmy učenia na základe odmeny, aby sa naučili chodiť. Program AlphaGo spoločnosti DeepMind je tiež dobrým príkladom tohto typu učenia: dostal sa na titulky novín v máji 2017, keď porazil majstra sveta Ke Jie v hre Go. Svoju víťaznú politiku sa naučila analýzou miliónov hier a potom hrala veľa hier proti sebe. Všimnite si, že učenie bolo počas zápasov proti šampiónovi vypnuté; AlphaGo len uplatňoval politiku, ktorú sa naučil.

4.2 Rozdelenie algoritmov a metód na základe typu útoku

4.2.1 Detekcia narušenia siete (Network Intrusion Detection (NIDS))

Najčastejšie metódy:

- Štatistické metódy:
  - Detekcia anomálií: modelovanie základných parametrov siete (stredná hodnota/rozptyl veľkosti paketov a frekvencie) pomocou Z-skóre alebo „Tukey fences“
  - Analýza entropie: detekcia DDoS (nízka entropia v cieľových IP adresách) alebo skenovanie (vysoká entropia v portoch)
  - Modely časových postupností: ARMA, ARIMA ARFIMA pre predpovedanie objemu sieťovej prevádzky
- ML metódy:
  - Supervised learning:
    - Random Forest/GBT: vysoká presnosť pri klasifikácii útokov (napr. DoS, Probe)
    - SVM: efektívne pri dátach s veľkým počtom príznakov (napr. NSL-KDD dataset)
  - Unsupervised learning:
    - Klastrovanie: k-means, DBSCAN – zgrupovanie podobných vzorcov prevádzky

- PCA: využíva redukciu dimenzionality na identifikovanie tokov outlier-s (bodov ležiacich mimo „štandardných“ pozícií)

- Deep Learning metódy:
  - LSTM/GRU: detekcia sekvenčných útokov (napr. brute-force útoky v čase)
  - Autoenkóder: označuje rekonštrukciu prevádzky s vysokou chybou ako anomálie
    - Graph Neural Networks (GNNs): modeluje sieťovú topológiu za účelom detekcie laterálneho pohybu
    - Reinforcement Learning (RL): Adaptívne ladenie politík pre dynamickú obranu

4.2.2 Detekcia Malware

Najpoužívanejšie metódy:

- Štatistické metódy:
  - Signature Hashing: Fuzzy hashing (ssdeep) pre detekciu variantov
  - Analýza hlavičky PE: Entropia sekcií (.text, .data) pre detekciu enkapsulácie
- ML metódy:
  - Založené na príznakoch:
    - Statické: N-gramy z bajt-kódu, importovaných tabuliek a reťazcov
    - Dynamické: Sekvencie volania API, zmeny registra (XGBoost, LightGBM)
  - Deep Learning metódy:
    - CNN: klasifikácia malvéru založená na obrázkoch (binárne reprezentované ako RGB obrázok)
    - RNN/Transformer: analýza sekvencie volaní API

4.2.3 Detekcia Spam/Ham & Phishingu

Najpoužívanejšie metódy:

- Štatistické metódy:
  - bayesovské filtre: Naive Bayes na hodnotenie pravdepodobnosti tokenu (slovo/URL).
  - TF-IDF: určuje váhu kľúčových slov (napr. "urgentné", "faktúra").
- ML metódy:
  - SVM/RF: klasifikácia pomocou lexikálnych príznakov (hlavička, telo).
  - Logistická regresia: východisko pre blacklist URL/domén
- Deep Learning metódy:
  - BERT/Transformers: kontextová klasifikácia phishingových e-mailov
  - CNN: odhaľuje škodlivé logá v e-mailoch

4.2.4 Detekcia spoofingu

Najpoužívanejšie metódy:

- Štatistické metódy:
  - SPF/DKIM/DMARC: využíva protokoly na overovanie e-mailov (kontroly štatistickej politiky).
  - Geolokačná analýza: využíva nesúlad IP-geo (napr. prihlásenie z neočakávanej krajiny).
- ML metódy:
  - IP spoofing:
    - Ensemble metódy: detekcia anomálií TCP SYN (napr. náhodné zdrojové IP adresy)
  - Email spoofing:
    - RNN: analyzuje nezrovnalosti hlavičky (napr. "Odpovedať" vs. "Od").

4.2.5 Detekcia botnetov

Najpoužívanejšie metódy:

- Štatistické metódy:
  - Korelácia toku: detegujte periodické výstražné znamenia-beaconing (napr. entropiu intervalov dátového toku)
  - Analýza DNS: nezvyčajné hodnoty TTL alebo rýchlosti NXDOMAIN
- ML metódy:
  - Supervised learning:
    - Random Forrest, SVM: klasifikujte prevádzku na základe veľkosti paketu, časovania, portov
  - Unsupervised learning:
    - Klastrovanie, k-means: využíva zoskupenie podobných komunikačných vzorov C&C
- Deep learning metódy:
    - LSTM: využíva modelovanie časové C&C signálov (heartbeat signals)
    - GNNs: detekcia topológií P2P botnetov

4.3 Rozdelenie algoritmov a metód podľa použitia neurónovej siete

Táto kapitola opisuje metódy a algoritmy strojového učenia pre detekciu anomálií v počítačových sieťach, ktoré delí na klasické metódy ML a metódy používajúce neurónové siete. Pre každú triedu metód uvádza jej výhody a nevýhody, a tiež najpoužívanejšie metódy. Pre jednotlivé metódy dáva odporúčanie – na detegovanie akého typu útoku sú vhodné. Pre metódy uvádza aj hodnotiace kritériá a dáva celkové odporúčanie metód.

Pre zabezpečenie bezchybnej a spoľahlivej činnosti súčasných počítačových sietí a informačných systémov je nevyhnutné chrániť tieto systémy pred útokmi rôzneho typu. Tieto útoky sa môžu prejaviť výskytom neobvyklých udalostí a situácií, alebo nezvyčajného vzoru sieťovej prevádzky. Takéto informácie sa získavajú priebežne sledovaním sieťovej prevádzky a zaznamenávaním logovacích správ z rôznych zariadení. V súčasných sieťach sa táto

činnosť spravidla deje automatizovane pomocou softvérových prostriedkov – detegujú sa tzv. anomálie. Za anomáliou sa môže, ale nemusí skrývať útok. Systém na anomáliu upozorní administrátora siete, alebo sám reaguje na vzniknutú situáciu.

Na detekciu takýchto anomálií v sieti (nezvyčajných udalostí a situácií v sieti, ako aj na detekciu neobvyklého správania sa používateľov) sa používajú metódy strojového učenia ML (Machine Learning). Tieto metódy delíme na dve veľké skupiny – klasické metódy ML a metódy ktoré využívajú neurónové siete, resp. hlboké neurónové siete DNN (Deep Neural Networks). Obe kategórie majú svoje výhody a nevýhody a prípady, keď je vhodné ich použiť. V jednoduchších úlohách je často postačujúce použiť rýchle klasické metódy ML, kým pre detekciu komplexných vzorov sa používajú metódy založené na neurónových sieťach, ktoré tu dosahujú lepšie výsledky.

V nasledujúcich častiach opíšeme algoritmy a metódy strojového učenia vhodné pre detekciu a predikciu bezpečnostných a operačných anomálií. Opíšeme ako sú definované bezpečnostné anomálie a operačné anomálie, aká je rozdiel medzi detekciou a predikciou anomálií. Metódy rozdelíme na dve skupiny – klasické algoritmy strojového učenia a algoritmy strojového učenia založené na neurónových sieťach. Opíšeme princíp algoritmov, potrebné vstupné údaje a ich prípravu, parametre algoritmov a proces trénovania. A príklady použitia na detekciu útokov.

4.3.1 Klasické algoritmy strojového učenia pre detekciu anomálií

Za klasické algoritmy strojového učenia považujeme algoritmy, ktoré nepoužívajú neurónové siete. Analyzujeme len algoritmy používané pre detekciu anomálií v počítačových sieťach

Strojové učenie (Machine Learning, ML) zohráva kľúčovú úlohu v oblasti kybernetickej bezpečnosti a správy IT systémov. Hoci hlboké neurónové siete (Deep Neural Networks, DNN) dosiahli významné výsledky v mnohých úlohách, ich nasadenie si často vyžaduje značné výpočtové zdroje, rozsiahle množstvo označených dát a odborné znalosti. V mnohých prípadoch však postačujú klasické algoritmy strojového učenia, ako sú rozhodovacie stromy, SVM, KNN, alebo metódy zhlukovania, ktoré môžu byť efektívnejšie, transparentnejšie a ľahšie interpretovateľné. Sústredíme sa na metódy používané pre detegovanie anomálií. Najprv uvedieme základné pojmy.

Definícia bezpečnostných anomálií a operačných anomálií

- **Bezpečnostné anomálie** sú neobvyklé udalosti alebo vzory v sieťovej alebo systémovej aktivite, ktoré môžu indikovať kybernetický útok, ako napr. prienik, malvér, DoS/DDoS, exfiltrácia dát alebo neoprávnený prístup.
- **Operačné anomálie** sa týkajú odchýlok v bežnom chode IT systémov, ako sú výpadky služieb, výnimočná spotreba prostriedkov, chyby softvéru alebo neplánované reštarty zariadení. Môžu mať technický pôvod, bez priameho bezpečnostného základu.

Detekcia anomálií vs. predikcia anomálií

- **Detekcia** anomálií znamená identifikáciu odchýlok v reálnom čase alebo spätne v historických dátach. Je to reaktívny prístup, ktorý sa zameriava na aktuálne alebo nedávne incidenty.
- **Predikcia** anomálií je proaktívna metóda, ktorej cieľom je predpovedať vznik anomálie na základe trendov, korelácií a vývoja dát. Využíva sa najmä pri prevencii výpadkov alebo kybernetických útokov.

4.3.1.1 Klasické algoritmy s učiteľom, bez učiteľa a s čiastočným učiteľom

V tejto časti rozdelíme klasické metódy strojového učenia do kategórií podľa toho, či si príslušné metódy vyžadujú označenie trénovacích údajov.

Pri riešení praktických úloh býva často náročné získať označené údaje. Expert alebo pomocný systém musí priradiť údajom značky (labels) - či sa jedná o dáta získané počas útoku, alebo sú to údaje o bežnej – normálnej situácii, keď neprebiehal žiadny útok.

Metódy strojového učenia delíme na tri základné typy:

- Učenie s učiteľom (supervised learning) – vyžaduje dátovú množinu s označenými príkladmi normálneho a anomálneho správania.
- Učenie bez učiteľa (unsupervised learning) – používa neoznačené dáta na identifikáciu vzorov a odľahlých hodnôt.
- Učenie s čiastočným učiteľom (semi-supervised learning) – kombinuje výhody oboch prístupov, čo je vhodné pri nedostatku označených dát.

Do jednotlivých tried strojového učenia môžeme priradiť nasledujúce algoritmy:

1. Supervised algoritmy (učenie s učiteľom – označené sú všetky trénovacie údaje):

- Rozhodovacie stromy (Decision Trees)
- Random Forest
- Support Vector Machines (SVM)
- k-Nearest Neighbors (KNN)
- Logistická regresia

2. Unsupervised algoritmy (učenie bez učiteľa – dáta nie sú označené):

- K-means zhlukovanie
- DBSCAN
- Principal Component Analysis (PCA)
- Isolation Forest

3. Semi-supervised prístupy:

- Kombinujú známe prípady s neoznačenými dátami.

4.3.1.2 Vstupné údaje pre algoritmy – zdroje a príprava dát

Pre algoritmy ML sa používajú vstupné údaje získané z rôznych zdrojov:

- Sieťové logy (napr. NetFlow, firewall, IDS/IPS)
- Systémové logy (syslog, Windows Event Log)
- Dáta o výkonnosti systémov (CPU, pamäť, I/O)
- Záznamy o prístupoch a autentifikáciách

Pred použitím údajov na trénovanie ML modelu je potrebné dáta pripraviť a predspracovať v nasledujúcich krokoch:

- Zber a agregácia údajov z rôznych zdrojov.
- Čistenie dát – odstránenie chýb, duplikátov a irelevantných údajov.

- Extrahovanie čŕt (feature engineering) – napr. počet spojení za minútu, veľkosť prenesených dát, počet zlyhaných prihlásení.
- Normalizácia alebo škálovanie hodnôt pre algoritmy citlivé na mierku.

Označenie údajov (labeling) – v prípade supervised učenia.

4.3.1.3 Parametre algoritmov a tréningový proces

Pre dosiahnutie čo najväčšej úspešnosti rozpoznania anomálií sa odporúča pre jednotlivé metódy vykonať väčší počet experimentov a pritom nastaviť vhodne nasledujúce parametre:

- **Rozhodovacie stromy / Random Forest:** hĺbka stromu, počet stromov, minimálny počet vzoriek na rozdelenie.
- **SVM:** voľba jadra (kernel), parameter C (regularizácia), γ.
- **KNN:** počet susedov *k*, metrika vzdialenosti (napr. Euklidovská).
- **K-means:** počet klastrov *k*, počiatočná inicializácia centroidov.
- **Isolation Forest:** počet stromov, počet vzoriek na strom.

Tréningový proces pozostáva s nasledujúcich krokov:

- Rozdelenie dát na trénovaciu a testovaciu sadu.
- Optimalizácia parametrov príslušného algoritmu pomocou cross-validácie.

Vyhodnotenie výkonnosti pomocou metrík ako presnosť detegovania anomálií, recall, F1 skóre alebo AUC.

4.3.1.4 Odporúčania klasických ML metód na rozpoznanie rôznych útokov

Uvádzame vybrané typy útokov a klasické metódy ML vhodné na rozpoznanie týchto útokov, taktiež uvádzame typ dát, ktoré je potrebné pre tieto metódy pripraviť:

- DoS útoky – klasifikácia na základe počtu požiadaviek za časovú jednotku (Random Forest).
- Port scanning – detekcia neobvyklého počtu krátkych TCP spojení (K-means, DBSCAN).
- Vnútorné hrozby – anomálne správanie zamestnancov (Isolation Forest, PCA).
- Malvér v sieti – vzory komunikácie C2 serverov (SVM, Decision Trees).
- Predikcia výpadkov – regresné modely alebo časové rady (napr. ARIMA alebo jednoduché ML regrese).

4.3.1.5 Kritériá na porovnanie algoritmov

Pre porovnanie ML algoritmov sa používajú kritériá v nasledujúcej tabuľke:

Tabuľka 2 Kritériá pre porovnanie ML algoritmov

Kritérium	Popis
Presnosť detekcie	Percento správne klasifikovaných anomálií.
Falošné poplachy (false positives)	Počet bežných udalostí označených ako hrozby.
Časová a výpočtová náročnosť	Rýchlosť učenia a detekcie, potreba HW.
Škálovateľnosť	Možnosť nasadenia na veľké dátové množiny.
Interpretovateľnosť	Možnosť vysvetliť rozhodnutia modelu.
Odolnosť voči šumu v dátach	Robustnosť voči neúplným alebo chybným údajom.
Požiadavky na označené dáta	Miera závislosti od labeled datasetov.

Tabuľka 2: Typy IP útokov

4.3.1.6 Odporúčaný algoritmus

Isolation Forest – je jedným z najvhodnejších algoritmov pre detekciu anomálií v bezpečnostných aj operačných dátach, a to najmä pre jeho:

- schopnosť pracovať bez označených dát (unsupervised),
- nízku výpočtovú náročnosť,
- výbornú detekciu odľahlých bodov,
- robustnosť voči šumu.

Pre klasifikačné úlohy s označenými dátami odporúčame Random Forest vďaka vysokej presnosti, dobrej generalizácii a schopnosti pracovať aj s neúplnými údajmi.

4.3.2 Algoritmy strojového učenia používajúce neurónové siete

Táto časť analyzuje algoritmy strojového učenia (ML), postavené na hlbokých neurónových sieťach , ktoré sú vhodné pre riešenie úlohy detekcie a predikcie bezpečnostných a operačných anomálií.

V prostredí dynamickej a komplexnej infraštruktúry informačných systémov je identifikácia a predikcia anomálií kľúčová pre zabezpečenie ich stability a bezpečnosti. Hlboké neurónové siete DNN (Deep Neural Networks) predstavujú efektívny prístup k riešeniu týchto úloh vďaka schopnosti modelovať nelineárne závislosti a učiť sa reprezentácie zo surových dát. Tento dokument sa zameriava na vysvetlenie princípov využitia DNN v oblasti detekcie a predikcie bezpečnostných a operačných anomálií. Uvádza analýzu a návrh vhodných algoritmov ML/AI, opis princípu, zdroje vstupných dát, ich prípravu a používané datasety, uvádza parametre modelov, popis procesu trénovania modelov.

4.3.2.1 Princíp algoritmov hlbokého učenia

Algoritmy založené na hlbokom učení sa opierajú o viacvrstvové architektúry neurónových sietí (napr. MLP, LSTM, Autoencodery, CNN, RNN), ktoré umožňujú:

- Automatické učenie reprezentácií (modelov, vzorov) zo vstupných údajov.
- Generalizáciu z príkladov – systém sa učí z historických prípadov normálneho a anomálneho správania.
- Zachytávanie časových a priestorových vzorov – čo je dôležité najmä pri predikcii.

4.3.2.2 Princíp algoritmov hlbokého učenia

K najznámejším modelom používajúcim neurónové siete DNN patria:

- **Autoencodery**: Trénujú sa na "normálnych" dátach a identifikujú anomálie podľa veľkej rekonštrukčnej chyby.
- **LSTM** (Long Short-Term Memory): Efektívne pri spracovaní časových radov, užitočné pri predikcii operačných výpadkov.
- **CNN** (Convolutional Neural Networks): Môžu byť použité pre analýzu sieťových tokov znázornených vo forme obrazov.
- **GANs** (Generative Adversarial Networks): Pre syntézu a detekciu zložitých typov anomálií.

4.3.2.3 Vstupné údaje a ich príprava

Typickými zdrojmi vstupných údajov pre trénovanie uvedených sietí sú:

- Sieťové logy (NetFlow, PCAP, firewall logy)
- Systémové logy (CPU, pamäť, I/O operácie)
- Aplikačné logy (web server, databázy)
- Metriky z monitorovacích systémov (napr. Prometheus, Nagios, Zabbix, Graphite)

Príprava vstupných dát pred ich použitím na trénovanie obsahuje:

- **Čistenie**: Odstránenie chýbajúcich alebo irelevantných údajov.
- **Normalizácia/škálovanie**: Min-Max alebo Z-score transformácia.
- **Extrahovanie vlastností**: Vytváranie agregačných metrických okien, štatistických charakteristík.
- **Sekvencovanie**: Pre modely pracujúce s časovými radmi.
- **Labelovanie**: Pri supervisovanom učení je potrebné mať anotované dáta (normálne vs. anomálne).

4.3.2.4 Parametre algoritmov a trénovanie

Pri trénovaní pre dosiahnutie čo najlepších výsledkov modelu možno voliť a meniť parametre modelov:

- **Počet vrstiev a neurónov**
- **Aktivačné funkcie** (ReLU, Sigmoid, Tanh)
- **Optimalizačný algoritmus** (Adam, RMSprop)
- **Batch size a learning rate**
- **Dropout/maskovanie** pre regularizáciu

Trénovací proces pozostáva s nasledujúcich krokov:

- **Rozdelenie dát na trénovaciu, validačnú a testovaciu množinu**
- **Trénovanie modelu** pomocou spätného šírenia chyby (backpropagation)
- **Ladenie hyperparametrov** (napr. cez Grid Search, Bayesian Optimization)
- **Hodnotenie modelu** pomocou metrík: presnosť, recall, F1-score, ROC-AUC

4.3.2.5 Odporúčania metód s neurónovými sieťami na detekciu útokov

Uvádzame vybrané typy útokov v počítačových sieťach a metódy ML založené na DNN, ktoré sú vhodné na rozpoznanie týchto útokov:

- **DDoS útoky**: LSTM deteguje náhle skoky v požiadavkách.
- **Botnet aktivita**: Autoencoder identifikuje neštandardné vzory v sieťovej komunikácii.
- **Skryté prieniky (APT)**: Kombinácia CNN a RNN pre detekciu dlhodobých vzorcov.
- **SQL injection**: NLP modely nad HTTP requestmi dokážu rozpoznať zneužitie syntaxe.
- **Malware v sieti**: GAN-based detekcia škodlivých tokov na základe behaviorálnych vzorov.

4.3.2.6 Kritériá pre porovnanie algoritmov

Pre porovnanie ML algoritmov sa používajú nasledujúce kritériá:

- **Presnosť detekcie/predikcie** (Precision, Recall, F1-score, Accuracy)
- **Latencia a výpočtová náročnosť**
- **Možnosť online učenia (inference v reálnom čase)**
- **Robustnosť voči neoznačeným alebo nekompletným dátam**
- **Schopnosť pracovať s nevyváženými dátami**
- **Interpretovateľnosť výsledkov**
- **Odolnosť voči útokom na model (napr. adversarial examples)**

4.3.2.7 Odporúčaný algoritmus

V závislosti od prípadu použitia možno odporučiť algoritmy:

- **Autoencodery** sú výbornou voľbou pre **neznačkované dáta** (unsupervised learning) a pre jednoduchú detekciu anomálií.
- **LSTM siete** sú vhodné pri **predikcii anomálií v časových radoch** a pri spracovaní sledov udalostí.
- **Hybridné modely** (napr. CNN-LSTM, Autoencoder-LSTM) kombinujú výhody oboch prístupov a sú dnes často najefektívnejšie.

Najvhodnejší algoritmus:

Autoencoder-LSTM hybrid – pre svoju schopnosť modelovať normálne správanie, odhaľovať anomálie na základe rekonštrukčnej chyby a súčasne sledovať časový kontext. Výborne sa osvedčuje v sieťovej bezpečnosti aj operačnej diagnostike.

5 ANALÝZA ŠTATISTICKÝCH PARAMETROV NA VYBRANÝCH DDOS ÚTOKOCH

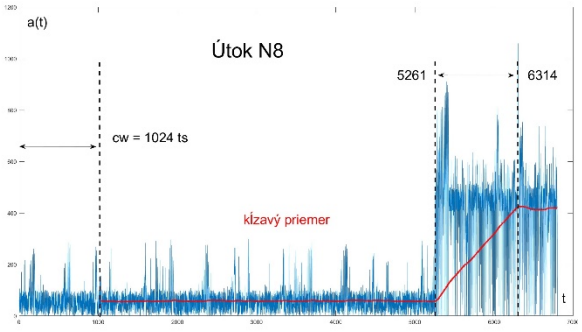
5.1 Popis použitých záznamov DDoS útokov pre testovanie reakcie štatistických koeficientov

Na rôznych záznamoch DDoS útokov budeme prezentovať reakciu vybraných štatistických parametrov. Vynechali sme množstvo experimentov, ktoré sme realizovali na rôznych simulovaných scenároch. Na týchto scenároch vybrané štatistické parameter fungovali „vzorovo“ a my sme nadobudli prvotnú predstavu o efektívnosti použitia jednotlivých parametrov pri rozpoznaní zmeny a nárastu simulovanej prevádzky. Pri nasadení na rozpoznanie reálneho útoku reakcie štatistických parametrov už neboli také jednoznačné. Pre prezentovanie sme vybrali osem rôznych záznamov DDoS útokov.

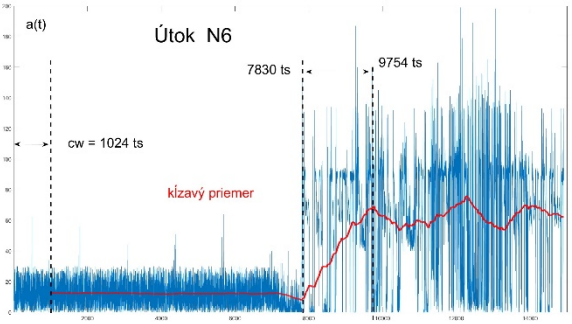
Prvé štyri záznamy predstavujú štandardné DDoS útoky (N-normal), pri ktorých sa parametre chovajú obdobne ako ma simulovaných dátach. Ďalšie dva záznamy predstavujú na prvý pohľad tiež štandardné útoky, avšak parametre sa pri nich chovajú podstatne odlišným neurčitým spôsobom (S-special). V poradí siedmy záznam je zástupcom tzv. Low Rate (LR) DDoS útokov, čiže útokov s nízkou priemernou intenzitou toku. Posledný záznam je útok, ktorý sa podľa nášho názoru nedá detegovať uvedenými štatistickými parametrami (P-ako problém). Pri záznamoch sme zároveň s počtom prírastkov paketov a(t) zobrazili aj ich kĺzavý priemer m(t) s výpočtovým oknom cw = 1024 ts.

Záznamy sú navzorkované do daných časových okien (ts-time slot), väčšinou ts = 1ms. Pre postupný výpočet parametrov sme použili posun tzv. výpočtového okna (cw – compute window) o jeden ts. Pri prevádzaní experimentov sme pre veľkosti výpočtových okien kvôli odhadu Hurstovho exponentu použili rôzne mocniny dvojky (256,512,1024,2048,4096). Po subjektívnom hodnotení priebehu hodnôt skúmaných parametrov sme sa rozhodli ďalej zaoberať hlavne s výpočtovým oknom o veľkosti cw = 1024 ts. Pri cw = 512 ts dochádzalo k významnému „rozvlneniu“ niektorých parametroch, pri cw = 2048 sa významne zväčšila doba výpočtu parametrov, hlavne Hurstovho exponentu.

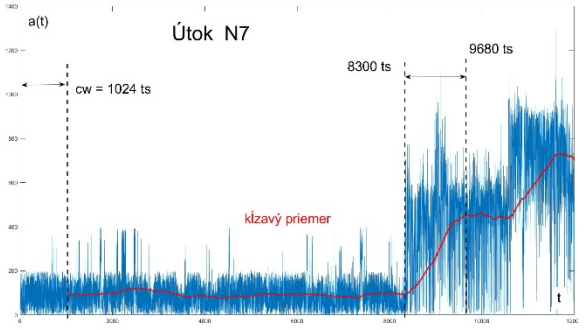
Na grafoch záznamov DDoS útokov sme zobrazili aj priebeh kĺzavého priemeru a zobrazili časový interval, v ktorom kĺzavý priemer m(t) nadobudol od začiatku útoku svoje lokálne maximum. Tento interval využijeme pri subjektívnom hodnotení ostatných štatistických koeficientov. Na všetkých záznamoch DDoS útokov môžeme pozorovať lineárny rast kĺzavého priemeru. Hľadáme koeficienty, ktorým pri začiatku útoku začnú rásť hodnoty rýchlejšie než lineárne.



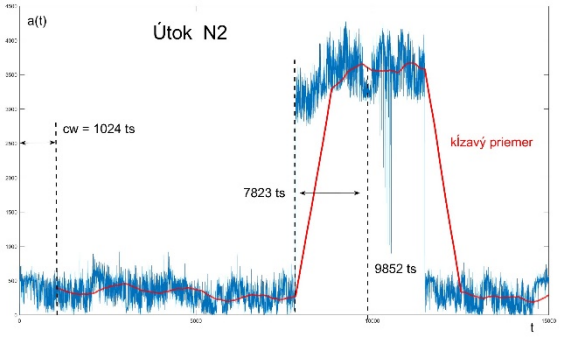
Obr. 15: Útok N8 a kĺzavý priemer



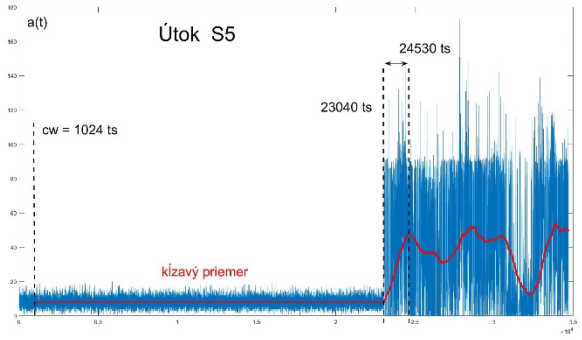
Obr. 16: Útok N6 a kĺzavý priemer



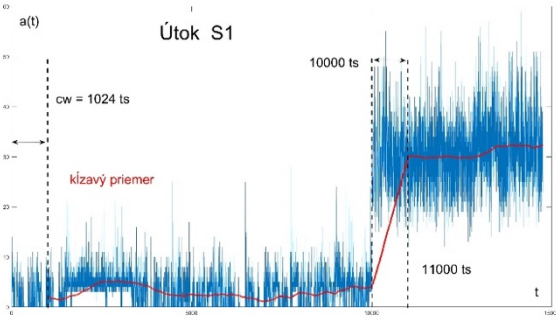
Obr. 17: Útok N7 a kĺzavý priemer



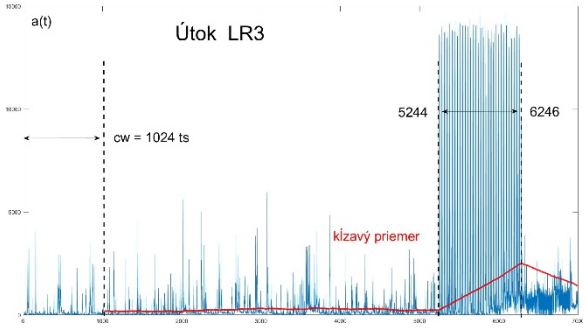
Obr. 18: Útok N2 a kĺzavý priemer



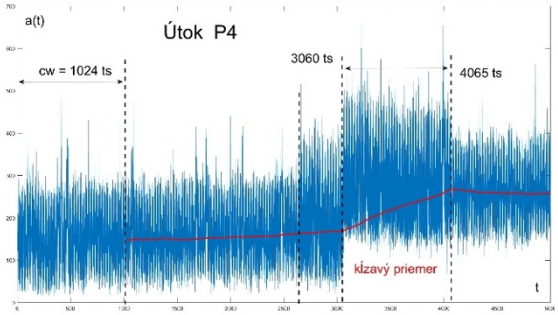
Obr. 19: Útok S5 a kĺzavý priemer



Obr. 20: Útok S1 a kĺzavý priemer



Obr. 21: Útok LR3 a kĺzavý priemer

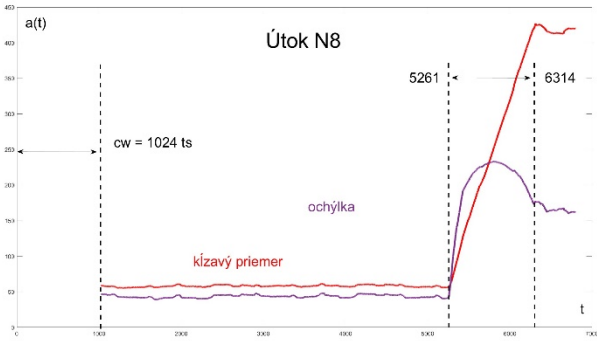


Obr. 22: Útok P4 a kĺzavý priemer

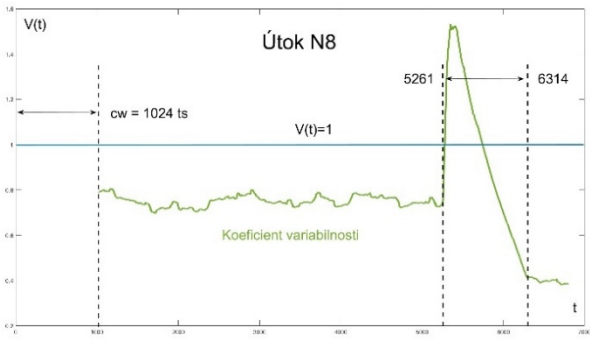
5.2 Použitie štatistických koeficientov

5.2.1 Variabilnosť, šikmosť a špicatosť

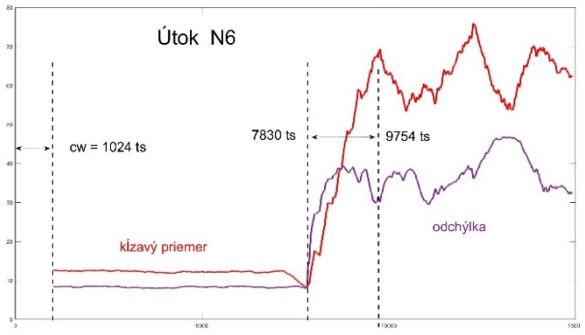
Ako prvými sa budeme zaoberať základnými štatistickými momentami, resp. koeficientami variabilnosti V(t), šikmosti K(t) a špicatosti Skw(t). Koeficient variabilnosti predstavuje podiel smerodajnej odchýlky a strednej hodnoty prírastkov v danom časovom okne cw. Variabilnosť vyjadruje mieru rozptylu prírastkov a(i) okolo strednej hodnoty. V prvom grafe zobrazíme priebeh kĺzavého priemeru a smerodajnej odchýlky, na druhom grafe sú hodnoty koeficientu variabilnosti:



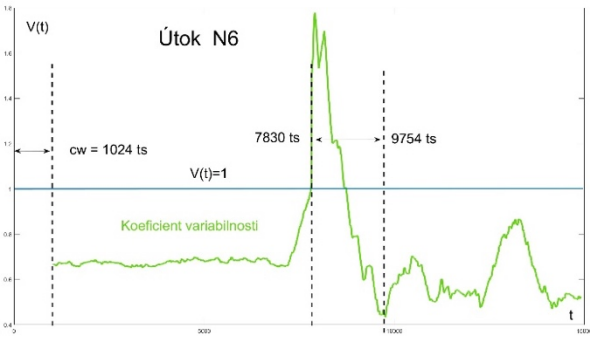
Obr. 23: Útok N8, priemer a odchýlka



Obr. 24: Útok N8, koeficient variabilnosti

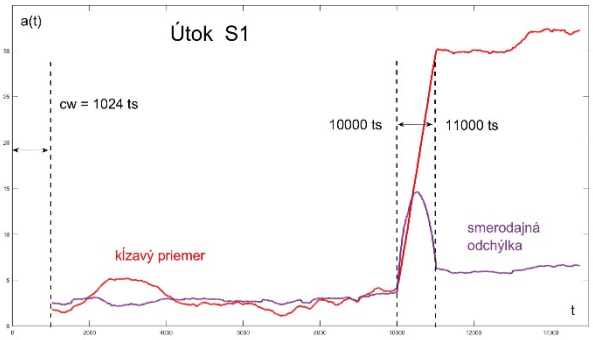


Obr. 25: Útok N6, priemer a odchýlka

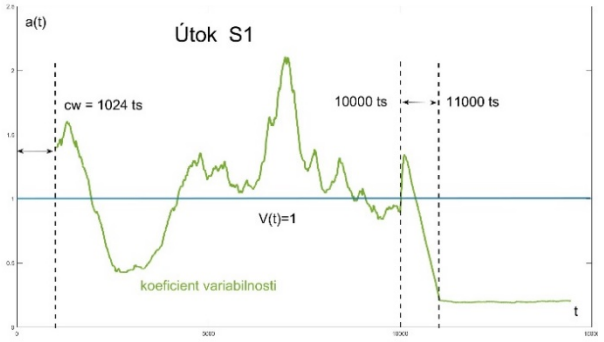


Obr. 26: Útok N6, koeficient variabilnosti

V prípade štandardných DDoS útokov (N2,N6,N7,N8) koeficient variabilnosti reagoval rovnako. Štandardná prevádzka predstavovala stacionárny tok s odchýlkou menšou ako stredná intenzita toku. Okamihu nástupu útoku odchýlka zareagovala rýchlejším nárastom hodnôt než bol lineárny rast kĺzavého priemeru, preto podiel odchýlky a priemeru nadobudol v okamihu útoku hodnotu $V(t_A) = 1$. Tento fakt by sa dal využiť pre rýchlu a jednoduchú detekciu pri takomto type útoku. To ale nie je pravidlo, na Obr.27 pri útoku S1 vidíme, že počas štandardnej prevádzky je narušená stacionarita a koeficient niekoľko krát pre útokom prekročil hodnotu $V(t_A) = 1$.

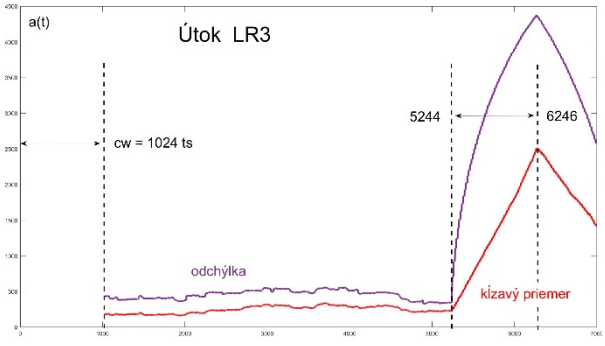


Obr. 27: Útok S1, priemer a odchýlka

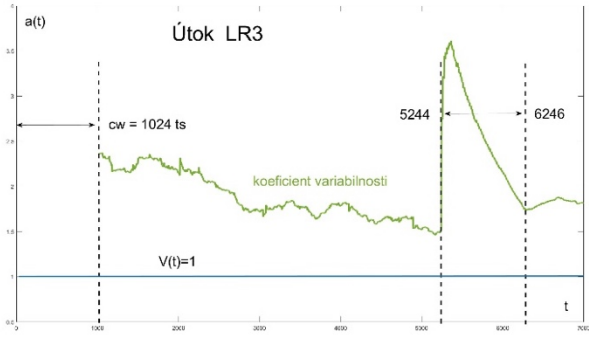


Obr. 28: Útok S1, koeficient variabilnosti

Na obr. 29 a 30 sú grafy priemeru, odchýlky a koeficientu variabilnosti pre tzv. Low Rate DDoS útok LR3:



Obr. 29: Útok LR3, priemer a odchýlka



Obr. 30: Útok LR3, koeficient variabilnosti

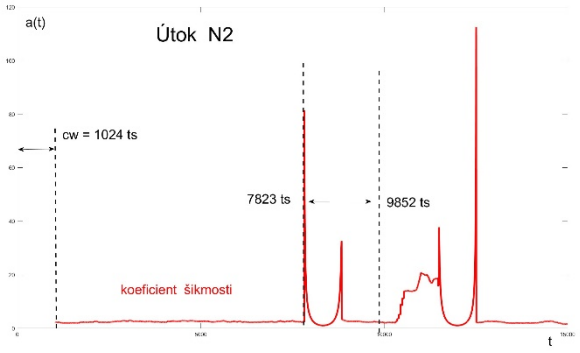
V prípade LR3 je počas bežnej prevádzky odchýlka väčšia než stredná intenzita toku, preto koeficient variabilnosti je počas celého priebehu V(t)>1 . Hodnota 1 sa teraz nedá použiť pre rýchlu detekciu. Ale v okamihu útoku hodnoty odchýlky opäť narástli rýchlejšie než lineárny rast strednej intenzity, čo sa prejavilo na výraznom skokovom náraste koeficientu variabilnosti. Táto zmena by sa mala dať rozpoznať pomocou predikčných metód (viď nasledujúca kapitola).

Ďalšími jednoduchými štatistickými parametrami sú koeficient šikmosti a koeficient špicatosti. Ide vlastne o tretí a štvrtý centrálny moment vyškálovaný smerodajnou odchýlkou:

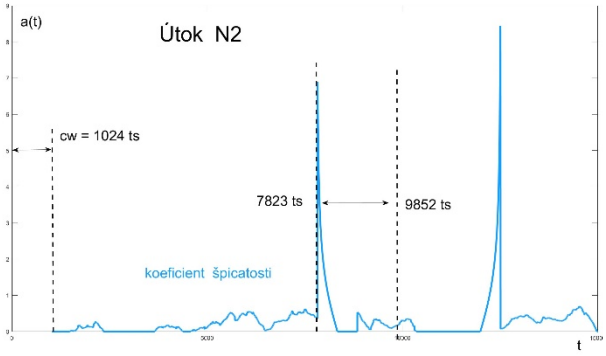
$$\mu_3 = \frac{1}{\sigma^3} E[X - EX]^3 \qquad \mu_4 = \frac{1}{\sigma^4} E[X - EX]^4 \qquad \sigma^2 = E[X - EX]^2$$

V našom prípade realizácie premennej X sú hodnoty prírastkov a(t) a koeficienty popisujú vlastnosti rozdelenia pravdepodobnosti prírastkov v danom výpočtovom okne cw. Pri postupnom načítavaní útočnej prevádzky do vyhodnocovaného okna cw sme predpokladali výraznú zmenu rozdelenia pravdepodobnosti a tým pádom aj zmenu koeficientov. Tento predpoklad sa potvrdil skoro u všetkých analyzovaných tokov. Obidva štatistické koeficienty zareagovali výrazným nárastom svojich hodnôt hneď na začiatku DDoS útoku.

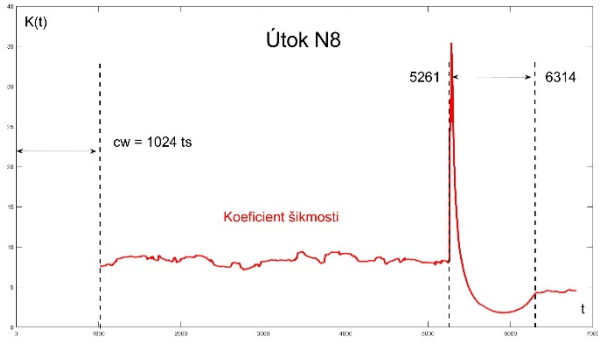
Ako prvé uvádzame grafy koeficientov aplikovaných na vybraných dvoch štandardných útokoch a následne niektoré problematické situácie.



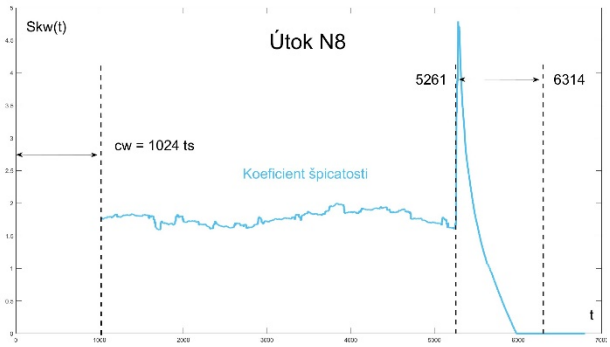
Obr. 31: Útok N2, koeficient šikmosti



Obr. 32: Útok N2, koeficient špicatosti



Obr. 33: Útok N8, koeficient šikmosti

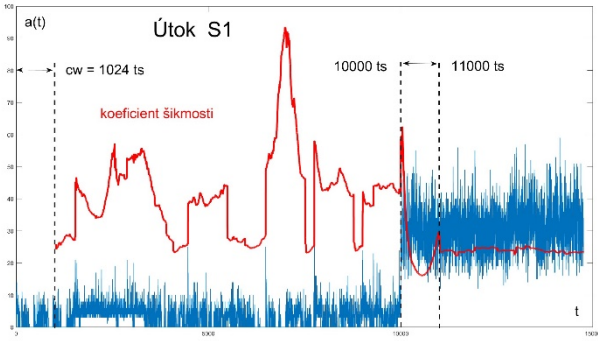


Obr. 34: Útok N8, koeficient špicatosti

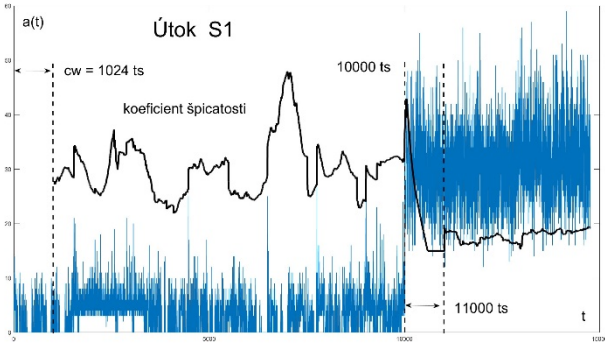
Pri štandardných útokoch typu N koeficienty šikmosti s špicatosti sa počas výpočtu na štandardnej prevádzke vďaka stacionárnosti len mierne vlnia, v okamihu nástupu DDoS útoku sa ich hodnoty výrazne zväčšia. Predpokladáme, že tento výrazný skok budeme vedieť strojovo rozpoznať pomocou vhodných predikčných metód.

Podstatne nepriaznivejšia situácia nastala pri neštandardných útokoch, napr. S1. Vďaka nestacionarite bežnej prevádzky a častému výskytu vysokých peakov obidva koeficienty silne kmitajú, vďaka čomu sa skok pri nástupe útoku stráca v celkovom priebehu toku hodnôt obidvoch koeficientov. Nástup DDoS útoku je v takomto prípade strojovo nerozpoznateľný.

Pre vizualizáciu reakcie koeficientov na zmenu charakteru bežnej prevádzky a na výskyt peakov sme zobrazili hodnoty koeficientov do spoločného grafu spolu s prírastkami toku S1.

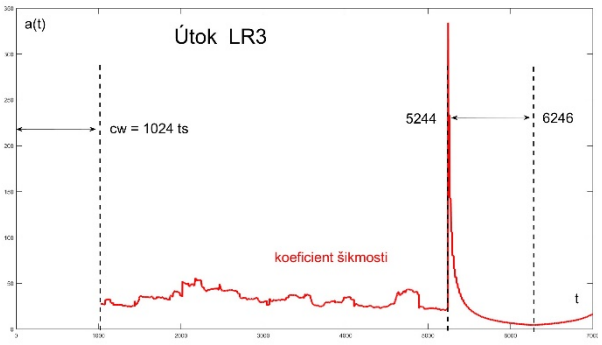


Obr. 35: Útok S1, koeficient šikmosti

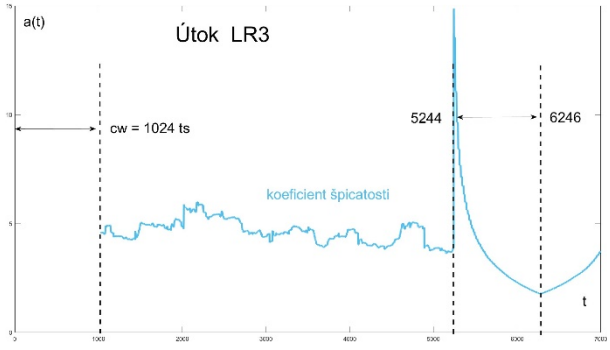


Obr. 36: Útok S1, koeficient špicatosti

Reakcia koeficientov pri útoku s nízkou strednou intenzitou LR3 dopadla rovnako pozitívne ako pri štandardných útokoch typu N:



Obr. 37: Útok LR3, koeficient šikmosti



Obr. 38: Útok LR3, koeficient špicatosti

Opäť vidíme možnosť strojovo rozpoznať výrazný nárast hodnôt koeficientov pri začiatku útoku od hodnôt popisujúcich bežnú prevádzku.

5.2.2 Detekčné hodnoty pre Autoregresiu a Koreláciu

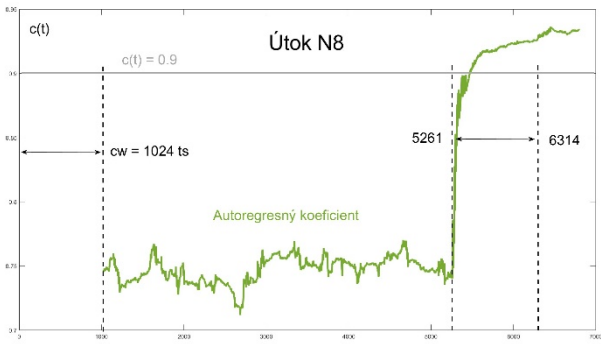
Ďalšie dva koeficienty popisujú autoregresnú a korelačnú závislosť medzi dvoma za sebou idúcimi cw oknami, resp. medzi náhodnými premennými X(t) a X(t-1), ktorých realizácie sú prírastky toku a(t) v uvažovaných oknách.

$$X(t) = c(t).X(t-1) + \varepsilon(t), \qquad \rho(t) = \frac{E[(X(t) - EX(t))(X(t-1) - EX(t-1))]}{\sigma(t)\sigma(t-1)}$$

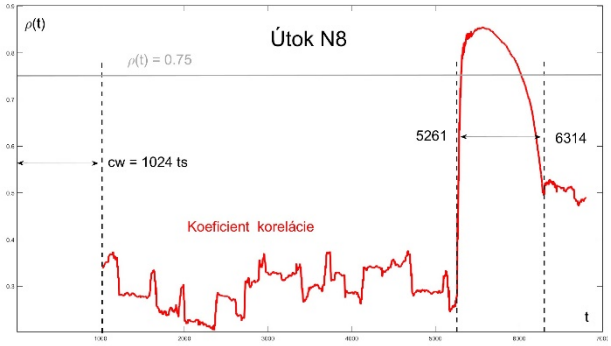
Veľkou výhodou tých koeficientov je, že ich hodnoty sa pohybujú v intervale (-1,1). V okamihu načítavania útoku do vyhodnocovaných cw okien pri väčšine útokoch začali ich hodnoty rýchlo rásť ku hornej hranici 1. Z tohto dôvodu sa je možne pre obidva koeficienty nastaviť určitú medznú hodnotu (threshold), ktorej prekročenie by signalizovalo začiatok DDoS útoku. Čím nižšia je táto hodnota, tým včasnejšia je signalizácia útoku, na druhú stranu moc nízke útoky môžu spôsobiť falošné hlásenia. Na základe prevedených experimentov a v snahe

identifikovať čo najväčšie množstvo DDoS útokov bez falošných hlásení sme nastavili hranicu pre autoregresný koeficient na c(t) = 0.9 a korelačný koeficient na ρ(t) = 0.75.

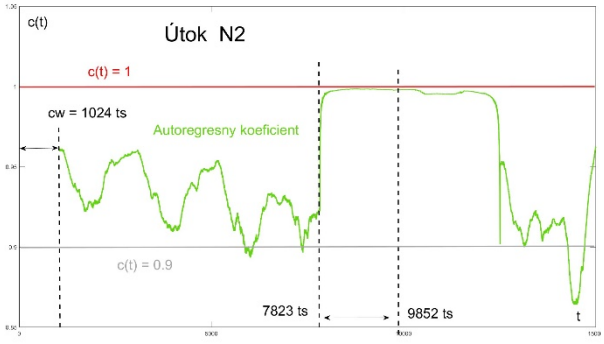
Na nasledujúcich grafoch sme opäť zobrazili vývoj koeficientov pre útoky N8, N2, S1 a LR3. Útok N8 predstavuje chovanie koeficientov pri väčšine štandardných útokov, na N2 vidíme štandardný útok na ktorom ale autokorelačný koeficient má všetky hodnoty väčšie ako 0.9 a detekcia sa nedá použiť. Pri neštandardnom útoku S1 obidva koeficienty pri útoku presiahli detekčné hladiny. Pre útok s nízkou intenzitou LR3 je táto metóda nepoužiteľná.



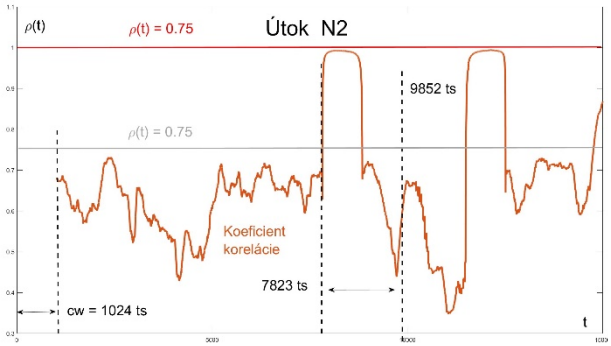
Obr. 39: Útok N8, Autoregresný koeficient



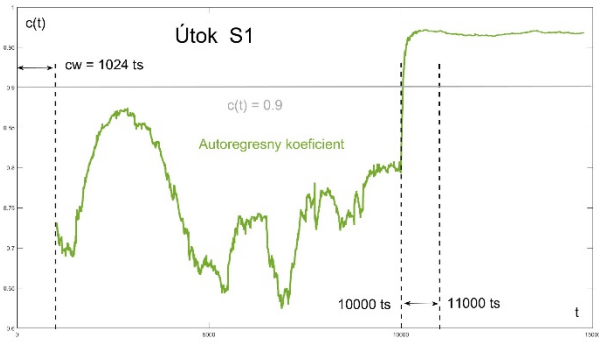
Obr. 40: Útok N8, koeficient Korelácie



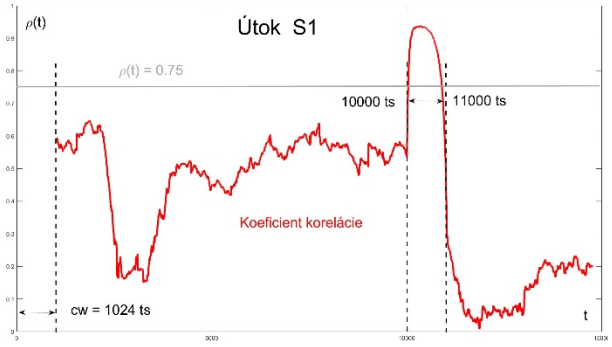
Obr. 41: Útok N2, Autoregresný koeficient



Obr. 42: Útok N2, koeficient Korelácie



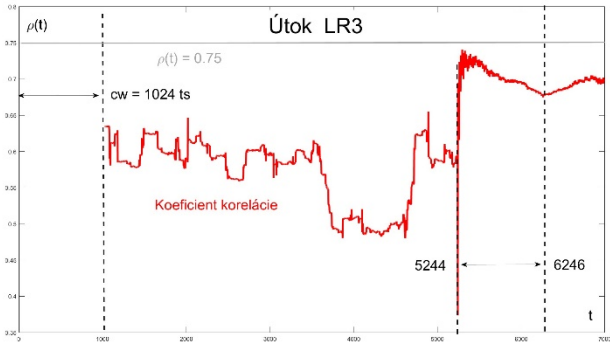
Obr. 43: Útok S1, Autoregresný koeficient



Obr. 44: Útok S1, koeficient Korelácie



Obr. 45: Útok LR3, Autoregresný koeficient



Obr. 46: Útok LR3, koeficient Korelácie

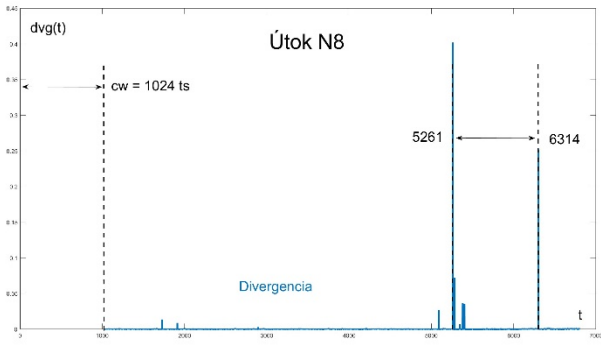
Pre väčšinu útokov sa dá predpokladať, že prekročenie detekčných hodnôt pri autoregresii a korelácii sa bude dať v kombinácii s ďalšími koeficientami využiť pre skorú jednoduchú detekciu záplavového útoku.

5.2.3 Reakcia Divergencie na DDoS útok

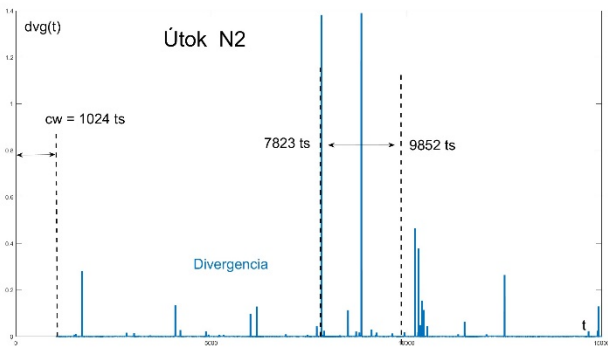
Posledným analyzovaným parametrom Kullback-Leibler Divergencia istým spôsobom porovnáva rozdelenie pravdepodobnosti P(.) prírastkov a(t) v za sebou idúcich prekrytých výpočtových oknách cw:

$$dvg(t) = \sum_{k=1}^{n} P(t-1) ln(\mathrm{P(t-1)/P(t)})$$

Výhodou Divergencie je, že na sledovaných vzorkách vo väčšine prípadoch reagovala okamžite na nástup DDoS útoku vysokým impulzom (peakom). Nevýhodou je, že mala tendenciu takto reagovať aj na peaky v bežnej prevádzke, čo by mohlo spôsobovať množstvo falošných hlásení.

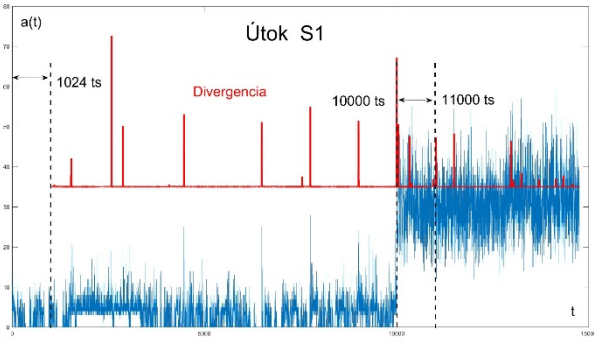


Obr. 47: Útok N8, Divergencia

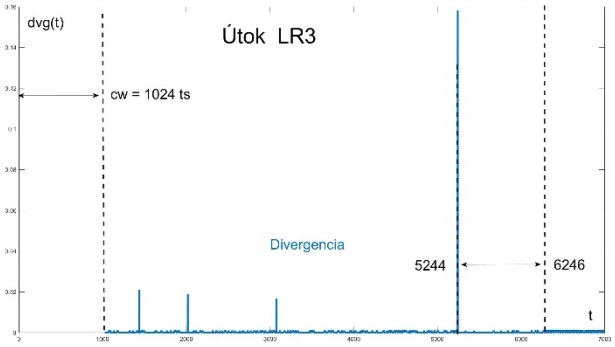


Obr. 48: Útok N2, Divergencia

Ďalšou nevýhodou použitia Divergencie je fakt, že vysoký impulz pri nástupu útoku trvá veľmi krátko oproti ostatným parametrom, preto jej aplikáciu v kombinácii s ostatnými parametrami vidíme ako problematickú. Použitie Divergencie sme preto odložili do budúcnosti.



Obr. 49: Útok S1, Divergencia

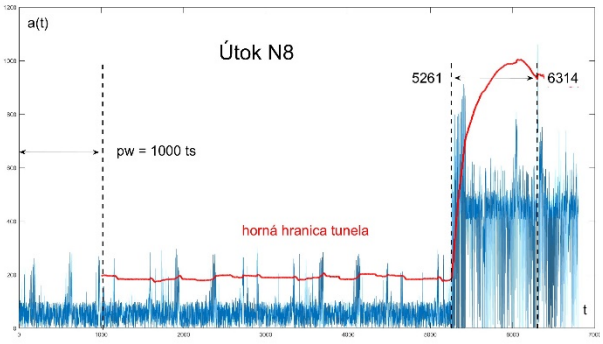


Obr. 50: Útok LR3, Divergencia

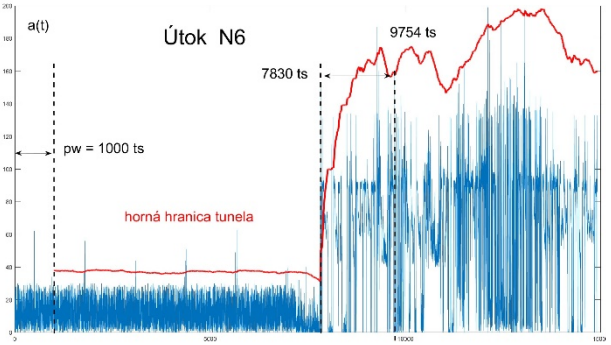
5.3 Využitie 3σ-Tunela pre rozpoznanie DDOS útokov

Na základe experimentov sme pre výpočet priemeru a odchýlky zvolili 1000 predchádzajúcich hodnôt a šírku interval sme zvolili (EX-3σ, EX+3σ) (pri menšej veľkosti intervalu dochádzalo k častým falošným hláseniam). Túto metódu sme nazvali predikčný 3σ-tunel.

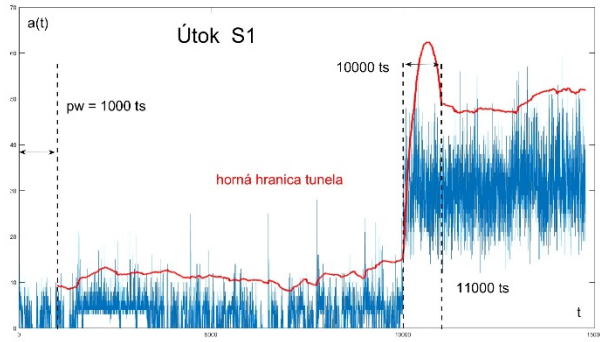
Vzniká prirodzená otázka, prečo nepoužiť predikčný tunel priamo na prírastky IP toku a(t) a nezdržiavať sa výpočtom štatistických koeficientov. Po prevedení experimentov sme zistili, že síce prekročenie hornej hranice tunela deteguje relatívne skoro nástup DDoS útoku, ale zároveň dochádza k častému hláseniu falošného poplachu. Pri zväčšení násobnosti smerodajnej odchýlky sa síce počet falošných hlásení zmenšil ale zároveň sa zmenšila schopnosť detegovať útok. Na nasledujúcich obrázkoch si ukážeme na vybraných útokoch len hornú hranicu 3σ-Tunela.



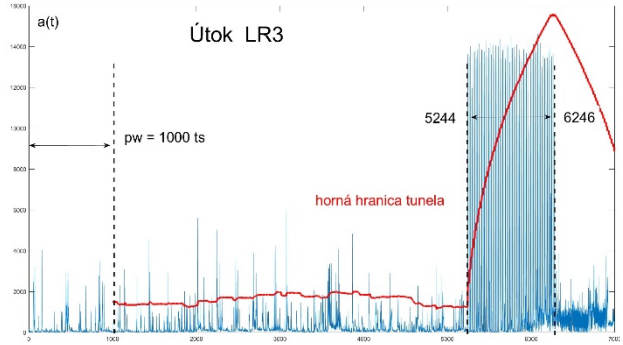
Obr. 51: Útok N8, horná hranica 3σ-Tunela



Obr. 52: Útok N6, horná hranica 3σ-Tunela

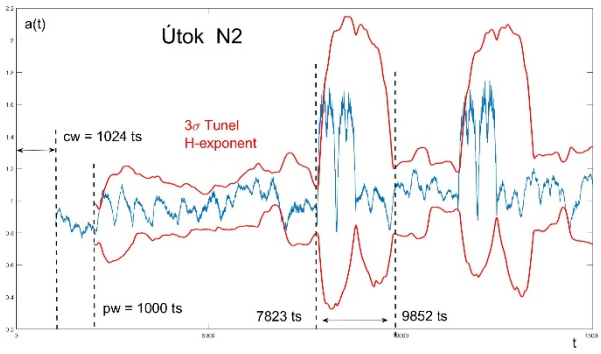


Obr. 53: Útok S1, horná hranica 3σ-Tunela

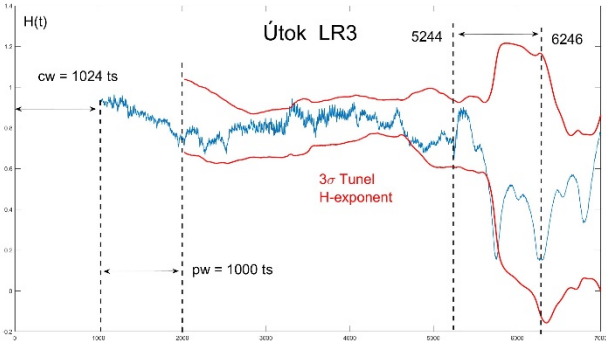


Obr. 54: Útok LR3, horná hranica 3σ-Tunela

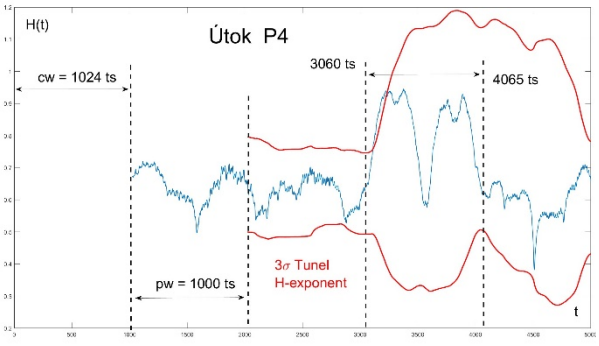
Na nasledujúcich grafoch budeme demonštrovať chovanie sa 3σ-Tunela na parametre s najväčšou variabilitou z pomedzi uvažovaných štatistických koeficientov, a to na Hurstovom exponente (odhadnutého RS štatistikou s odstránením autoregresie AR1).



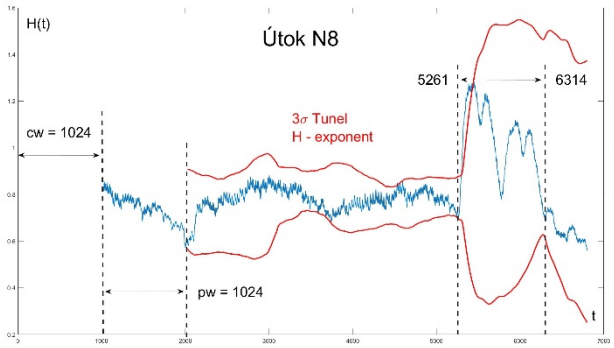
Obr. 55: N2, 3σ-Tunel pre H-exponent



Obr. 56: LR3, 3σ-Tunel pre H-exponent



Obr. 57: P4, 3σ-Tunel pre H-exponent



Obr. 58: N8, 3σ-Tunel pre H-exponent

Vo väčšine prípadoch (až na útok LR3) 3σ-Tunel skoro rozpoznal začiatok DDoS útok s minimálnym, niekedy až nulovým počtom falošných hlásení.

Ako sme už uviedli, na základe experimentov sme nastavili veľkosť predikčného okna na 1000 ts a šírku intervalu pre 1001 hodnotu na trojnásobok smerodajnej odchýlky. Pri takomto nastavení sme vo väčšine experimentov (rôzne koeficienty a rôzne útoky) získali skoré rozpoznanie začiatku útoku s minimálnym počtom falošných hlásení. Na predchádzajúcich obrázkoch sú to 3 prípady zo 4. Takto nastavený tunel má dve užitočné vlastnosti:

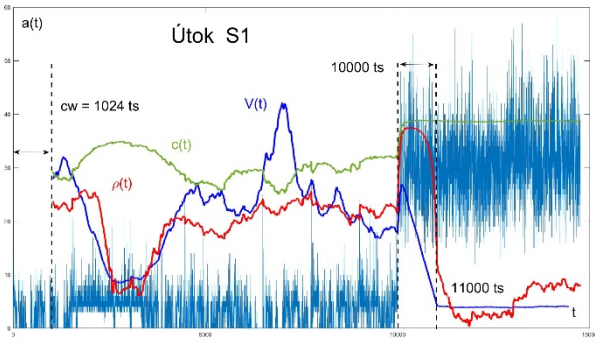
1. Má dostatočne „dlhú pamäť“ na to, aby dokázal pokryť lokálnu variabilnosť štatistického koeficientu a nespôsoboval veľké množstvo falošných hlásení
2. Má dostatočne „dlhú pamäť“ na to, aby nedokázal zareagovať na relatívne veľký nárast hodnôt skúmaného koeficient pri začiatku DDoS útoku (veľký vzhľadom na relatívnu variabilnosť)

Samozrejme aby sme mohli vysloviť nejaké relevantnejšie závery a odporúčania, je potrebné previesť množstvo experimentov na záznamov rôznych DDoS útokov.

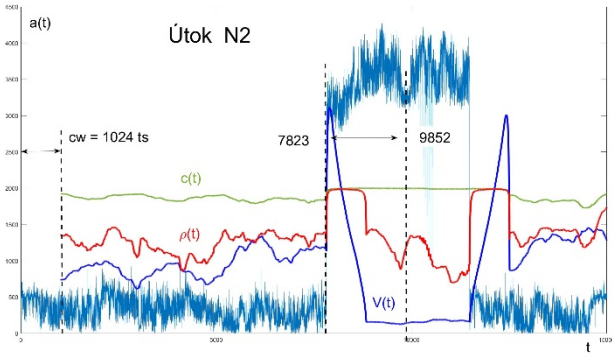
5.4 Detekčné funkcie pre strojové rozpoznanie DDoS útokov

V predchádzajúcich kapitolách sme uviedli dva spôsoby rozpoznania DDoS útokov pomocou štatistických koeficientov. Prvým spôsobom bolo určenie detekčných hodnôt pre vhodné parametre, 1,00 pre variabilnosť V(t) a H-exponent, 0,90 pre autoregresný koeficient c(t) a 0,75 pre korelačný koeficient ρ(t). Druhý spôsob pomocou predikčného tunela bol analyzovaný v predchádzajúcej kapitole.

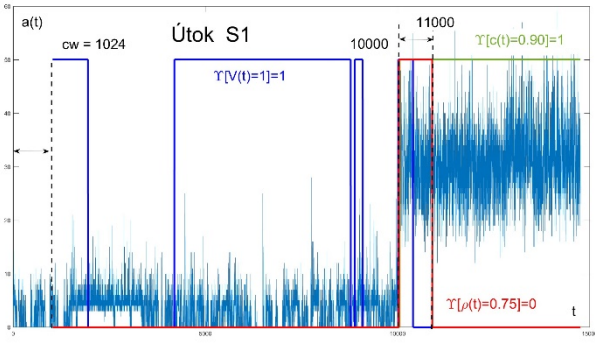
Pri využití hraničných hodnôt sme zaviedli dvojhodnotovú logickú funkciu Y[.], ktorá vyhodnocuje prekročenie hraničnej hodnoty pre daný štatistický koeficient. Nasledujúcich grafoch si ukážeme variabilnosť V(t), autoregresný koeficient c(t) a korelačný koeficient ρ(t) a priebeh ich funkcie Y[.] pre útoky S1a N2:



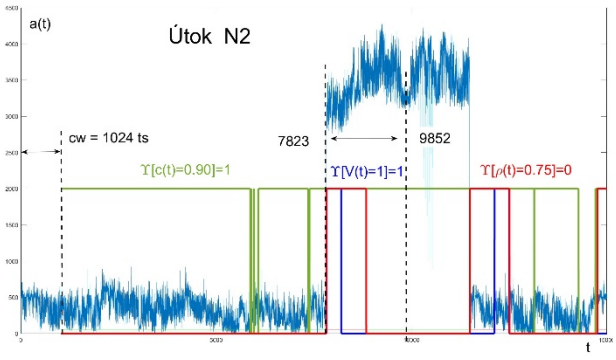
Obr. 59: S1, koeficienty V(t), c(t), ρ(t)



Obr. 60: N2, koeficienty V(t), c(t), ρ(t)



Obr. 61: S1, priebeh funkcií Y[.]

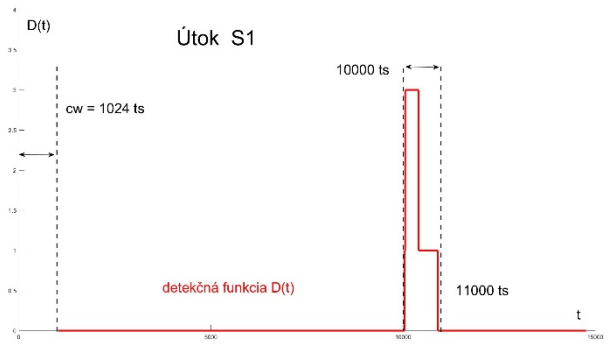


Obr. 62: N2, priebeh funkcií Y[.]

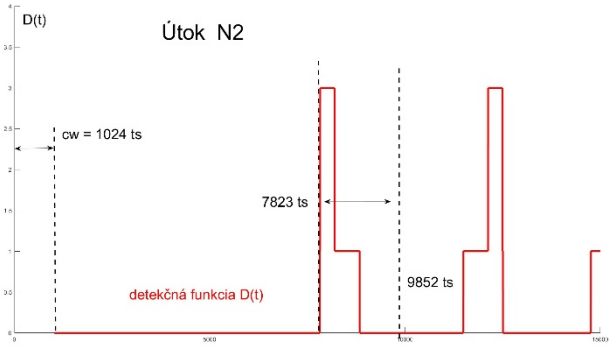
Množiny hodnôt, na ktorých koeficienty prekračujú svoje detekčné hranice, nazveme detekčné intervaly. Na obr. 61 a 62 vidíme, ako sa tieto intervaly navzájom prekrývajú. Aby sme eliminovali falošné hlásenia a zároveň dosiahli včasné rozpoznanie útokov, zavedieme pre sledované koeficienty novú detekčnú funkciu D(t), pričom pre zjednodušenie označíme napr. pre koeficient V(t) funkciu Y[V(t)>1,00] ako $Y_V(t)$:

$$D(t) = Y_V(t).Y_C(t) + Y_V(t).Y\rho(t) + Y_C(t).Y\rho(t)$$

Na nasledujúcich obrázkoch vidíme priebeh detekčnej funkcie pre útoky S1 a N2:



Obr. 63: S1, priebeh detekčnej funkcie D(t)



Obr. 64: N2, priebeh detekčnej funkcie D(t)

Vďaka tvaru funkcie D(t) sme eliminovali eventuálne falošné hlásenia a zároveň podľa hodnôt 1 alebo 3 vieme rozpoznať, a akom čase detegovali útok dva alebo všetky 3 štatistické parametre.

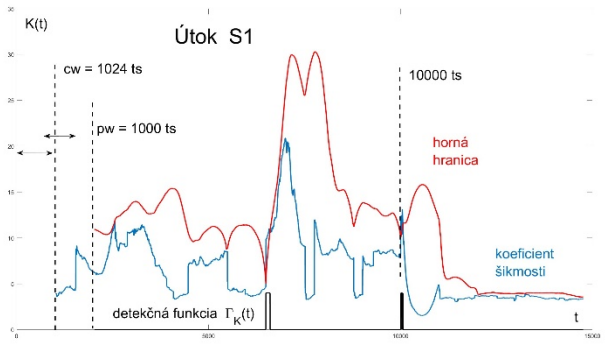
V tabuľke Tab.3 uvádzame vyhodnotenie použitia detekčnej funkcie D(t) pre analyzované útoky. Prvá hodnota predstavuje počet falošných hlásení, druhá určuje dobu, kedy detegovali útok minimálne dva koeficienty a tretia hodnota označuje dobu, kedy detegovali útok všetky tri štatistické koeficienty V(t), c(t) a ρ(t).

útok	S1	N2	LR3	S5	N6	N7	N8
D(t)	0 – 34 - 52	0 - 5 - 5	x	x	0 – 75 - 119	0 - 228 - 480	0 - 47 - 210

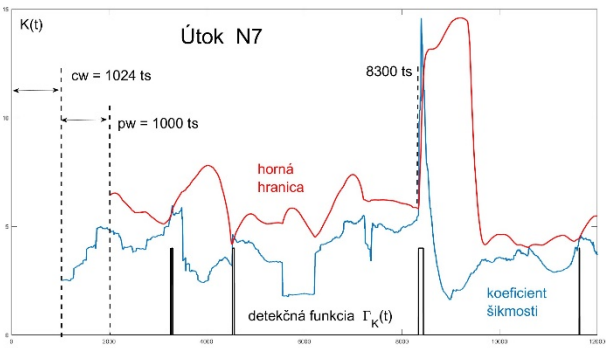
Tab.3: Vyhodnotenie rozpoznania DDoS útoku pomocou detekčnej funkcie D(t)

Rozpoznanie útoku podľa hraničnej hodnoty bolo účinné hlavne pri štandardných útokoch, pričom vo viacerých prípadoch dva koeficienty prekročili svoje hraničné hodnoty podstatne skôr než všetky tri. Pri neštandardných útokoch ako LR3, P4 a S5 žiadne dva detekčné intervaly, ak vôbec vznikli, nemali spoločný prienik, a preto útok nebol rozpoznaný

Rovnakú metodiku budeme aplikovať aj na rozpoznanie útoku pomocou predikčného tunela. V predchádzajúcej kapitole sme určili, pri zohľadnení doby rozpoznania útoku a počtu falošných hlásení, prvé tri najefektívnejšie štatistické koeficienty: šikmosť K(t), špicatosť Skw(t) a variabilnosť V(t). Keďže sme vybrali koeficienty, ktorým sa pri nástupe DDoS útoku výrazne zväčšia hodnoty, preto sa budeme zaoberať len hornou hranicou predikčného tunela. Dobu prekročenia hornej hranice vyhodnotíme ako detekčný interval a popíšeme ho dvojhodnotovou funkciou Γ[.], pričom napr. pre koeficient špicatosti označíme $\Gamma_K(t) = \Gamma[K(t)>Hr(t)]$, kde Hr(.) je horná hranica predikčného intervalu pre koeficient K(.).



Obr. 65: S1, priebeh funkcie Γ(t)



Obr. 66: N7, priebeh funkcie Γ(t)

Podobne ako pri predchádzajúcej metóde, pri rozpoznávaní pomocou hraničnej hodnoty, aj teraz zavedieme detekčnú funkciu, ktorá bude kombináciou identifikátorov Γ(t):

$$D_P(t) = \Gamma_K(t).\Gamma_{Skw}(t) + \Gamma_K(t).\Gamma_V(t) + \Gamma_{Skw}(t).\Gamma_V(t)$$

V tabuľke 4 uvádzame vyhodnotenie použitia detekčnej funkcie $D_P(t)$.

útok	S1	N2	LR3	S5	N6	N7	N8
$D_P(t)$	1 – 6 - 28	0 - 1 - 1	1 - 1 - 1	1 - 18 - 41	1 - 12 - 12	1 - 39 - 49	0 - 1 - 1

Tab.4: Vyhodnotenie rozpoznania DDoS útoku pomocou detekčnej funkcie D(t)

5.5 Záver

Na základe doteraz prevedených experimentoch vidíme využitie vybraných štatistických koeficientov má podstatne väčší potenciál pri rýchlom rozpoznaní DDoS útokov (*FAD – Fast Attack Detection*) než štandardné sledovanie priemernej a špičkovej intenzity. Pri DDoS útoku je silný predpoklad, že sa tieto intenzity výrazne zvýšia. Avšak priemerná intenzita rastie len lineárne v závislosti od veľkosti kĺzavého okna, čo vedie k neskorej detekcii útoku, a rozpoznávanie útoku pomocou nastavenia nejakého tresholdu pre špičkovú rýchlosť môže viesť k častým falošným hláseniam vďaka výskytu peakov v IP toku.

Pri prechode štandardnej IP prevádzky do DDoS útoku sa výrazne zmení nie len priemerná a špičková intenzita, a zmení sa celková pravdepodobnostná štruktúra záplavového toku. Na zachytenie tejto zmeny sme využili prezentované štatistické koeficienty. Tie môžeme rozdeliť na dva typy, koeficienty, ktoré pri nástupu útoku prekročili nejakú detekčnú hodnotu a koeficienty, ktoré na útok reagovali výraznou zmenou svojich hodnôt resp. impulzom. Niektoré koeficienty dokonca patria do obidvoch skupín. Na základe tohto delenia sme vytvorili:

- Metódu použitia detekčných hraníc (MDH)
- Metóda použitia predikčných tunelov (MPT)

Porovnaním tabuliek 3 a 4 vidíme, že MPT rozpoznala každý uvažovaný typ útoku a to podstatne skôr ako MDH, avšak pri väčšine útokov signalizovala jedno falošné hlásenie. Predpokladáme, že tento problém by sa mohol odstrániť viacerými spôsobmi, napríklad

pridaním ďalšieho vhodného detekčného koeficientu, ale nejakým typom vyhladenia hodnôt koeficientu pred použitím predikčného tunela, napr. Exponenciálnym alebo Holtovým vyhladením.

PRÍLOHY

Súbor: V7 Príloha (KPB3) – Analýza dostupných datasetov.docx