Univerzita Pardubice
Fakulta Ekonomicko-Správní

Modelování algoritmů – generování rozhodovacích pravidel pomocí rough množin

Vladimír Dvořák

Diplomová práce
2009

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.
Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.
V Pardubicích dne 17. 8. 2009

Vladimír Dvořák

Poděkování:
Velice bych rád touto cestou poděkoval vedoucímu své diplomové práce doc. Ing. Jiřímu Křupkovi, Ph.D., za zájem, připomínky a čas, který mi věnoval.
Též bych rád poděkoval celé své rodině za podporu, která mi během studií velmi pomáhala.

ANOTACE
Práce se zabývá odhalováním komunikačních anomálií v datových sítích s využitím aparátu teorie hrubých množin. V práci je navržen paketově orientovaný způsob diskriminace mezi normálním a anomálním síťovým provozem. Dále byla vytvořena soustava skriptů, která umožňuje sestavit důvěryhodnostní model síťového chování a testovat tento model.

KLÍČOVÁ SLOVA
hrubé množiny; klasifikace; anomálie; bezpečnost počítačových sítí

TITLE
Algorithm modeling – generation of decision rules using rough sets

ANNOTATION
The thesis deals with discovering comunication anomalies in data networks using Rough Set Theory aparat. There is designed packet-oriented kind of discrimination between normal and anomal network traffic. Also was developed complex of scripts which allow creation of credible model of network behavior and allow to test this model.

KEYWORDS
rough sets; clasification; anomalies; computer network security

Obsah

1 Úvod

Bezpečnost počítačových sítí je v posledních letech stále více diskutována a trend se zdá být neklesající. Důvodem jsou nejen sofistikovanější způsoby porušení bezpečnostních pravidel, ale také všeobecně zvyšující se počítačová gramotnost populace. Programové vybavení tak je trvale vystavováno tlaku ze strany nově objevovaných bezpečnostních chyb. Vývoj bezpečnostních mechanismů jako jsou stále důmyslnější vícevrstvé firewally s podporou detekce průniku a zkvalitňování bezpečnostní politiky ale nedokáží příliš dobře odhalovat nové a nepopsané formy útoků. Důvodem je, že vetšinou jsou navrženy tak, že porovnávají vzorky síťového provozu se signaturami ve své databázi. Velice rychle a efektivně takové systémy dokáží odhalit známé typy útoků. Na toho času neznáme typy útoků jsou ale bezbrané. Proto se v poslední době rozmáhají snahy integrovat do systémů detekce průniku oblast umělé a výpočetní inteligence. Statistické metody totiž na tomto poli selhávají.

Cílem práce je navrhnout takový bezpečnostní model, který bude pro stávající bezpečnostní mechanismy zajišťovat podporu před nově vzniklými komunikačními změnami, které potenciálně mohou nést škodlivý síťový provoz.

Návrh bude využívat nástrojů otevřeného (opensource) software a jako referenční operační systém použit GNU Debian[1].

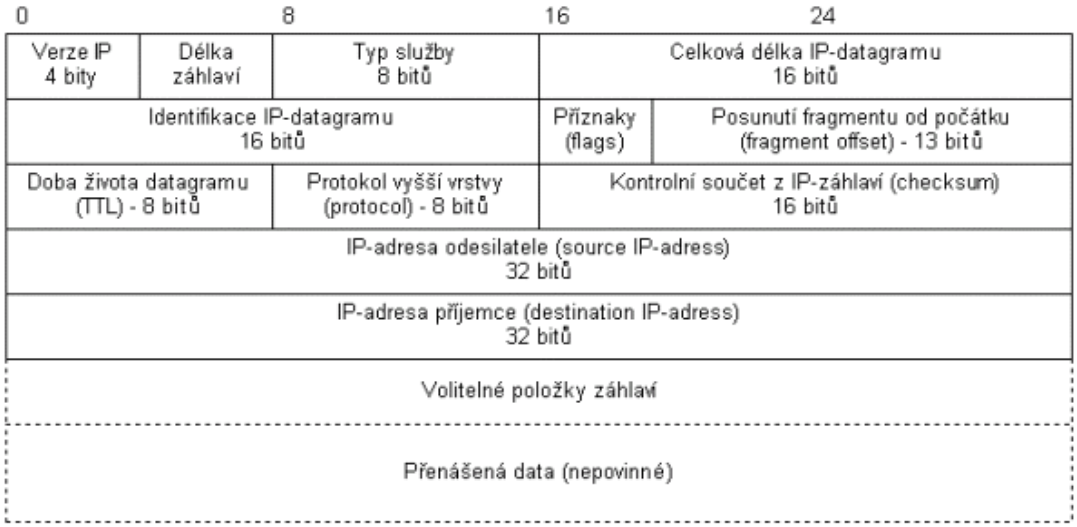
1 Dostupný z http://www.debian.org/

2 Architektura TCP/IP a problematika IDS

Rodina protokolů TCP/IP (Transport Control Protocol/Internet Protocol) je v současné době nejrozšířenější architekturou využívanou v Internetu pro komunikaci mezi vzdálenými počítači. Z tohoto důvodu se v této práci budeme zabývat analýzou právě těchto protokolů. V rámci pojetí způsobu klasifikace síťového provozu, paketově orientovanému, je třeba se věnovat hodnocení vlivu jednotlivých parametrů v záhlaví transportních protokolů architektury TCP/IP. Zde se mají na mysli protokoly TCP, UDP a ICMP (další transportní protokoly vzhledem k méně častému využívání na koncových stanicích neuvažujeme).

Protokol IP

Protokol IP (Internet Protocol), definice v RFC 791[2], přenáší tzv. IP datagramy mezi vzdálenými počítači. Každý IP datagram ve svém záhlaví nese adresu příjemce, což je úplná směrovací informace pro dopravu IP datagramu k adresátovi [1]. Následující obrázek znázorňuje záhlaví protokolu IP (dle [2]):



Obrázek 1: Záhlaví protokolu IPv4 [Zdroj: [2]]

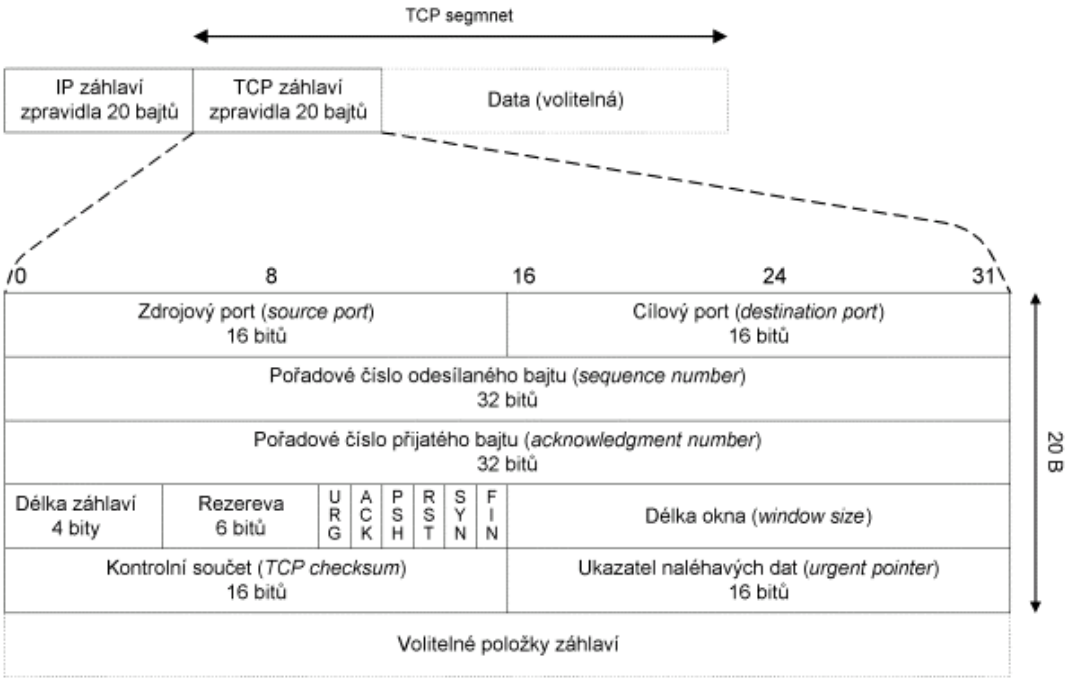
Protokol IP v současné době existuje ve dvou verzích, IPv4 a IPv6. V této práci se nehodnotí protokoly síťové vrstvy jako diskriminátory kvality síťového protozu, proto nepokládáme za důležité dělat rozdíl mezi verzí 4 a verzí 6. Pokud bychom uvažovali tokově orientovanou analýzu, pak rozdíl v návrhu těchto verzí mohl hrát důležitou roli. Rozdíly mezi verzemi pak je možné najít v literatuře, např. [3].

Význam pro analýzu paketově-orientovaného aparátu **je minimální**.

2 http://tools.ietf.org/html/rfc791

Protokol TCP

Protokol TCP (Transmission Control Protocol), definován v RFC 793[3], je transportním protokolem, zajišťuje spojově orientovanou přenosovou službu. Záhlaví protokolu TCP [4]:



Obrázek 2: Záhlaví protokolu TCP [Zdroj: [4]]

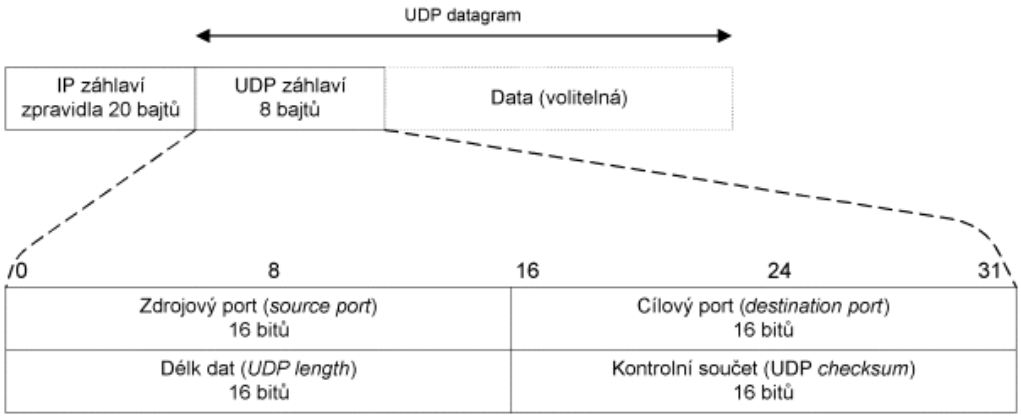
Vzhledem k tomu, že mnoho druhů síťového provozu používá právě protokol TCP pro přenos dat, bude analýza záhlaví jednotlivých TCP paketů hrát klíčovou roli v klasifikační analýze pomocí nástroje teorie hrubých množin.

Význam pro analýzu je zásadní. Vzhledem k tomu, že se jedná o spojovaně-orientovaný protokol, nabízí celou řadu možností jak s tímto nakládat. Můžeme sledovat počet pokusů o navázání spojení, počty odmítnutých spojení, ukončených spojení.

Protokol UDP

Protokol UDP, definován v RFC 768[4], je jednoduchou alternativou k protokolu TCP. Protokol UDP je nespojovaná služba (narozdíl od protokolu TCP), tj. nenavazuje spojení. Odesilatel odešle UDP datagram příjemci a už se nestará o to, zda-li se datagram náhodou neztratil (o to se musí postarat aplikační protokol). UDP datagramy jsou baleny do IP-datagramu. Záhlaví protokolu UDP [5]:

3 http://tools.ietf.org/html/rfc793
4 http://tools.ietf.org/html/rfc768

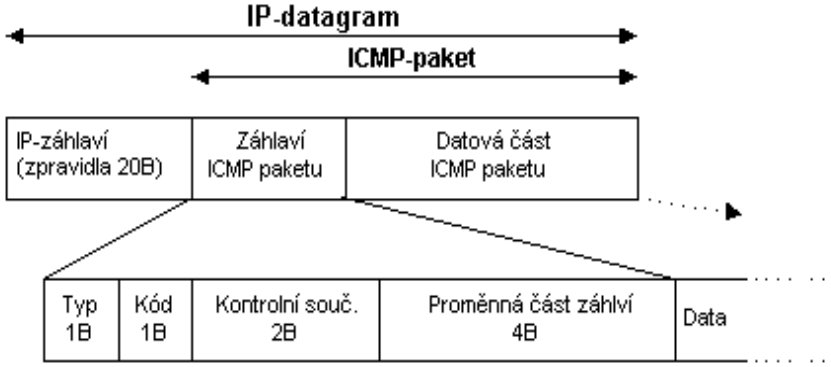


Obrázek 3: Záhlaví protokolu UDP [Zdroj: [5]]

Využití při analýze je potenciálně možné, ale v rámci našeho schématu není podporováno. Důvodem je neschopnost rozlišit stav UDP „spojení“ a tak jej z analýzy vylučujeme. Představa zapojení UDP parametrů by byla vhodná spíše u tokově-orientovaného klasifikačního schématu, kdy lze na základě párování (zdrojový port, cílový port) v kombinaci s adresou příjemce, resp. odesilatele z IP záhlaví determinovat alespoň přibližný stav komunikace. Možnost využitkovat UDP v rámci paketově-orientované analýzy je spíše spekulativní, kdy bychom mohli např. sumarizovat počty jednosměrných UDP paketů k jednomu cílovému portu. Tato síťová situace by indikovala snahu komunikovat s nedostupnou službou. Potíž ale spatřujeme v tom, že implementace UDP retransmitů je závislá na aplikaci, která určuje periodu a frekvenci při opakovaném zasílání UDP zpráv, ta může být zcela různá.

Protokol ICMP

Protokol ICMP (Internet Control Message Protocol), definován v RFC 792[5], patří mezi služební protokoly užívané v rámci architektury TCP/IP. Používají ho operační systémy počítačů v síti pro odesílání chybových zpráv, například pro oznámení, že požadovaná služba není dostupná nebo že potřebný počítač nebo router není dosažitelný [6]. ICMP záhlaví vypadá následovně:



Obrázek 4: Záhlaví ICMP protokolu [Zdroj: [2]]

5 http://tools.ietf.org/html/rfc792

První byte v ICMP záhlaví reprezentuje druh ICMP zprávy (Typ 3 – nedoručitelný IP datagram, Typ 4 – sniž rychlost odesílání, Typ 5 – změň směrování, apod.). Druhý byte, kód, se váže k typu zprávy. Například Typ 3 vypovídá o nedoručitelnosti IP datagramu příjemci. O důvodech informuje kód a ten může nabývat těchto hodnot [7]:

Typ 3	
Číslo kódu	**Textový popis** (překlad z angl.)
0	net unreachable (nedostupná síť)
1	host unreachable (nedostupný hostitel)
2	protocol unreachable (nedostupný protokol)
3	port unreachable (nedostupný port)
4	fragmentation needed and DF set (nutnost fragmentace)
5	source route failed (selhání směrování podle zdroje)

Tabulka 1: Chybové kódy protokolu ICMP pro nedoručitelné IP datagramy

Tabulka od dob standardizace protokolu ICMP byla výrazně rozšířířena o další kódové informace. Tyto jsou specifikovány v novějších RFC dokumentech. Seznam rozšiřujících kódů společně, které standardy je popisují, lze nalézt např. zde [8].Další typy, resp. kódy ICMP zpráv zde nebudeme uvádět, lze je možné dohledat např. [tamtéž].

Význam pro analýzu je spíše **podpůrného** charakteru, ale nepochybně výsledky měření počtu ICMP zpráv (obzvlášť negativních) se výrazně podílí na celkovém informačním přínosu co se týká stavu sítě. Příkladem může být situace, kdy se během krátké doby objeví na síťovém rozhraní několik ICMP zpráv kódu Port Unreachable. Toto může indikovat např. možnost skenování portů (viz dále) nebo jinou síťovou anomálii. Proto hraje analýza ICMP protokolu (záhlaví) velkou roli při diskriminaci mezi normálním či anomálním síťovým provozem.

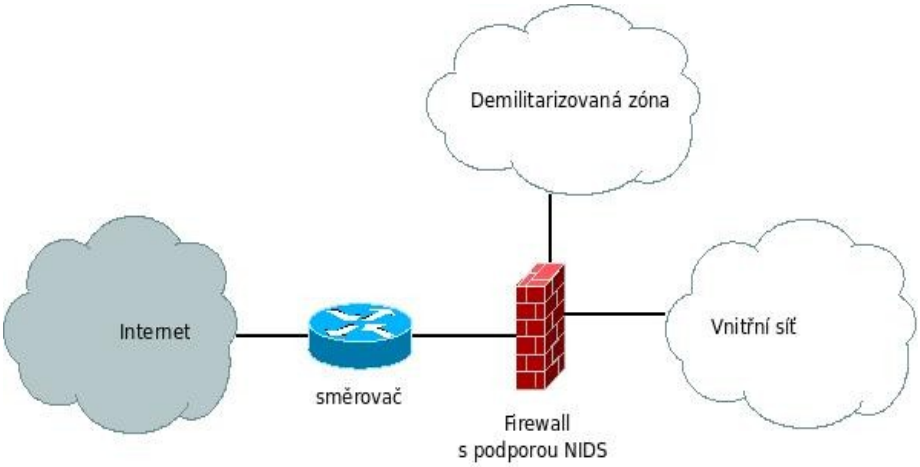
HIDS a NIDS systémy

Systémy IDS (Intrusion Detection Systems) bývají implementovány jako součást bezpečnostního zajištění IS[6] (informačního systému). Cílem systémů IDS je detekovat průnik anebo odhalit jen samotný pokus o průnik do IS. Systémy IDS obecně slouží jako detektory rozpoznávající napadaní či pokusy o napadení koncových stanic. Rozlišují se na (dle [9]):

- **hostitelské (HIDS - Host Intrusion Detection System)**, jde v podstatě o specializovaný program instalovaný přímo na hlídané koncové stanici, který monitoruje a vyhodnocuje lokální aktivity,
- **síťové (NIDS – Network Intrusion Detection System)**, samostatné zařízení či program, které monitoruje na něj svedený síťový provoz, z následné analýzy provozu poté usuzuje na možné problémy a incidenty u jiných koncových stanic.

6 Informační systém z hlediska bezpečnosti definujeme jako soubor následujících komponent: data, software, hardware, lidé a procesy. Tyto komponenty se pak nazývají aktiva a jsou to oblasti, které chráníme.

Síťový (NIDS) pak lze znázornit následujícím způsobem:



Obrázek 5: Příklad integrace NIDS systému do bezpečnosti sítě [Zdroj: vlastní]

Spolupodílí se tedy obecně na podpoře zajištění následujících principů:

- **důvěrnost**, k objektům IS mají přístup pouze autorizované subjekty,
- **integrita**, objekty IS mohou modifikovat pouze autorizované subjekty,
- **autenticita**, ověření identity, pravosti subjektů,
- **dostupnost**, objekty IS jsou autorizovaným subjektům dostupné do určité garantované doby, nedojde tedy k odmítnutí služby, kdy subjekt nedostane to, na co má právo.

V některých zdrojích [10] se též objevuje ještě jeden bezpečnostní princip – **nepopiratelnost**. Je charakterizován takto: zajišťuje, aby daný subjekt nemohl později popřít to, co předtím vykonal. Je-li zajištěna nepopiratelnost, pak v případě sporu dvou stran může nezávislá třetí strana rozhodnout, zda daný úkon proběhl nebo ne. Dále je pojem nepopiratelnost pojen se zodpovědností, tedy nepopiratelnost zodpovědnosti [11]: ochrana zajišťující možnost prokázat druhé straně odeslání/příjem zprávy, tzn. znemožnit jí popření odeslání/příjmu zprávy. Vzhledem k vágnímu pojmu „prokazatelnost" si dovoluji tento princip vynechat neb jej pokládáme za vysvětlitelný pouze důkazním řízením v rámci práva. Stejný názor sdílím u pojmu „nepopiratelnost". Zdůvodněním našeho přístupu je následující úvaha: Je možné na úrovni kryptografických mechanismů prokázat, že digitálně podepsaný dokument soukromým klíčem je skutečně podepsán jeho vlastníkem, nebyl tedy tento klíč zcizen a dokument tak podepsán někým jiným? Matematickým aparátem jsme schopni tedy zajistit výše uvedené principy, ale tento je prozatím dle našeho názoru na úrovni právní vědy.

Síťové anomálie

Pod pojmem síťová anomálie si lze představit jakýkoliv typ provozu, který je svým způsobem nežádaný, který by se neměl na síti objevovat. Mezi síťové anomálie lze zařadit vše, co se nějakým způsobem odlišuje od běžného provozu na síti. Jakýkoliv typ útoku vedené po počítačové síti (Internetu) lze chápat jako anomálii. Příkladem mohou být útoky

typu DoS nebo DDoS, nebo tzv. skeny.

Typy síťových anomálií (dle [12]):

- **Anomálie síťového provozu** (Network Operation Anomalies), anomálie způsobené např. změnou konfigurace síťových zařízení (přidáním nových zařízení nebo změny limitů).
- **Flash Crowd anomálie,** do této kategorie se zařazují takové síťové anomálie typicky vzniklé v důsledku vypuštění nového softwaru nebo extrémním zájmem o webové stránky v důsledku publicity.
- **Anomálie v důsledku zneužití sítě** (Network Abuse Anomalies), dva typy zneužití sítě mohou být identifikovány za použití toků (flows): DoS útoky a port skeny. Toto síťové chování je velice četné.

Detekce síťových anomálií (dle [13]):

- **porovnávání signatur,** zde hledáme konkrétní typy anomálií. V tomto případě používáme techniku Pattern Matching, kde porovnáváme otisk známeho útoku, který máme v databázy, s provozem na síti. Tato technika je velice úspěšná na známé útoky, ale nedokáže odhalit typy nové.
- **stavová analýza protokolu**, komunikační protokoly, používaní v Internetu, jsou ve valné většině specifikované v dokumentech RFC. Všechny odchylky od těchto standardů pak mohou být klasifikovány jako anomální síťové jednání.
- **behaviorální analýza**, též označovaná NBAD (Network Behavioral Analysis and Detection) začala být populární teprve až v posledních letech (patrně s rozmachem disciplín Data Miningu, strojového učení, dalších metod umělé inteligence). Tato analýza spočívá v tom, že NBAD systém získá model síťového chování daného prostředí (tuto fázi můžeme nazvat učením), tento model prohlásí za „správný". Následuje fáze síťového pozorování. Jakmile detekuje odchylku, reaguje dle definovaných pravidel (zašle výstrahu emailem, sms, apod.). Tato analýza se vyznačuje potenciálně možným výskytem velkého počtu planých poplachů (false positive alerts).

IDS systémy jsou často implementovány s dalšími bezpečnostními nástroji [14] :

- Systémy pro analýzu (hodnocení) zranitelnosti **(vulnerability assessment systems)** - ověřují odolnost na známé útoky, generují "snímky" bezpečnostních stavů systému v určitém čase pro možnost zjištění problémů v důsledku lidských chyb nebo pro potřeby auditu.
- Systémy pro kontrolu integrity souborů **(integrity checkers systems)** - používají kryptografické kontrolní sumy pro kontrolu kritických datových objektů. **Integrity checkers** mohou preventivně ochránit systém před útokem (příprava útočníka často spočívá v předběžné modifikaci určitých systémových souborů).
- "Hrnce s medem" **(Honey Pot)** - Systém Honey Pot odláká útočníka od kritických objektů na snadno dostupný terč a během jeho aktivit sbírá o něm informace do doby, než je možno vygenerovat odezvu.
- "Vycpané cely" **(Padded Cell)** - Systém Padded Cell izoluje útočníka do simulovaného prostředí, kde sice může postupovat podle svého plánu, ale bez reálně poškozujících výsledků.

Implementací systémů IDS existuje celá řada i v oblasti otevřeného software (opensource). Seznam známých a používaných opensource IDS systémů [15]:

- Snort (Everyone's favorite open source IDS),
- Ossec Hids (An Open Source Host-based Intrusion Detection System),
- Fragroute/Fragrouter (A network intrusion detection evasion toolkit),
- Base (The Basic Analysis and Security Engine),
- Squil (The Analyst Console for Network Security Monitoring).

Snort

Tento software patří mezi velice používané nástroje v oblasti HIDS a NIDS. Program Snort zmiňujeme z důvodu relativně snadné integrace dodatečných softwarových doplňků (pluginů), jeho architektura je dostatečně modulární. Integrace podsystému užívající klasifikačních schopností matematického aparátu teorie hrubých množin je tedy reálná.

Snort může pracovat v několika režimech [16]:

- **Sniffer mode**, v tomto režimu Snort naslouchá na síťovém rozhraní a nasbírané datagramy zobrazuje v reálném čase na konzoli. V tomto režimu pracuje Snort jako obyčejný "sniffer", podobně tedy jako program *tcpdump* (viz dále).
- **Packet Logger Mode**, principielně se podobá Sniffer módu, rozdíl je pouze v cíli výstupu - v tomto případě se jedná o výstup do souboru.
- **Network Intrusion Detection System mode**, zde se jedná o komplexní přístup k NIDS; provádí analýzu síťového provozu a porovnává nasbíraná síťová data s uživatelem definovanými pravidly. Na základě těchto pravidel se rozhoduje o odezvě – zda zahlásit upozornění o detekovaném útoku (alerting) nebo *ad hoc* sestavit blokovací pravidlo a tento typ provozu nepropustit dále do sítě.
- **Inline mode**, zde Snort získává datagramy z iptables (namísto z libpcap[7]) a na základě definovaných pravidel se rozhoduje, zda daný datagram zahodí nebo propustí dále do sítě.

7 Stručné pojednání o knihovně libpcap v kapitole 5

3 Analýza síťového provozu

3.1 Protokoly pro analýzu síťového provozu

Protokoly, které se využívají pro analýzu síťového provozu, jsou většinou založené na analýze toků (anglicky flow nebo IP traffic flow). Definovat tok není až tak jednoduché, lze nalézt velké množství variant těchto definic. Důvod možno spatřovat v rozdílnosti návrhu jednotlivých protokolů, kde se na tok nahlíží různým způsobem. V této práci se budeme držet definice toku tak, jak je definován v [17]: **síťový tok** je identifikován jako **jednosměrný proud paketů mezi daným zdrojovým a cílovým počítačem** – určen IP adresou síťové vrstvy a portem definovaným na vrstvě transportní. Tok je identifikován kombinacemi následujících polí:

- source IP address (zdrojová IP adresa),
- destination IP address (cílová IP adresa),
- source port number (zdrojový port),
- destination port number (cílový port),
- protocol type (typ protokolu),
- type of service (typ služby),
- input interface (vstupní rozhraní).

V dokumentu RFC 5470 je tok definován jako [18] množina IP datagramů procházející observačním bodem v síti po určitý časový interval. Všechny pakety patřící do jednotlivého toku má množinu společných vlastností. V tomto RFC [tamtéž] je též specifikováno, že tok je z tohoto časového intervalu extrahován funkcí, která na základě informací ze záhlaví síťové (IP adresa odesilatele a IP adresa cíle) a transportní (cílový a zdrojový port) vrstvy sestavuje jednotlivé toky. Více informací lze nalézt v zmíněném RFC.

V rámci analýzy toků je též diskutována bezpečnost počítačových sítích. Otázky, zda je možné identifikovat bezpečnostní incident na úrovni těchto monitorovacích protokolů jsou předmětem výzkumu. Výzkum se snaží odpovědět na otázky integrace monitorovacích nástrojů do systémů typu IDS, resp. NIDS.

sFlow protokol

sFlow je standard pro monitoring počítačových sítí, tedy technologie pro sledování síťového provozu v datových sítích. sFlow systém se skládá z sFlow Agenta (integrovaný software ve switchi, routeru nebo nainstalovaná služba na serveru) a centrálního sběrného systému - kolektoru (sFlow Collector). Sflow monitorovací systém byl navržen tak, aby poskytoval kontinuální monitoring síťového provozu. Pro komunikaci mezi agenty a kolektorem využívá vlastního formátu předávání zpráv [20]. sFlow, narozdíl od tokových protokolů NetFlow a IPFIX, je vzorkovacím[8] protokolem (packet sampling protocol) [21]. Kombinaci statistik síťového rozhraní (interface counters) a síťových vzorků (flow samples)

8 Vzorkovacé techniky při popisu síťového provozu jsou široce používané. Využívá se zde náhodilosti ve vzorkování, aby se předešlo periodicky opakujícím se vzorům v síťovém provozu. Více zde [19].

balí do sFlow datagramu a ten pak zasílá sFlow kolektoru [22].

Více informací o sFlow protokolu lze najít v RFC 3176. Toto RFC je k času psaní této práce aktuální. Existuje též textový soubor (memo), který popisuje nadcházející sFlow protokol verze 5. Dokument je volně přístupný na stránkách http://www.sflow.org/.

NetFlow protokol

NetFlow je otevřený protokol vyvinutý společností Cisco Systems, určený původně jako doplňková služba k Cisco směrovačům. Jeho hlavním účelem je monitorování síťového provozu na základě IP toků, které poskytuje administrátorům i manažerům podrobný pohled do provozu na jejich síti v reálném čase. Proto tvoří důležitou a nepostradatelnou součást zabezpečení každé počítačové sítě a je užitečný pro ISP[9], kteří na základě NetFlow statistik mohou svým zákazníkům účtovat ceny služeb v závislosti na množství přenesených dat. S pomocí NetFlow statistik lze odhalovat vnější i vnitřní incidenty, úzká místa v síti, dominantní zdroje provozu, efektivněji plánovat budoucí rozvoj sítě, sledovat, kdo komunikoval s kým, jak dlouho a s pomocí kterého protokolu. [23] S protokolem NetFlow, podobně jako s sFlow, lze pracovat též nástroji otevřeného software. Např. *softflowd*, *pmacct*, *fprobed* a další. NetFlow verze 9 je definován v [24].

IPFIX protokol

Potřeba standardizovaného a přitom univerzálního protokolu pro sledování toku dat na IP (Internet Protocol) sítích dala vzniknout protokolu IPFIX (Internet Protocol Flow Information Export). Vyvinula jej stejnojmenná pracovní skupina v rámci organizace IETF (Internet Engineering Task Force[10]). IPFIX je inspirován a postaven na základu protokolu NetFlow (viz výše) firmy Cisco. IPFIX využívá transportního protokolu SCTP (Stream Control Transmission Protocol). Transportní protokoly pracují na čtvrté vrstvě ISO/OSI referenčního modelu, SCTP protokol tedy je na stejné úrovni jako TCP, UDP nebo ICMP. Z těchto důvodů je tedy třeba mít zakomponovanou podporu SCTP protokolu v rámci jádra operačního systému a ostatní zařízení, ze kterých získáváme údaje musí podporovat též tento protokol. V zásadě je možné vyslovit, že IPFIX je *de facto* NetFlow V9 protokol nad SCTP transportním protokolem (NetFlow v9 over SCTP).

3.2 Nástroje pro sběr dat ze sítě

Tcpdump

Pro získání dat ze sítě lze využít některého ze známých nástrojů pro odposlech síťového provozu. Pro účely získání datagramů protékající monitorovaným místem lze využít program *tcpdump*, který je součástí většiny linuxových či unixových distribucí. Tento program slouží jako analyzátor datového provozu na úrovni jednotlivých síťových datagramů. Z tohoto důvodu není vhodné jej použít pro sledování toků. Pro monitoring

9 ISP – Internet Service Provider – poskytovatel připojení k Internetu

10 Internet Engineering Task Force je mezinárodní organizace, který se podílí na vývoji otevřených standardů Internet. IETF je členěna na tzv. working groups (pracovní skupiny), které se skládají s odborníků dané oblasti.

a analýzu toků lze doporučit některý z dále diskutovaných programů, které často fungují jako nadstavba programu *tcpdump*.

Parametry programu tcpdump byly zvoleny účelově pro konkrétní algoritmus předzpracování těchto dat:

```
root@server:~#tcpdump -n -ttt -vvv -p -l -S -s 0 ip | tee /tmp/data
```

Parametry:

- **-tt**, vypíše časovou značku každého získaného datagramu (přesnost v mikrosekundách, získáváno na jádře 2.6.27)
- **-ttt**, vypíše časový rozdíl (v mikrosekundách) mezi současným a předchozím řádkem výstupu, resp. mezi současným a předchozím datovým datagramem
- **-vvv**, vyšší detailnost výstupu.
- **-p**, neuvádět zařízení (síťový adaptér) do promiskuitního režimu[11].
- **-s**, v tomto kontextu, "snaplen" direktiva, která specifikuje maximální počet bytů k sejmutí v každém příchozím datagramu, hraje klíčovou roli v klasifikaci. Některé protokoly (zejména textově orientované jako např. RTSP, SIP, atd.) využívají proměnlivou délku hlaviček. Dávají tak větší šanci lépe identifikovat nově sestavovaný datový tok. Získávat zbytečně velkou část datagramu s sebou nese své nevýhody, které spočívají ve vyšších systémových nárocích. Př. pro textově orientované protokoly lze doporučit hodnotu 500-750 bytů.
- **-c**, počet datagramů, které tcpdump zachytí a skončí.
- **-w**, Tento parametr uloží nasbíraná data do souboru, který je pak možné analyzovat bud přímo samotným programem tcpdump anebo některým z analytických nástrojů zmíněných v kapitole 3.3)
- **ip**, tcpdump bude sledovat pouze protokol IP (ostatní protokoly síťové vrstvy budou eliminovány).

Ve skriptech, které využíváme pro sběr a analýzu dat, se program *tcpdump* vyskytuje dosti často. Z tohoto důvodu přikládáme další příklady, které jsou v různých obměnách ve skriptech použity. Následující příklad zobrazí všechny pakety typu TCP/SYN (pokusy o navázání spojení protokolem TCP):

```
tcpdump -n 'tcp[13] & 2 !=0'
```

V dalším případě vypíše všechna odmítnutí spojení. Jsou to takové případy, kdy klientská stanice vyšle požadavek na spojení se serverem, na jehož cílovém portu nenaslouchá žádná služba. Server pak odpoví klientské stanici paketem, který má v hlavičce TCP aktivní RST bit (reset). Je nutné poznamenat, že v TCP hlavičce je též aktivní ACK bit.

```
tcpdump -r data -n 'tcp[13] == 20'
```

Následující příklad zobrazí všechny ICMP zprávy typu Port Unreachable. Tyto zprávy zasílá server na pokusy o komunikaci s portem, na kterém nenaslouchá žádná služba. V tomto případě se jedná o odezvu na transportní UDP komunikaci, která je nespojovaná

11 Promiskuitní režim síťové karty je takové nastavení, kdy veškerý síťový provoz, který tímto zařízením prochází, je zpracován mikroprocesorem. Pokud síťová karta přijme rámec, který jí není určen, potom (není-li karta v promiskuitním režimu) tento rámec zahodí.

(connectionless) a ke stavovým informacím využívá protokolu ICMP:

```
tcpdump -r data -n 'icmp[icmptype]== 3'
```

Program *tcpdump* není jediným, jsou zde i alternativní: *snoop* (Sun Snoop distribuován s operačním systémem Solaris), *etherpeek* (odposlouchávací program pro MAC OS), *netmetrix* (komerční program od společnosti HP), *ns* (network simulator od Lawrence Berkley National Laboratory). Vybral jsem *tcpdump*, neboť se s ním lze setkat zřejmě nejčastěji u operačních systému založených na platformě opensource, které jsou v oblasti serverů velice často využívané.

Pmacct

Pmacct (Promiscuous mode IP Accounting) je balík několika nástrojů pro pasivní monitoring sítě. Tyto nástroje umožňují klasifikaci síťového provozu. Pmacct podporuje protokoly sFlow (verze 2, 4 a 5) a také protokoly NetFlow (verze 1, 5, 7, 8 a 9). Data lze exportovat v obecném formátu tak, že umožňuje jejich vizualizaci v nástrojích např. RRDtools, GNUPlot, Net-SNMP, MRTG a Cacti (jedná se zde o opensource nástroje). Architektura nástroje Pmacct umožňuje data ukládat do SQL databází (MySQL, SQLite nebo PostgreSQL) a následně je exportovat do výše uvedených protokolů sFlow nebo NetFlow.

Softflow

Rozdíly mezi *tcpdump* a NetFlow/sFlow/IPFIX systémy spočívají v tom, že *tcpdump* zobrazuje (analyzuje) síťový provoz v reálném čase, tokové protokoly ovšem navíc v reálném čase zaznamenávají statistické údaje o datových tocích. Tokové protokoly jsou zpravidla implementovány ve směrovačích a integrovány do firewalů. Softflow získává data ze sítě (IP datagramy) tím, že naslouchá na síťovém rozhraní, a konvertuje je do formátu NetFlow protokolu. Následujícím příkazem spustíme službu Softflow, která bude monitorovat síťové rozhraní *eth0*:

```
root@server:#softflowd -i eth0 -t maxlife=300 -n localhost:6666
```

Parametr *maxlife* vynutí ukončení toku po 300 sekundách. Výchozí nastavení ukončí monitoring toku teprve v momentě, jakmile je řádně ukončen (např. je zaslán klientské straně poslední ukončující TCP paket daného spojení). Poslední parametr *-n* definuje cílový server, kolektor, kam se NetFlow pakety zasílají. Příkaz níže pak zobrazí aktuální toky:

```
root@server:~#softflowctl dump-flows
ACTIVE seq:42 [81.91.82.54]:80 <> [192.168.100.106]:45577
proto:6 octets>:6709 packets>:8 octets<:730 packets<:8
start:2009-07-29T18:12:39.566 finish:2009-07-29T18:12:39.999
tcp>:1b tcp<:1b flowlabel>:00000000 flowlabel<:00000000
EXPIRY EVENT for flow 42 in 293 seconds
```

Argus

Argus patří mezi proudové analyzátory (stream analyzer). Jedná se o klient-server aplikaci, kde serverová část sbírá data ze sítě a pomocí klientských nástrojů tato data lze analyzovat. Argus je proudový analyzátor, tzn. že data neanalyzuje na úrovni datagramů (nebo paketů), ale tak, že z nich sestavuje komunikační páry (kdo s kým komunikoval). Příklad užití:

```
root@server:#/usr/sbin/argus -w /var/log/argus/argus.log -n\
/var/run/argus.pid
```

Výše uvedeným příkazem byl spuštěn server Argus. Do souboru *argus.log* bude ukládat binární proudová data, které analyzujeme pomocí klientských programů. Analýzu nasbíraných dat provedeme následujícím příkazem (klientských analytických programů v rámci Argus balíku je celá řada):

```
root@server:#cat /var/log/argus/argus.log | ramon -M Matrix -r\
- filter
```

Uvedený skript pošle na vstup programu *ramon* soubor se získanými daty a zobrazí matici komunikačních uzlů:

```
17:41:09.060960  192.168.100.106 192.168.100.1 2743 2743 234431     489875
17:55:18.892791  192.168.100.106 81.91.82.54   2157 2278 208055     611094
18:17:50.445731  192.168.100.106 88.191.33.32   860 1112  70477    1631141
. . .
```

Podprobnější informace o užití programu Argus lze najít v manuálových stránkách k jednotlivým programům tohoto balíku, případně na Internetových stránkách autorů[12]

3.3 Nástroje pro analýzu síťových dat

Na Internetu je dostupno velké množství programových prostředků pro analýzu získaných dat ze sítě. U většiny aplikací se jejich vlastnosti kryjí, tzn. že pomocí různých nástrojů můžeme získat stejné výsledky. Vybíráme zde takové, které pokládáme za velmi vhodné pro použití při analýze dat získaných z odposlechu síťového provozu.

Tcpstat

Poskytuje statistiky ze síťového rozhraní. Naslouchá na požadovaném síťovém rozhraní a definovaném intervalu generuje výstup. Je možné jej též využít pro získání statistického materiálu z binárního souboru síťových dat získaných programem tcpdump (viz výše).

Statistické údaje, které je možné získat:

- šířka pásma,
- počty datagramů,
- počet datagramů za časovou jednotku (sekunda),
- průměrná velikost datagramu,
- směrodatná odchylka od průměrné velikosti datagramu,
- zatížení rozhraní atd.

12 http://www.qosient.com/argus/

Příklad užití:

```
root@server:#tcpstat -f 'port (smtp or http)' -o '%S %b\n' -r\
analyza 10
1247130270 383716.80
1247130280 0.00
1247130290 95012.80
1247130300 42180.80
```

Výše uvedený příklad provede statistickou analýzu získaného vzorku dat souboru *analyza*. Bude sledovat SMTP[13] nebo HTTP[14] provoz a výstupem jsou pak dva sloupečky – první (zleva) zobrazuje časovou značku (parametr %S) a druhý počet přenesených bitů za sekundu (parametr *%b*); pozn. znak „\n“ pak symbolizuje odřádkování. Úplně poslední parametr 10 pak stanovuje interval, tedy dobu, za kterou jsou naměřené hodnoty kalkulovány.

Tcptrace

Tcptrace je analytický nástroj orientovaný na analýzu spojení pomocí transportního protokolu TCP. Program *tcptrace* získává data z výstupu programu *tcpdump* a generuje sumarizaci jednotlivých TCP spojení. Dle našeho názoru je vhodný pro analýzu toků, nikoliv malého počtu paketů (v rámci pevně ohraničené množiny dat). Z malé množiny získaných paketů (jako výchozí jsme použili 30 paketů, více v diskusi 6.5) totiž nelze příliš dobře determinovat jednotlivá spojení. Důvodem je stochastičnost náhodných spojení ve smylu časové délky, počtu vyměněných paketů, četnost jejich výskytu v síti v čase apod. Pro analýzu toků je *tcptrace* velice vhodný.

Tcpflow

Tcpflow je program, který sbírá data přenášená jako součást TCP spojení (toků) a ukládá data konvenčním způsobem vhodným pro analýzu protokolů vyšších vrstev (aplikačních). Každý datový tok je pak reprezentován dvěma soubory – jedním souborem směrem od klienta k serveru a druhám souborem opačného směru. Názvy souborů mají následující podobu (příklad):

```
192.168.100.106.56820-192.168.100.001.00080
```

Název začíná IP adresou výchozího hostitele včetně zdrojového portu, pokračuje cílovým hostitelem a cílovým portem. Ve výše uvedeném příkladě se jedná o navázání spojení počítače 192.168.100.106 se serverem 192.168.100.1 a službou naslouchající na portu 80 (HTTP). Obsah souboru pak bude např.:

```
GET /index.html HTTP/1.0
```

pro získání souboru index.html z webového serveru.

13 Simple Mail Transfer Protocol – protokol pro přenos zpráv elektronické pošty (definován v RFC 821)

14 Hyper Text Transfer Protocol – bezstavový protokol určený zejména pro přenos webových stránek (poslední verze definovaná v RFC 2616)

Nfdump

Balík *nfdump* je určen k získání a následnému zpracování dat protokolu NetFlow. Získaná data (NetFlow data) jsou ukládána na disk. Pro sběr dat je třeba spustit program *nfcapd* (součástí balíku), který běží na pozadí jako démon a funguje jako tzv. NetFlow Collector. Tento balík v referenční linuxové distribuci (Debian) podporuje pouze protokol NetFlow, pro podporu protokolu sFlow pak je třeba rekompilace zdrojových kódů. *Nfdump* lze využít jako kolektor pro Cisco routery a další zařízení, která generují NetFlow statistiky.

3.4 Generování maligního síťového provozu

Možnost ověření, zda systém funguje správně, je najít takové principy, které dostatečně a spolehlivě ověří, zda navržený systém pracuje správně. Pro získání takového síťového provozu, který jsme schopni ohodnotit jako škodlivý, nebo potenciálně škodlivý, je třeba využít služeb takového softwaru, který se pro toto nežádoucí typ chování na síti používá. Zcela nepochybně ke generování nežádoucího síťového provozu mohou posloužit tzv. benchmarky. Software, který je explicitně určen pro testování softwaru. Příkladem může být program *ab* (Apache Benchmark), který slouží pro otestování nastavení webového serveru Apache[15]. Tento typ software typicky otevírá velké množství spojení a tak je možné toto chování odlišit od normálního.

Program Nmap

Tento program je mezi odborníky na počítačovou bezpečnost velmi znám. Využívá se zejména k získání informací o vzdáleném systému, o síti a jako nástroj bezpečnostního auditu. Nmap je nástroj spouštěný z příkazové řádky, ale též je k dispozici grafické rozhraní GUI[16] *zenmap*. Nmap je využíván zejména jako skener portů. Výstup takového skenu je seznam portů, které jsou otevřené, zavřené nebo detekované v několika dalších stavech. Stavy portů, které *nmap* rozlišuje:

- **open**, aplikace aktivně přijímá pokusy o navázání spojení (TCP nebo UDP pakety). Cílem skenování portů je vlastně objevení právě otevřených portů. Každý otevřený port je *de facto* cestou do systému. Útočníci mají za cíl exploitovat (zneužít) otevřené porty, zatímco administrátoři blokují tyto porty prostřednictvím firewallů (povolují pouze legitimní přistupy, např. z lokální sítě k databázové službě).
- **Closed**, zavřený port je přístupný (přijímá a odpovídá na testovací paketu Nmapu), ale žádná aplikace na něm nenaslouchá. Obecně se doporučuje přístup k těmto portům blokovat filtračními firewallovacími pravidly.
- **Filtered**, V tomto případě nmap není schopen spolehlivě určit zda je port otevřen, zda na něm nenaslouchá nějaká služba. Důvodem je, že na cílovém systému blokuje testovací pakety nmapu filtrační firewallovací pravidlo, které zabraňuje přístupu k tomuto portu. Takto „zajištěné" porty odrazují útočníky, neboť získávají od systému příliš málo informací. Někteří administrátoři odesílají na takové pakety, které přichází na nevyužité porty, ICMP kontrolní zpravávy typu Destination Unreachable: Communication Administratively Prohibited (typ 3, kód 13 v ICMP

15 Dostupný na adrese www.apache.org

16 GUI (Graphical User Interface) – grafické uživatelské rozhraní

hlavičce).

- **Unfiltered**, Port je přístupný, ale Nmap není schopen vyhodnotit, zda je otevřený či uzavřený (stav open nebo closed). Skenování nefiltrovaných portů se provádí speciálními technikami typu např. Windows sken, SYN sken nebo FIN sken. Tyto dodatečné skenovací techniky dokáží určit, zda je port otevřen.
- **open|filtered**, Tento stav se objevuje u takových skenovacích technik, kde otevřené porty nedávají žádnou odezvu. Takto klasifikuje porty UDP sken, IP protokol sken, FIN, NULL a Xmas sken.
- **closed|filtered**, Tento stav je zobrazen tehdy, pokud Nmap nedokáže určit stav portu, zda je otevřen, či filtrován. Využíván v IP ID skenu.

Nmap a techniky skenování portů

Skenování portů lze zařadit mezi vyvolané síťové anomálie. Dle [25] jsou rozlišovány tyto druhy skenů: **verikální** (které jsou cíleny na jednoho konkrétního hostitele a zkoumají velké množství portů) a **horizontální** (jsou cíleny na velký počet hostitelů (rozsahy IP adres), kde zkoumají jeden konkrétní port na všech hostitelých). Techniky, které se využívají pro detekci otevřených portů:

- **TCP SYN scan (parametr -sS)**, Skenovací technika, která se používá jako výchozí. Pracuje na principu polovičních spojení (half-open scanning), neotevírá plné TCP spojení. Začne tím, že zašle iniciální SYN paket (stejně jako při navazování TCP spojení) a čeká na odpověď cílového systému. Pokud se vrátí SYN/ACK paket, je port otevřen a naslouchá na něm služba (status open). Pokud systém odpoví RST paketem, indikuje, že na portu žádná služba nenaslouchá (status closed). V případě, že i po několika opakováních SYN paketů vzdálený systém neodpovídá, port je označen jako filtrováný (status filtered).
- **TCP connect scan (parametr -sT)**, Tato technika se používá v případech, pokud není možné spustit *nmap* v privilegovaném režimu (pod uživatelem *root*). Využívá systémového volání *connect()* a navazuje regulérní spojení s porty cílového stroje. Jednou z nevýhod tohoto postupu je, že vzdálený systém zpravidla zaznamenává pokusy o přihlášení (namísto SYN skenu, kde se spojení nikdy nevytvoří) a tak uživatel *nmap*-u může být snadno a rychle odhalen.
- **UDP scans (parametr -sU)**, Techniky UDP skenování jsou založeny na zaslání prázdného UDP paketu s nastaveným zkoumaným cílovým portem. Pokud vzdálený systém odpoví ICMP zprávou Port Unreachable, port je označen jako zavřený (status closed). Obdobně jako u SYN skenu, pokud nedojde k žádné odezvě na několik opakování stejného UDP paketu, port je označen jako filtrován (status open|filtered).
- **TCP NULL, FIN a Xmas (parametry -sN, -sF a -sX)**, Tyto tři skenovací techniky využívají malého nedostatku v doporučení RFC 793 k TCP protokolu, kde je definován otevřený a uzavřený stav portů. Všechny tyto skeny patří dotakzvané skupiny skrytých skenů. Při skenování se odesílá jeden paket a pouze jeden paket seočekává jako odpověď. Při těchto skenech se odesílají pakety s netradičně nastavenými příznaky v hlavičce TCP paketu. Až na FIN sken je kombinace těchto příznaků taková, že se v běžném provozu nemůže vyskytnout [13]. Pro více informací odkazuji např. na manuálovou stránku k programu *nmap*.

Příklad užití programu *nmap*:

```
server:~# nmap -PN -sS  atrey.vdsoft.org
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-
08-17 18:09 CEST
Interesting ports on atrey.vdsoft.org (81.91.82.55):
Not shown: 1671 closed ports
PORT    STATE     SERVICE
 21/tcp open      ftp
 22/tcp filtered  ssh
 25/tcp open      smtp
 53/tcp filtered  domain
 80/tcp open      http
110/tcp filtered  pop3
143/tcp filtered  imap
443/tcp open      https
993/tcp open      imaps
MAC Address: 00:16:3E:63:EC:42 (Xensource)
Nmap finished: 1 IP address (1 host up) scanned in 1.585
seconds
```

Výše je uvedený výstup tzv. vertikálního SYN skenu.

Program Mausezahn

Program Mausezahn (dále *mz)* je charakterizován v manuálové stránce výstižně: fast traffic generator written in C which allows you to send nearly every possible and impossible (překl. z angl.: „rychlý generátor síťového provozu psaný v jazyce C, který umožňuje zaslat možné i nemožné"). Mausezahn podporuje dva režimy: **raw-layer-2** a **higher-layer**. První umožňuje zaslat do síťového rozhraní jednotlivý byte, tedy bez ohledu na sestavování hlaviček vyšších protokolů. Druhý režim pak umožňuje uživateli vkládat (modifikovat) do hlaviček protokolů vyšších vrstev vlastní příznaky (flags). Program je možné využít jako nástroj k:

- generátor síťového provozu (př. zatěžování sítě),
- pro penetrační testování firewallů a IDS systémů,
- pro útoky typu DoS[17],
- pro možnost odhalit chyby v softwarových implementacích,
- atp.

Příklad užití programu Mausezahn (útok SYN Flood):

```
root@server# mz eth0 -A rand -B 192.168.100.1 -c 0 -t tcp\
"dp=1-1023,flags=syn" -P "Toto je SYN Flood"
```

Příkaz nahoře zašle stanici o IP adrese 192.168.100.1 na cílové porty v rozmezí 1 až 1023 z náhodných IP adres (parametr „-A rand") neomezený počet pokusů (parametr „-c 0") o navázání spojení TCP/SYN. Tato spojení se samozřejmě neuskuteční neboť zdrojové adresy již odpovědi na potvrzení přijetí TCP/SYN paketu nevrátí.

17 Denial of service (útok typu odepření služby) je takový typ útoku, kdy dojde k výraznému zhoršení kvality v přístupu k požadované službě, případně k jejímu kompletnímu znepřístupnění (např. pomocí úmyslného přetížení serveru).

4 Využití teorie hrubých množin

V posledních letech teorie hrubých množin (dále jen THM) zaznamenala velký zájem ze strany výzkumu a aplikace. THM má následující výhody:

- učení na malé datové množině,
- jednoduchost.

THM byla využita v mnoha oblastech data miningu a strojového učení, v bioinformatice, ekonomii, v oblastech zpracování signálů, robotice, včetně integrace do expertních systémů, je též možné ji využít v systémech na podporu rozhodování. THM je matematický nástroj pro analýzu vágních a nepřesných dat. (Anatomie tohoto přístupu je zcela odlišná z teorií fuzzy množin prof. Zadeha.)

V mnoha aplikacích [26] je dána množina objektů (stavů, procesů, pozorování atd.), ale my je nejsme schopni rozlišit pomocí dostupných měření, pozorování nebo popisů. Jinými slovy, náš nedostatek znalostí může být vážnou překážkou při rozhodování o objektech, fenoménech, procesech atd. **Tedy základní poznatek je, že vágnost nebo nepřesnost vede k nerozlišitelnosti**, která formálně reprezentuje náš nedostatek znalostí a je základní myšlenkou podtrhující filozofii hrubých množin. Relace nerozlišitelnosti se používá k definování základních operací na množinách – dolní a horní aproximaci množiny, které se využívají místo přesných pojmů. Obvykle ztotožňujeme pojmy s podmnožinami nějakého univerza. S každou množinou dat pak nakládáme jako se speciálním druhem rozhodovací tabulky a dolní a horní aproximace se používají k analýze vlastností rozhodovacích tabulek. Např. rozhodovací tabulka může obsahovat data o pacientech trpících nějakou určitou chorobou. Symptomy pacientů jsou pak považovány za podmínkové atributy a např. zdravotní stav pacienta se dá považovat za rozhodovací atribut. Pak hlavní otázkou je, zda zdravotní stav pacienta se dá definovat pomocí jeho symptomů. Jinými slovy, zajímáme se o závislosti mezi atributy v rozhodovací tabulce. Přístup teorie hrubých množin se ukázal pro analýzu otázek tohoto typu jako velmi vhodný. V současnosti byla implementována a úspěšně aplikována celá řada systémů analýzy počítačových dat, a to v průmyslu, lékařské sociologii, psychologii, pedagogice a dalších oborech. Matematicky vycházíme z pojmu aproximační prostor jako dvojice $A = (U, R)$, kde U je množina nazývaná univerzum a R je binární relace na U nazývaná relace nerozlišitelnosti. Tato relace musí být samozřejmě reflexivní a symetrická, tj. $R(x, x)$ a $R(x, y)$ implikuje $R(y, x)$ pro všechna $x, y \in U$. Tedy v obecném případě jde o tzv. relaci tolerance.

4.1 Definice informačního systému

Data jsou v informačních systémech reprezentována zpravidla v tabulkách. Každý řádek tabulky reprezentuje případ, událost, obecně sledovaný objekt. Každý sloupec pak reprezentuje měřenou hodnotu, sledovanou hodnotu na daném objektu; tato hodnota se nazývá atributem objektu. Každý objekt může mít více atributů. Tato tabulka bývá nazývána v teorii hrubých množin informačním systémem (dále v této kapitole budu používat tento pojem ve smyslu teorie hrubých množin). Formálně je informační systém charakterizován jako dvojice

$I = \langle U, A \rangle$, kde U je neprázdná konečná množina objektů zvaná *Univerzum* a A neprázdná konečná množina atributů. Dále uvažujeme funkci $g: U \times A \to V$, kde V je množina hodnot, které atribut může nabývat. Tato funkce g pak zajišťuje přiřazení hodnot atributů pro prvky univerza U.

Příklad informačního systému:

	Čas měření (s)	**Počet TCP/SYN**	**Počet ICMP/PU**[18]	**Počet TCP/RST**
o_1	0.5	12	1	2
o_2	1.1	1	5	1
o_3	0.5	12	1	2
o_4	0.8	1	3	1

Tabulka 2: Příklad informačního systému v teorii hrubých množin

Z tabulky je patrné, že objekt o_1 a objekt o_3 jsou totožné (mají naměřené stejné hodnoty). Tento případ se nazývá v THM *nerozlišitelnost*.

V mnoha aplikacích je výsledek klasifikace znám. Tato aposteriorní[19] znalost je vyjádřena zvláštním atributem, který nazveme rozhodovací atribut. Tento proces je znám jako učení s učitelem (supervised learning). Informační systémy tohoto druhu se nazývají rozhodovací systémy. Rozhodovací systém (formálně) je jakýkoliv informační systém v následující formě $I = \langle U, A \cup \{d\} \rangle$, kde $d \notin A$ je rozhodovací atribut. Prvky A se nazývají podmínky. Rozhodovací atribut může nabývat různých hodnot, binární hodnoty jsou u tohoto prvku dosti často používané.

Příklad rozhodovací tabulky:

	Čas měření (s)	**Počet TCP/SYN**	**Počet ICMP/PU**	**Počet TCP/RST**	**Anomální síťový provoz**
o_1	0.5	12	1	2	1
o_2	1.1	1	5	1	0
o_3	0.5	12	1	2	0
o_4	0.8	1	3	1	0

Tabulka 3: Příklad rozhodovací tabulky

Nerozlišitelnost a ekvivalence

Rozhodovací systém (rozhodovací tabulka) obsahuje veškeré znalosti, které jsou k dispozici k danému modelu. Tato tabulka může nabývat velkých rozměrů (implikuje zpravidla vysoké paměťové nároky na zpracování výpočetním systémem). Některé objekty mohou mít stejné vlastnosti (nerozslišitelnost objektů), resp. mohou se vyskytovat v tabulce vícekrát stejné objekty (ekvivalence objektů). Stejně tak některé atributy mohou být redundantní.

18 Port Unreachable – odezeva ICMP protokolu na přístup k portu, na kterém nenaslouchá žádná služba
19 ze zkušenosti

Ekvivalence objektů

Binární relace $R \subseteq X \times X$, která je

- **reflexivní** (objekt je v relaci sám se sebou x*R*x),
- **symetrická** (jestliže x*R*y potom y*R*x),
- **tranzitivní** (jestliže x*R*y a y*R*z potom x*R*z),

potom se tato relace nazývá ekvivalenční relací [27]. Ekvivalence je relace, která je současně reflexivní, symetrická a tranzitivní. Zachycuje nám vztah vypovídající o tom, že prvky, které jsou ekvivalentní, lze z jistého sledovaného hlediska zaměnit. Ekvivalence definuje rozklad množiny, na které je definována, na podmnožiny navzájem ekvivalentních prvků. Někdy se těmto množinám říká třídy ekvivalence nebo ekvivalenční třídy. Libovolný prvek z množiny rozkladu pak může sloužit jako reprezentant této množiny [28].

Třída ekvivalence prvku $x \in X$ sestává ze všech objektů $y \in X$ tak, že x*R*y. Nechť $I = \langle U, A \rangle$ je informačním systémem, potom s jakoukoliv $B \subseteq A$ je spojena ekvivalenční relace $IND_A(B)$:

$$IND_A(B) = \{(x_1, x_2) \in U^2 | \forall g \in B \, g(x_1) = g(x_2)\} \quad , \tag{1}$$

$IND_A(B)$ se pak nazývá *B-nerozlišitelná relace*.

Pokud objekty $(x_1, x_2) \in IND_A(B)$, potom jsou tyto objekty x_1, x_2 mezi sebou nerozlišitelné pomocí atributů v *B* [27].

Poznámka: Různé varianty neostrého a ostrého uspořádání nepopisují zcela adekvátně reálný svět. Zde totiž nejsme obvykle při posuzování preferencí schopni rozlišit drobné rozdíly. Tato relace nerozlišitelnosti (pod prahem, který vnímáme nebo bereme v úvahu) nemůže být ekvivalencí v námi zavedením slova smyslu. (Pro širší obeznámení s pojmem ekvivalence a rozlišitelnost odkazuji na literaturu [28])

4.2 Dolní a horní aproximace, hraniční region

Relace ekvivalence v podstatě člení prostor Univerza na množiny prvků (příkladů, objektů). Tyto části Univerza můžeme chápat jako podmnožiny. Podmnožiny, které nás zajímají, mají stejnou hodnotu výsledného atributu. Může se ale stát, že koncept „Anomální provoz ANO či NE" nemusí být zcela jasně určen. Nejsme schopni ostře a jasně definovat anomální provoz z konkrétního měření. Některé objekty určitě patří, jsme schopni též určit objekty, které nepochybně do množiny nepatří, a konečně jako poslední možnost – objekty se vyskytují na hraně – pokud tento hraniční region je neprázdná množina, obsahuje alespoň jeden případ, potom se této množině říká hrubá (rough set). Formálně můžeme výše uvedené popsat pomocí matematického aparátu následovně:

Nechť $I = \langle U, A \rangle$ je informační systém a nechť $B \subseteq A$ a $X \subseteq U$. Nyní jsme schopni aproximovat *X* využitím informací obsažených v *B* zkonstruováním *B*-dolní (-lower) a *B*-horní (-upper) aproximaci X.

Dolní a horní aproximaci značíme $\underline{B}(X)$ a $\overline{B}(X)$, kde

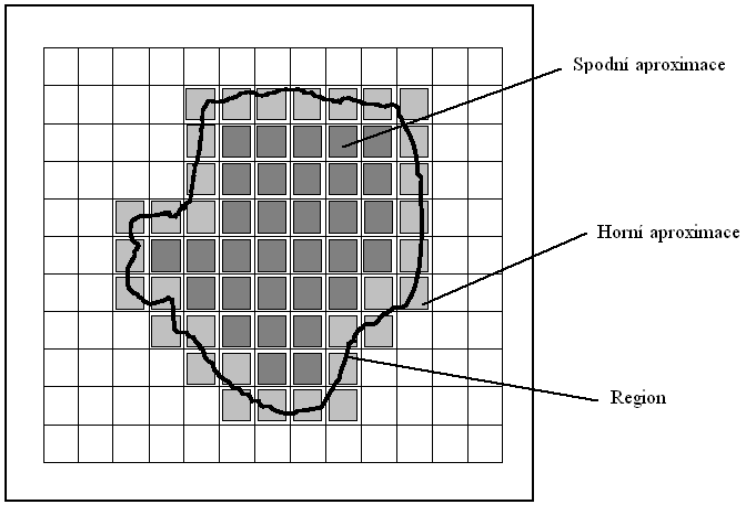
$$\underline{B}(X)=\{x|[x]_B\subseteq X\} \quad \text{a} \quad \overline{B}(X)=\{x|[x]_B\cap X\neq\emptyset\} \quad . \tag{2}$$

Objekty v $\underline{B}(X)$ mohou být s jistotou zařazeny (klasifikovány) jako prvky podmnožiny *X* (na základě znalostí z *B*), zatímco objekty v $\overline{B}(X)$ mohou být klasifikovány jako potenciální členové podmnožiny *X* na základě znalostí z *B*.

Dolní aproximace je popis objektů, o nichž lze s jistotou tvrdit, že náležejí do podmnožiny. Dolní aproximace bývá též nazývána pozitivní oblastí.

Hrubá množina (rough set)

Rough množina (hrubá, aproximovaná) je podmnožina *X* univerza *U*, která je definována pomocí horní a dolní aproximace a pro kterou platí, že hraniční region (boundary region) je neprázdná množina. Pomocí pojmu rough množiny lze definovat přibližnou přesnost, s jakou nalezená aproximace reprezentuje vybranou množinu *X*.



Obrázek 6: Reprezentace - dolní, horní aproximace, hraniční region [Zdroj: [29]]

Redukce dat a redukty

Množství dat, které máme zpracovávat, může být skutečně velmi rozsáhlé. Obzváště v oblasti počítačových sítí, kde přenos dat je jádrem jejich smysluplnosti. Nároky na systémové prostředky a výkon systémů na jejich zpracování přirozeně roste. Zajištění požadovaného výkonu ovšem velmi úzce koreluje s finančními nároky na sestavení takového výpočetního prostředí a tak je nutné racionálně k těmto datům přistupovat a vybrat z nich jen ta, které jsou pro nás významná. Zde se ale rodí dilema, která data můžeme a která v žádném případě nesmíme eliminovat. Ve své podstatě budeme z těchto dat získávat informace (v konečném důsledku znalosti), která jsou v nich skryta – půjde tedy o *netriviální získávání implicitních, dříve neznámých a potenciálně užitečných informací z dat* [30]. Z těchto důvodů je redukce dat velmi zásadní problém.

Jednou z možností, jak redukovat kvantum dat, je najít stejné objekty a ty sdružit v jeden; jeden, který je bude reprezentovat. Tento proces se nazývá hledání ekvivalentních tříd; tedy objektů, které jsou mezi sebou tzv. nerozlišitelné. Druhým rozměrem této redukce je zachovat jen ty atributy, které zachovávají vztah nerozlišitelnosti a tím vytvářejí aproximační prostor. Vynechané atributy se stávají redundantními a jejich vynecháním nemůžeme znehodnotit klasifikaci. Nalezení minimální redukce (tzn. s minimální kardinalitou atributů mezi všemi redukty) je NP-těžký problém[20]. Je snadné dokázat, že počet redukcí v informačním systému s m atributy může nabývat počtu $\binom{m}{m/2}$. To znamená, že výpočet redukcí není triviální úloha. Problém nalezení minimální redukce je tedy úzkým místem teorie hrubých množin. Existuje ovšem několik různých přístupů (heuristik), které toto úspěšně řeší v přijatelném čase. Tyto heuristiky jsou např. založeny na genetických algoritmech [27].

Redukty jsou charakterizovány v [27] maticemi rozlišitelnosti (discernibility matrices) a rozlišujícími funkcemi (discernibility functions). Nechť množina $U=\{x_1,x_2,\ldots,x_n\}$ a $A=\{a_1,a_2,\ldots,a_n\}$ jsou množiny informačního systému $I=\langle U,A\rangle$. Maticí rozlišitelnosti $M(I)$ v I je symetrická matice typu $m\times n$ s prázdnou diagonálou a prvky $c_{ij}i$:

$$c_{ij}=\{a\in A:a(x_i)\neq a(x_j)\} \quad . \tag{3}$$

Funkce rozlišitelnosti f_I je funkcí Booleovských proměnných $\bar{a}_1,\bar{a}_2,\ldots,\bar{a}_m$ korespondující s atributy $a_1,a_2,\ldots,a_m$ a je definována následovně:

$$f_I(a_1,a_2,\ldots,a_m)=\wedge\{\vee c_{ij}:1\leq j\leq i\leq n,c_{ij}\neq 0\} \quad , \tag{4}$$

kde $c_{ij}=\{a:a\in c_{ij}\}$.

20 NP-těžký problém souvisí s výpočetní složitostí. Potřeba času a paměti v závislosti na čase potřebném pro výpočet je charakterizována tak zvanou výpočetní složitostí algoritmu. Třída NP rozhodovacích problémů znamená, že existuje nedeterministický sekvenční algoritmus běžící v polynomiálním čase, který řeší daný problém [31].

5 Návrh modelu detekce síťových anomálií

Zde přicházíme s úvahou, jak definovat normalitu síťového chování, síťového provozu. Je třeba si uvědomit, že sítě mezi sebou jsou propojeny různými typy spojovací techniky, kabeláže, switchů, routerů; nejen od různých výrobců, ale též kombinují prvky různých přenosových rychlostí. Tato heterogennost sítě (např. Internet) je dána:

- různé přenosové rychlosti,
- nerovnoměrná a značně neproporcionální zátěž síťových segmentů (některé části sítě jsou saturované, jiné pak dostatečně nevyužité),
- různé typy sítí a užívaných přenosových protokolů (ATM[21], Ethernet, PPP[22]),
- atd.

Tyto rozdíly vedou k nepředvidatelnosti výsledného statistického snímku nasbíraných dat za daný časový interval. V jednom datovém balíku bude existovat velké množství pokusů o navázání spojení z jednoho hostitele na druhý, což by za jiných okolností vedlo k hypotéze např. TCP SYN Flood útoku, ovšem při bližší analýze se prokázalo, že cílový hostitel je z důvodu přerušení kabeláže odpojen od sítě a zdroj tak vysílá opakované pokusy o navázání spojení. Model tedy není možné nastavit jako striktní a musí bezpodmínečně uvažovat výše uvedené „přirozené“ anomálie.

Formálně lze model (vycházíme z předpisu dle Skowrona [27]) informačního systému pro detekci síťových anomálií definovat: $I=\langle U, A\cup\{d\}\rangle$, kde U je množina všech naměřených objektů, A množina atributů a d je množina rozhodnutí o vadnosti či bezvadnosti daného objektu (objekt je tedy chápán jako vzorek naměřených dat). Vycházíme-li z výše uvedeného, je možné sestavit konkrétní realizaci modelu:

$$I=\langle U,\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}\cup\{d\}\rangle \quad , \tag{5}$$

kde $d\in\{0,1\}$. Hodnota d pak znamená normálnost, resp. anomálnost objektu.

Volba diskriminátorů

Otázkou je, zda volba znaků (atributů) je schopna zajistit přesné zařazení objektů do tříd, tj. diskriminaci. Byla navržena řada postupů jak provést selekci znaků. Principem většiny metod je zajištění dostatečné separability tříd a volba takových diskriminátorů, které vedou k maximalizaci nějaké míry. Jindy se volí postup, kdy začínáme se všemi původními diskriminátory a postupně vypouštíme takové, které vedou k nedostatečné separaci [32].

Výběr atributů není triviální úloha. Každý datagram je v podstatě unikátní. Není-li evidentně malformován útočníkem, není tedy možné rozhodnout o jeho vadnosti či bezvadnosti. Mějme uspořádané množiny TCP, ICMP a ADDITIONAL. Tyto množiny obsahují vybrané (sledované) hodnoty jednotlivých parametrů protokolu TCP, ICMP a parametrů ADDITIONAL, které budou diskutovány dále. Jednotlivé prvky (v rámci

21 ATM – Asynchronous Transfer Mode

22 PPP – Point-to-Point Protocol – metoda pro přenos IP datagramů po sériových linkách; exituje více variant např. PPoE (Point-to-Point over Ethernet) nebo PPoA (Point-to-Point over ATM)

zobecnění) mohou nabývat jakýchkoliv číselných:

$$TCP = \{syn, ack, fin, urg, push, rst\} \quad , \tag{6}$$

$$ICMP = \{portunreachable\} \quad , \tag{7}$$

$$ADDITIONAL = \{tcpconn\} \quad . \tag{8}$$

Vektor $\boldsymbol{p_i} = \{TCP, ICMP, ADDITIONAL\}$, i=1, …, n; kde n je počet naměřených datagramů (v konfiguračním souboru *rough.conf* je počet definován proměnnou *PACKAGE_SIZE*), ze kterých se následně sestaví objekt (viz dále).

Dále mějme matici $M_k(n \times m)$. Tato matice se skládá z n řádků. Každý řádek představuje jeden datagram. Sloupce reprezentují naměřené hodnoty v každém z datagramů. Index k pak označuje číslo matice a může nabývat hodnot j=1,..., k, kde k je počet naměřených matic (v *rough.conf* je toto omezení definováno proměnnou *COUNTER_LIMIT*). Matice $\boldsymbol{M}$ vypadá mnemotechnicky následovně:

$$\begin{aligned} \boldsymbol{p_1}&: \{TCP_1, ICMP_1, ADDITIONAL_1\} \\ &\cdots \\ \boldsymbol{p_n}&: \{TCP_n, ICMP_n, ADDITIONAL_n\} \quad , \end{aligned} \tag{9}$$

formálně pak klasicky:

$$\boldsymbol{M_k} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & & & a_{2m} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} . \tag{10}$$

Dále uvažujme algoritmus, jehož výstupem je m-rozměrný vektor $\boldsymbol{o}$, nazýváme *objekt*, který určitým způsobem charakterizuje matici $\boldsymbol{M}$. Pokud budeme analyzovat síťový tok v reálném čase, je na algoritmus kladen požadavek nízké výpočetní náročnosti. Algoritmus pro sestavení i-tého charakteristického objektu (vektoru) lze formálně popsat takto:

$$\boldsymbol{o_k} = (\sum_{i=1}^{n} a_{i1}, \sum_{i=1}^{n} a_{i2}, ..., \sum_{i=1}^{n} a_{im}) \quad , \tag{11}$$

kde n je počet nasnímaných paketů v matici, m je počet měřených veličin a a_{ij} je j-tá měřená veličina na i-tém získaném paketu síťovým analyzátorem. Tento vektor bude dále pokládán za objekt (případ) ve smyslu teorie hrubých množin.

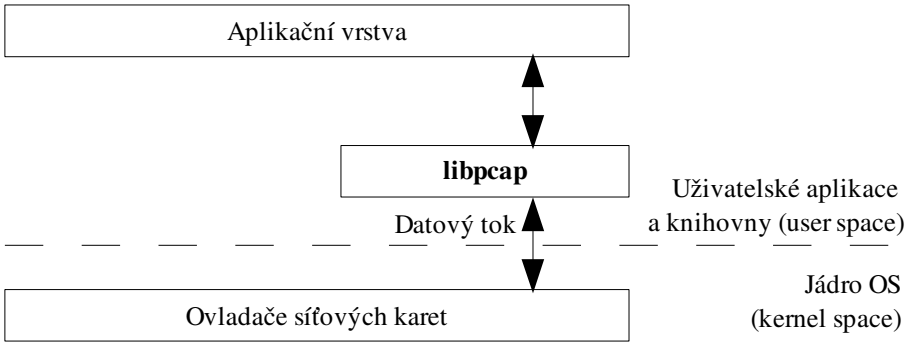
Nyní hledáme takové parametry (atributy, sloupce) matice ***M***, které ji nejlépe budou charakterizovat. Atributy (diskriminátory), které uvažuji za dostatečně vhodně charakterizující balík dat, jsou následující:

- navázání spojení (TCP/SYN),
- potvrzující paket (TCP/ACK),
- odmítnuté spojení (TCP/RST),
- ukončení spojení (TCP/FIN),
- urgentní paket (TCP/URG),
- push[23] paket (TCP/PSH),
- kontrolní zpráva o nedosažitelném portu (ICMP/Port Unreachable).

Budeme-li nasazovat výše uvedený model na systém typu HIDS, potom je možné doplnit ještě další atribut (diskriminátor): **počet aktivních spojení**. U systémů typu NIDS pak tento atribut je irelevantní, neboť s firewallem zpravidla žádné spojení navázané není.

Technologický model

Jádro operačního systému přijme datagram přicházející ze sítě a předá jej libpcap. (libpcap - system interface for user-level packet capture). Z *libpcap* získává aplikace jednotlivé pakety.



Obrázek 7: Pohle na tok dat operačním systémem [Zdroj: vlastní]

Libpcap poskytuje (system independent) framework pro nízko-úrovňový monitoring síťového provozu. Operační systémy využívají různé rozhraní pro sběr paketů, libpcap je založena na BSD[24] paket filtru (dále BPF[25]), BPF je standard pro 4.4BSD, BSD/OS[26], NetBSD, FreeBSD a OpenBSD operační systémy.

Níže uvedený diagram (Diagram 1) reprezentuje proces zpracování a vyhodnocení síťových dat. Za využití *libpcap* (knihovna popsána výše) jsou získávány síťové datagramy (Sběr dat). Množství a sledované parametry nejsou v diagramu v rámci snahy o zobecnění zmiňovány. Datový balík (nikoliv paket, ale vzorek naměřených dat) je posléze zaslána programovým prostředkům na zpracování, předpřípravě těchto dat a vytvoření objektu, resp. objektů pro předání k vyhodnocení (Nástroj pro zpracování dat pomocí hrubých

23 Příznak PUSH v TCP základví se zpravidla používá k signalizaci, že TCP segment nese aplikační data, příjemce má tato data předávat aplikaci. Použití tohoto příznaku není ustáleno. [1]
24 BSD – Berkley Software Distribution
25 BPF – Berkley Packet Filter
26 OS – Operating System

množin). Tento subproces (může být reprezentován aplikací RSES nebo jinou vlastní aplikací) aplikuje na předložený objekt, resp. objekty pravidla (z báze pravidel) a vyhodnotí objekty buď jako normální nebo anomální síťový provoz. V případě detekce anomálie uloží tento objekt do skladu anomálií pro pozdější vyhodnocení. Při této akci vyhodnocovaní aplikace reagovat dalšími způsoby (zaslání výstražného emailu, zablokování počítače firewallovacím pravidlem, aj.):

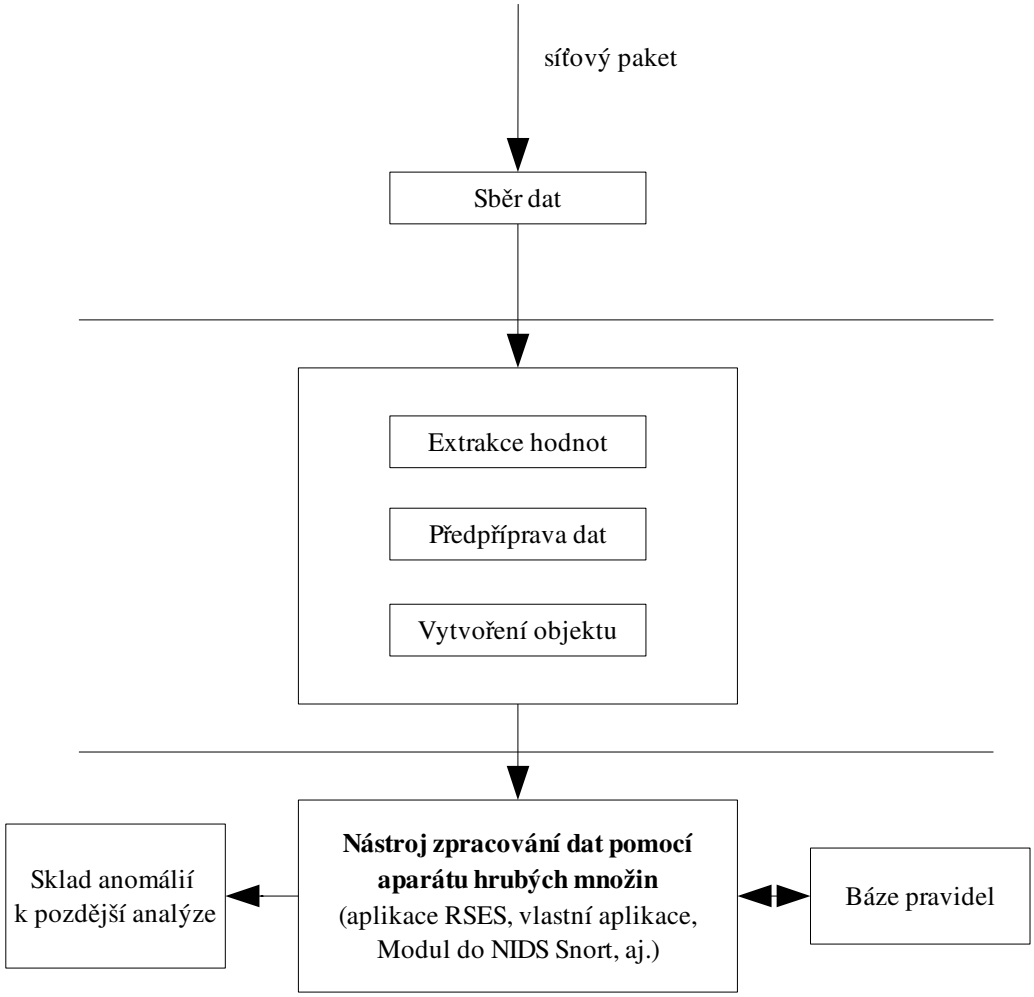


Diagram 1: Schéma cílového modelu [Zdroj: vlastní]

Pro zpracování a ohodnocení síťových dat byl sestaven vývojový diagram (Diagram 2). Tento model využívá sady vlastních skriptů pro získání dat (část skriptu *collect_and_clasify.sh*), analýzu dat a vytvoření „syrového" objektu (skript *create_object.sh*), přípravy dat pro předložení ke klasifikaci (skript *scaling.sh*) a zformátování do takové podoby, které rozumí některý z analytických nástrojů (skript *rough_format.sh*). Následující diagram reprezentuje způsob předzpracování dat:

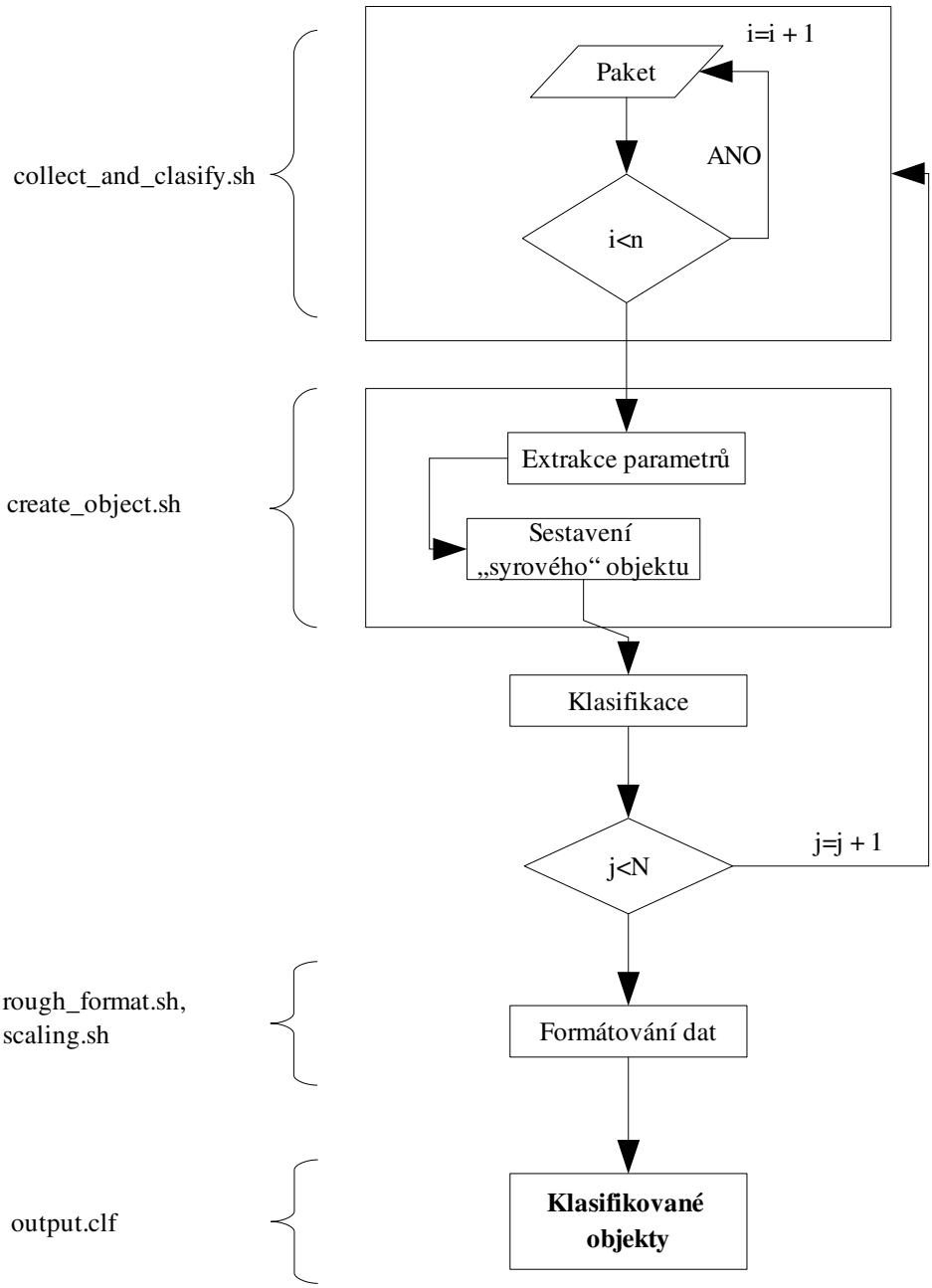


Diagram 2: Diagram sestavení souboru klasifikovaných objektů [Zdroj: vlastní]

V diagramu (Diagram 2) je definována rozhodovací situace „i<n“, kde *i* je hodnota pořadí přijatého paketu (určuje číslo získaného paketu ze sítě a s každým nově přijatým paketem se zvyšuje o 1). Systém získává pakety do té chvíle, kdy platí podmínka „i<n“, kde *n* reprezentuje počet získaných paketů pro analýzu (tato hodnota je definována v souboru *rough.conf*, v proměnné *PACKAGE_SIZE*). O sběr paketů se stará část skriptu *collect_and_clasify.sh*. Jakmile je stanovený počet paketů nasnímán, skript *collect_and_clasify.sh* volá skript *create_object.sh*, který pomocí analytických příkazů *tcpdump*, *tcpstat*, *nestat*, aj., provede extrakci parametrů z jednotlivých paketů a sestaví „syrový“ objekt. Objekt je dále klasifikován. Klasifikace se děje dvěma způsoby: asistovaně (ručně, uživatelem) anebo automaticky. Následuje rozhodovací situace „j<N“,

kde *N* reprezentuje množství objektů, které bude předloženo ke zpracování. Jakmile je porušena výše uvedená podmínka, je volán (volitelně) skript *scaling.sh,* který provede na daných atributech zařazení jejich hodnot do tříd. Poté je spuštěn skript, který zformátuje objekty do jednoho z podporovaných formátů (rses nebo rslib). Výsledkem celého procesu je soubor s objekty k analýze (soubor *output.clf*).

6 Zpracování síťových dat

Získaná data jsou v syrovém tvaru (raw data). Abychom je mohli předložit nějakému algoritmu, musíme je předzpracovat, tedy upravit do takového formátu, kterému bude daný algoritmus rozumět. Data musí být:

- pro algoritmus srozumitelná (textové hodnoty převést na číselné, …),
- v podporovaném formátu.

Z dat napřed je nutné vyextrahovat informace, tedy z následující řádky je třeba vybrat takové atributy, které budou dostatečně a spolehlivě diskriminovat validitu provozu:

```
059319 IP (tos 0x10, ttl 64, id 31072, offset 0, flags [DF], proto TCP
(6), length 60) 10.106.38.35.38979 > 81.91.82.54.80: S, cksum 0x8185
(correct),3229248970:3229248970(0) win 5840 <mss 1460,sackOK,timestamp
726186 0,nop,wscale 6>
```

Z výše uvedeného příkladu výstupu programu *tcpdump* je patrné, že se jedná o pokus o navázání spojení počítače o IP adrese 10.106.38.35 (zdrojový port pak využil 38979) se serverem o IP adrese 81.91.82.54 ke službe naslouchající na portu 80 (webová služba HTTP[27]). Navázání spojení je zde určeno příznakem SYN v hlavičce protokolu TCP; tento příznak je zvýrazněn. Podrobnější informace o výstupu programu *tcpdump* naleznete v jeho manuálové stránce (příkaz: *man 8 tcpdump*).

6.1 Sběr dat (collect_and_clasify.sh)

Skript *collect_and_clasify.sh* slouží pro vytvoření množiny klasifikovatelných objektů.

```
#!/bin/bash
CONF=/root/RSES/rough.conf
if [ -z $1 ]; then echo "Zadej nazev souboru jako parametr.";
exit 1; fi
if [ $2 = "auto" ]; then
 if [ -z $3 ]; then
      echo "Zadej automaticky klasifikator. Priklad: \
./collect_and_clasify.sh soubor auto 1";
      exit 1; fi
 fi

if [ -x $CONF ]; then echo "Nenalezen konfiguracni soubor \
$CONF"; exit 1; fi

source $CONF

# Jmeno informacniho systemu
NAME=$1

# Vystupni soubor
DATA_FILE=$NAME
# Soubor s klasifikovanymi objekty
DATA_FILE_CLF=${DATA_FILE}.clf
# Vystupi soubor s celym informacnim systemem pro rough sets
```

27 Hyper Text Transfer Protocol

```
DATA_FILE_IS=${DATA_FILE}.sys
# Soubor, jehoz data jsou skalovana
DATA_FILE_SCALED=${DATA_FILE_CLF}.scaled
# COUNTER - citac pro pocet nasnimanych datagramu
COUTER=0
tcpdump_cmd="tcpdump -w $DATA_FILE -n -tt -vvv -p -l -S -c \
$PACKAGE_SIZE ip"
tcpdump_cmd_show="tcpdump -r $DATA_FILE -n -tt -vvv -p -l -S \
ip"
sestaveni_objektu="/root/RSES/create_object.sh"
skalovani="/root/RSES/scaling.sh"

rm ${DATA_FILE}.clf
while [ "${decision}" !=  "done" ];
do
      let COUNTER=COUNTER+1
      echo "Citac: $COUNTER"

      $tcpdump_cmd 2>/dev/null
      characteristics=`$sestaveni_objektu $DATA_FILE \
2>/dev/null`
#     $tcpdump_cmd_show
      echo "Charakteristiky: $characteristics"
      echo "Klasifikuj [0,1,2,..], pro ukonceni napis \"done\""
      if [ $2 = "auto" ]; then decision=$3; else
      read decision
      fi
      rs_object="${characteristics}:${decision}"
      if [  $COUNTER -gt $COUNTER_LIMIT ]; then
decision="done"; fi
      if [ "$decision" = "done" ]; then continue;
      else
            echo $rs_object
            echo $rs_object >> ${DATA_FILE}.clf
      fi
done
$ROUGH_FORMAT ${DATA_FILE_CLF}
```

6.2 Sestavení objektu (create_object.sh)

Nyní vše je nutné spojit všechny dílčí analýzy do jednoho celku a vytvořit klasifikovatelný objekt. Sestavíme tedy následující skript *create_object.sh*:

```
#!/bin/sh
DATAFILE=$1
tcpdump_cmd="tcpdump -n -r $DATAFILE"
tcpstat_cmd="tcpstat -r $DATAFILE"

# ATRIBUT: CAS
#STARTTIME=`$tcpdump_cmd -r $DATAFILE -tt | head -n 1 $PACKAGE
| awk '{print $1}'`
#ENDTIME=`$tcpdump_cmd -r $DATAFILE -tt   | tail -n 1 $PACKAGE
| awk '{print $1}'`
#DIFF=`echo "$ENDTIME - $STARTTIME"| bc`

# ATRIBUT: Pocet pokusu o navazani spojeni (TCP/SYN)
TCP_SYN=`$tcpdump_cmd -n 'tcp[13] ==2' | wc -l `
#TCP_SYN_COUNT=`$tcpstat_cmd -f 'tcp[13] ==2' -o '%n'`
```

```
# ATRIBUT: Pocet odmitnuti spojeni (TCP/RST)
TCP_RST=`$tcpdump_cmd -n 'tcp[13] &4 !=0' | wc -l `

# ATRIBUT: Pocet ukonceni spojeni (TCP/FIN)
TCP_FIN=`$tcpdump_cmd -n 'tcp[13] ==1' | wc -l`

# ATRIBUT: Pocet ukonceni spojeni (TCP/ACK)
TCP_ACK=`$tcpdump_cmd -n 'tcp[13] ==16' | wc -l`

# ATRIBUT: Pocet ukonceni spojeni (TCP/URG)
TCP_URG=`$tcpdump_cmd -n 'tcp[13] ==32' | wc -l`

# ATRIBUT: Pocet ukonceni spojeni (TCP/PSH)
TCP_PSH=`$tcpdump_cmd -n 'tcp[13] ==8' | wc -l`

# ATRIBUT: Pocet odmitnuti spojeni ICMP zpravou (port
unreachable)
UDP_PUNREACHABLE=`$tcpdump_cmd -n 'icmp[icmptype]== 3' | wc -l`

# ATRIBUT: pocet navazanych spojeni
TCP_CONN=`netstat -n -4  --protocol=inet | grep -v 127.0 | grep
ESTABLISHED | wc -l`

echo "$TCP_SYN:$TCP_RST:$TCP_FIN:\
$TCP_ACK:$TCP_URG:$TCP_PSH:\
$UDP_PUNREACHABLE:$TCP_CONN"
```

Tento skript je velice důležitý. Extrahuje ze získaných dat charakteristiky, které budou dále syntetizovány v atributový prostor. Poslední tři řádky jsou výstupem skriptu. Na kvalitě zpracování tohoto skriptu velice záleží, neb se stane úzkým místem (z hlediska výkonu) v rámci procesu analýzy dat v reálném provozu. Skript je využíván ve dvou procesech:

1. proces vytváření modelu,
2. proces zpracování a klasifikace dat za reálného provozu.

V prvním případě požadavek výkonu není až tolik podstatný – vytváření modelu na soudobých výpočetních systémech i kancelářského typu není příliš výkonově náročný a navíc můžeme tento proces zpracovávat offline[28]. V druhém pak požadavek optimality vytvoření objektu patří mezi klíčové. Poslední řádka skriptu *create_object.sh* je jeho výstupem, tedy objekt. Jednotlivé atributy jsou oddělené dvojtečkou.

6.3 Škálování (scaling.sh)

Formálně lze pro stanovení počtu intervalů použít tzv. Sturgesovo pravidlo[33], podle něhož počet intervalů k je přibližně dán tímto vzorcem

$$k \sim 1 + 3{,}3 \cdot \log(n) \quad , \tag{12}$$

kde n je rozsah souboru (v našem případě pak počet klasifikovaných objektů). Máme-li již stanoven počet intervalů, pak šířka intervalu bude

28 Zde se má na mysli, že zpracování nevyžaduje připojení k síti.

$$\frac{x_{max} - x_{min}}{k}, \tag{13}$$

kde $x_{max} - x_{min}$ je variační rozpětí mezi maximální a minimální hodnotou znaku v daném souboru. Pro 100 hodnot pak bude Sturgesovo pravidlo implementováno tímto způsobem:

```
root@server:~# echo “scale=0; 1 + 3.3 * l(100})*l(10)” | bc -l
7
```

Tučně zvýrazněná hodnota je výstupem příkazu, udává 7 tříd. Pro výpočet počtu tříd podle Sturgesova pravidla byl využit program *bc*. Nyní provedeme rozdělení hodnot do tříd pomocí skriptu *scaling.sh*:

```
#!/bin/bash
CONF=/root/RSES/rough.conf
source $CONF
# Soubor s daty
FILE_TO_RANGE=$1
# Index skalovaneho atributu
FIELD=$2
# Vystupni soubor (zatrideny atribut $FIELD)
OUTPUT=${FILE_TO_RANGE}.scaled
# Pocet klasifikovanych objektu
N=`cat ${FILE_TO_RANGE} | wc -l`
# Pocet trid navrhnutych Sturgesovym pravidlem
C=`echo "scale=0; 1 + 3.3*l($N)/l(10)" | bc -l $BC_LB`
source $CONF
# Vypocet maximalni, minimalni hodnoty a rozsahu
MIN=`cat $FILE_TO_RANGE | cut -d: -f$FIELD | sort -n | head -n
1`
MAX=`cat $FILE_TO_RANGE | cut -d: -f$FIELD | sort -n | tail -n
1`
RANGE=`echo "scale=0; $MAX - $MIN" | bc -l`
# Sirka jedne tridy
CLASS_RANGE=`echo "scale=0; $RANGE / $C * $SENSITIVITY" | bc -l
`
echo "Pocet objektu: $N"
echo "Pocet trid: $C"
echo "Rozsah hodnot: $RANGE"
echo "Sirka tridy: $CLASS_RANGE"
object=""
for i in `cat $FILE_TO_RANGE`;
do
      WILL_BE_RANGED=`echo $i | cut -d: -f$FIELD`
      NUMBER_TO_RANGE=`echo "$WILL_BE_RANGED * $SENSITIVITY" |
bc -l `
      if [ $NUMBER_TO_RANGE -gt $CLASS_RANGE ]; then
CLASS=`expr $C - 1`;
            else
             CLASS=`echo "scale=0; mod($NUMBER_TO_RANGE,$C)" |
bc -l $BC_LB`
            fi
      for a in `seq $ATTRIBUTES_COUNT`; do
            if [ $a == $FIELD ]; then echo -n "${CLASS}:"
                  else
                        number=`echo $i | cut -d: -f$a`
                        echo -n "${number}:"
                  fi
```

```
        done
        # Odradkovani objektu
        echo
done
```

Skript lze využít následujícím způsobem. Nabývá-li 5-tý atribut mnoha různých hodnot, je možné je zařadit do tříd:

```
root@server:~# /root/RSES/scaling.sh test.clf 5 > \
test.clf.scaled
```

Výsledkem bude soubor *test.clf.scaled*, jehož hodnoty 5-tého atributu budou zařazeny do tříd. Pro počet tříd, jak bylo uvedeno, využíváme Sturgesova pravidla.

Dynamické a statické škálování

Je možné použít dva způsoby škálování – **fixní** nebo **dynamické**. V případě užití fixní škálovací stupnice (dále statické škálování), tedy takové, která roztřídí objekty časový atribut podle předem pevně stanovených hranic. Druhý způsob škálování je dynamický, závislý na škálovací stupnici stanovené v trénovací fázi. Každý testovaný objekt, resp. měřený atribut, musí nevyhnutelně využít buď jeden nebo druhý škálovací princip. Výhoda prvního přístupu je v tom, že je tzv. univerzální, možná i v konečném důsledku při použití vyhodnocovacích algoritmů rychlejší. Dynamický přístup má ovšem nespornou výhodu v tom, že nespotřebovává velké množství prostoru hodnot pro atribut. To ovšem má za následek potenciálně nižší přesnost (pokud statické škálování podporuje velké množství zpřesňujících tříd), na druhou stranu šetří výsledný pravidlový prostor. Škálovací dilema tedy řeší otázku, zda je výhodnější využít menší množství tříd pro atribut s tím, že jejich počet definujeme dynamicky podle počtu měřených objektů, nebo stanovit pevné a neměnné škálovací měřítko pro všechny případy. První varianta lépe přiléhá požadavkům konkrétního případu pro klasifikaci objektů v rámci teorie hrubých množin. Je ovšem možné pohodlně využít druhé varianty dynamického škálování. V takovém případě je ale nutné spolu s natrénovaným modelem pravidel distribuovat na cílový systém též parametr škálovací stupnice; tedy počet tříd, do kterých je daný atribut rozdělen. Tento počet tříd pak klasifikační algoritmus na cílovém systému musí bezpodmínečně respektovat. Osobně jsem zastáncem druhé, dynamické, varianty s tím, že na cílový systém, kde bude probíhat klasifikace provozu, bude distribuována i soustava parametrů – škálování a velikost datového balíku (vzorku).

6.4 Formátování (skript rough_format.sh)

Tento skript formátuje syrová data (v podobě nasbíraných objektů) do takové podoby, které bude rozumět aplikace, která daná data zpracovává. Skript podporuje dva výstupní formáty dat (definované proměnnou *OUTPUT_TYPE* v souboru *rough.conf*):

- **rses**, který je lze použít s programem RSES (Rough Set Exploration System),
- **rslib**, tento formát lze použít s RSL (Rough Set Library), knihovna pro jazyk C.

Příklad užití:

```
root@server:/tmp#/root/RSES/rough_format.sh MODEL.clf
```

Výstupem pak bude soubor *MODEL.clf.sys*, který je již připraven pro zpracování vybraným nástrojem. Aplikace RSES dokáže načíst pouze soubor s koncovkou *.tab*, proto je nutné zformátovaný soubor *MODEL.clf.sys* přejmenovat.

Rozdíl mezi formátem rses a rslib je pouze v hlavičce před samotnými daty:

Formát RSES

```
TABLE example.clf
ATTRIBUTES 9
"attr1" numeric 0
"attr2" numeric 0
"attr3" numeric 0
"attr4" numeric 0
"attr5" numeric 0
"attr6" numeric 0
"attr7" numeric 0
"attr8" numeric 0
"attr9" numeric 0
OBJECTS 1623
0 0 0 20 0 0 0 3 0
0 0 0 19 0 0 0 3 0
. . .
```

Formát RSLIB

```
NAME: MODEL.clf
ATTRIBUTES: 9
OBJECTS: 1623
0 0 0 20 0 0 0 3 0
0 0 0 19 0 0 0 3 0
. . .
```

6.5 Konfigurační parametry

Jedním z diskutabilních parametrů pro nastavení sběrného systému je *PACKAGE_SIZE*. Tento parametr definuje kolik datagramů se má ze sítě získat před samotnou klasifikací. Velice důležité je zmínit, že je nutné jeho hodnotu zachovat stejnou jak při získávání dat pro generování pravidel, tak při aplikaci vygenerovaných pravidel (modelu) v rámci hledání síťových anomálií. Nedodržení (nekonzistence) v tomto smyslu vede k aplikaci špatného modelu a nesprávné klasifikaci síťového provozu. Z tohoto důvodu je třeba pravidla (model) distribuovat na cílové systémy, mimojiné, společně s tímto parametrem. Dalším tématem k diskusi by mohl být počet měřených hodnot v datovém balíku. Zda soustava atributů (poslední řádka skriptu *create_object.sh*)

```
echo "$TCP_SYN:$TCP_RST:$TCP_FIN:\
      $TCP_ACK:$TCP_URG:$TCP_PSH:\
      $UDP_PUNREACHABLE:$TCP_CONN"
```

je dostatečná v rámci přesnosti klasifikace. Otázkou je, zda konkrétně tyto parametry dostatečně a spolehlivě diskriminují síťový provoz z hlediska normality. Pokud zvýšíme atributový prostor o dodatečné parametry měření (např. zavedeme další parametry počtu TCP a UDP paketů, aj.), zcela nepochybně zpřesníme inferenční výsledky. Výsledkem ovšem bude ale větší množství pravidel, jejichž aplikace bude mít za následek snížení rychlosti inferenčního systému.

6.6 Simulace webového provozu (http_traffic.sh)

Pro simulaci bežného síťového provozu je připraven skript *http_traffic.sh*. Tento jednoduchý skript simuluje očekávaný provoz typu HTTP. Tento skript realizuje skutečně vzdálený pohled od reálného síťového provozu a lze jej využít v podmínkách laboratorního typu nebo jako podpůrný prostředek při modelování normálního síťového provozu.

```
#!/bin/bash
TMP=`mktemp`
DATAFILE="adresy.txt"
LOAD="30"
LIMIT="1000"
SIZE=`cat $DATAFILE | wc -l`
for i in `seq 1..$LIMIT`; do
        let R=$RANDOM%${SIZE}; echo $R
      site=`cat $DATAFILE | sed "$R,$R !d"`
      echo "Stahuji ... $site"
       wget -O $TMP -p -l 1  http://${site}/ -o /dev/null \
>/dev/null 2>&1
        let S=$RANDOM%{LOAD}
      echo "Cekam $S sekund..."
      sleep $S
done
rm $TMP
```

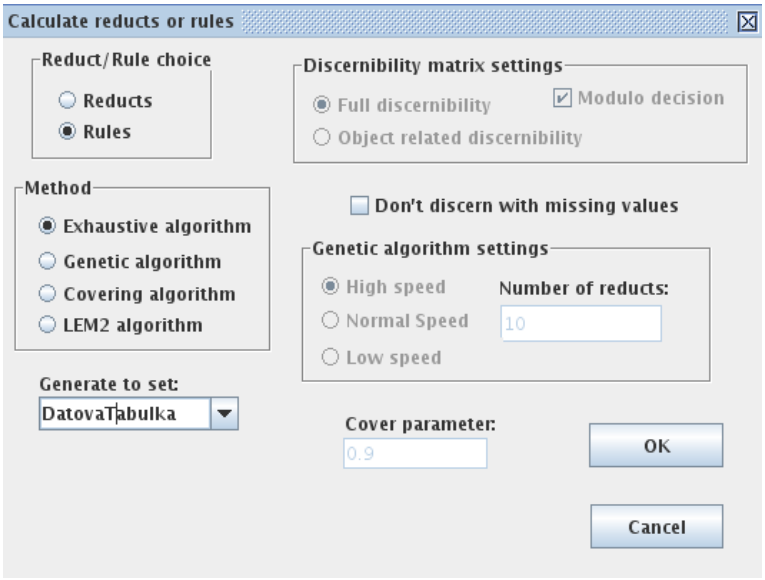
Skript pracuje na principu internetového prohlížeče tak, že v cyklu načítá náhodně vybranou webovou stránku a poté čeká náhodnou dobu před další iterací. Proměnná *DATAFILE* definuje soubor se seznamem adres, které se budou v cyklu načítat, proměnná *LOAD* definuje rozsah (od 0 do hodnoty této proměnné), ve kterém se bude pohybovat náhodná hodnota stanovující dobu čekání mezi jednotlivými iteracemi. Proměnnou *LOAD* je dobré přizpůsobit očekávanému zatížení sítě. Zvýšením de facto snižujeme frekvenci opakování. Proměnná *LIMIT* určuje maximální počet opakování.

Simulačních skriptů by bylo lze napsat na každý typ legitimního síťového provozu několik. Dále při simulaci tohoto provozu lze doporučit kombinovat spouštění jednotlivých skriptů: časově každý zvlášť (abychom podchytili každý typ provozu samostatně) a paralelní spouštění s možným časovým překryvem.

7 Klasifikace síťového provozu

Za využití aplikace RSES (Rough Set Exploration System doustupný z webových stránek http://logic.mimuw.edu.pl/~rses/) provedeme demonstraci fungování detekce síťové anomálie. RSES byl vybrán z důvodu komfortní práce a hlavně díky názorným prezentačním vlastnostem.

7.1 Generování pravidel v RSES



Obrázek 8: Dialogové okno aplikace RSES pro generování pravidel [Zdroj: vlastní]

Metody/algoritmy pro generování pravidel

- **Exhaustive algorithm** („Úplný" algoritmus), tento algoritmus je založen na generování pravidel pomocí výpočtu objektově orientovaných redukcí. Ukázalo se, že minimální počet konzistentních pravidel pro danou rozhodovací tabulku můžeme získat tím, že prostě eliminujeme objekty s nadbytečnými deskriptory.

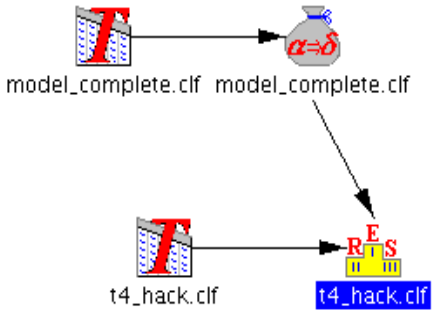
- **Genetic algorithm** (Genetický algoritmus), je schopen vygenerovat definovaný počet výsledných pravidel na základě operací genetického programování. Je popsán v článku [34].

- **Covering algorithm** („Pokrývání/Pokrývací" algoritmus), tento algoritmus hledá minimální množinu pravidel, které pokrývají celou množinu objektů. Tento způsob generování pravidel je popsaný v práci [35].

- **LEM2 algorithm**, jedná se o další variantu obecného pokrývacího algoritmu, více o LEM2 v práci [36].

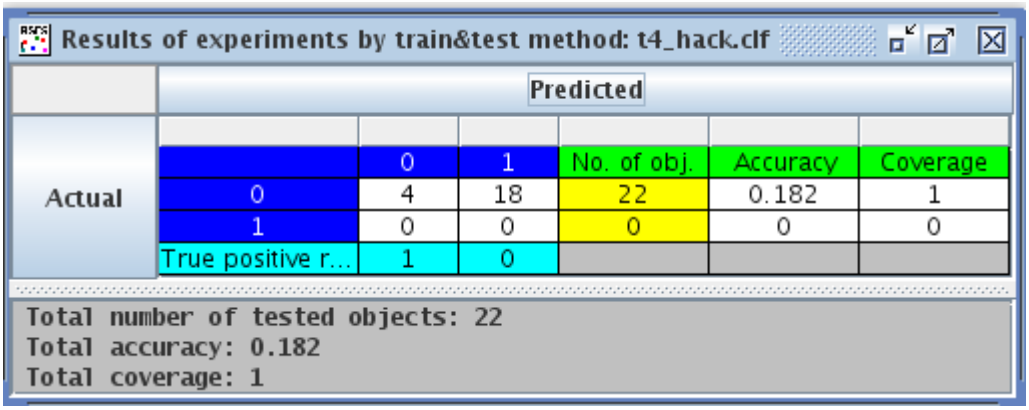
7.2 Interpretace výsledků

Mějme důvěryhodnostní model *model_complete.clf*. Jedná se o soubor objektů naměřených v laboratorních podmínkách a obsahuje jak normální síťový provoz, tak též případy skenování portů. Dále mějme soubor případů *t4_hack.clf* (tzv. testovací množina dat), který obsahuje objekty charakteristické právě pro skenování portů, ale též rezidua běžného provozu. Z důvěryhodnostního modelu vytvoříme pravidla. Dále předložíme testovací množinu těmto pravidlům a získáme výsledek klasifikace. Dvojitým poklikáním na ikonu reprezentující výsledky klasifikace,



Obrázek 9: Označená ikona výsledků klasifikace [Zdroj: vlastní]

zobrazíme výsledky:



Obrázek 10: Okno s výsledky klasifikace [Zdroj: vlastní]

Okno s výsledky klasifikace poskytuje různorodé informace. V centrální části je umístěna tabulka pro posuzování úspěšnosti (klasifikačních) algoritmu (dále *confusion matrix*):

	0	1
0	4	18
1	0	0
True positive r...	1	0

Obrázek 11: Tabulka pro posuzování úspěšnosti klasifikace v RSES [Zdroj: vlastní]

Řádky matice korespondují s aktuálními rozhodovacími třídami (všechny možné hodnoty

rozhodnutí). Sloupce pak reprezentují rozhodovací hodnoty, které vrátil klasifikátor. V našem případě (Obrázek 11) 18 hodnot.

Další dodatečné sloupce poskytují informace v tomto pořadí (popis je inspirován [37]):

- **No. of obj.** (zkratka pro Number of objects), je počet objektů v datové množině, které patří do třídy v daném řádku,
- **Acuracy** (překl. z angl. přesnost), poměr správně ohodnocených (klasifikovaných) objektů třídy ku celkovému počtu objektů,
- **Coverage** (překl. z angl. pokrytí), poměr klasifikovaných objektů (těch, které byly rozpoznány klasifikátorem) dané třídy ku celkovému počtu všech objektů v dané třídě.

Poslední řádek tabulky obsahuje **True positive rate** pro každou třídu. Pod tabulkou jsou dále zmíněny doplňující hodnoty:

- **Total number of tested objects**, celkový počet testovaných objektů na dané testovací množině,
- **Total accuracy**, poměr mezi správně klasifikovanými případy ku celkovému počtu testovaných případů,
- **Total coverage**, procentuální zastoupení objektů, které byly rozpoznány klasifikátorem.

V našem příkladě (Obrázek 10) je **Total coverage** rovno 1, což znamená, že všechny předložené objekty (testovaná množina *t4_hack.clf*) byly rozpoznány (oklasifikovány). Jak je vidět dále, máme 4 objekty, které byly rozpoznány jako 0 (normální chování) a 18 objektů (případů), kdy se jednalo o anomálii, resp. 18 objektů bylo zařazeno do třídy 1, kterou chápeme jako anomální provoz.
Tím, že **Total coverage** je 1, znamená to, že každý objekt byl rozpoznán klasifikátorem a tak byl schopen všechny objekty zatřídit. Pokud by Total coverage byl menší jak 1, potom by bylo vhodné objekty, které nebyly rozpoznány, nově oklasifikovat a zařadit do důvěryhodnostního modelu(*model_complete.clf).*

7.3 Konfigurace vlastních skriptů

Před samotným zahájením sběru a následné analýzy dat je nutné provést zkonfigurování podpůrných skriptů. Konfigurace není jen v korektním nastavení cest k programům (*tcpdump*, *tcpstat*, aj.), ale též nastavení proměnných. Tyto proměnné je pak třeba přizpůsobit druhu sítě, na které se bude systém učit a následně pracovat.

Neprve provedeme editaci souboru *collect_and_clasify.sh*. Zde změníme následující parametry:

```
ATTRIBUTES_COUNT=4
PACKAGE_SIZE=30
OUTPUT_TYPE=rses
```

První parametr, *ATTRIBUTES_COUNT*, definuje mohutnost atributového prostoru (počet měření, které provádíme na jednom vzorku síťového provozu včetně jednorozměrného vektoru pro klasifikaci). Druhý parametr, *PACKAGE_SIZE*, je velice důležitý.

Zde definujeme velikost datového vzorku k analýze. Na vytěžovaných sítích doporučuji hodnotu zvětšit. (Tento parametr je nutné zachovat i při samotné analýze dat pomocí aparátu hrubých množin.) Parametr *OUTPUT_TYPE* informuje skript, v jakém formátu nasbíraná a naklasifikovaná data ukládat. Formát *rses* využijeme při analýze v programu RSES (Rough Set Exploration System), formát *plain* pak při analýze pomocí softwarových prostředků využívajících knihovny Rough Set Library psané v jazyce ANSI[29] C.

7.4 Sběr a klasifikace dat

Nyní připravíme datový model pro následné vygenerování pravidel. Tento model bude obsahovat jak normální síťový provoz, tak z hlediska našeho pohledu anomální. Normální provoz lze řešit různými postupy. Je třeba mít na paměti, že během konstrukce modelu (získávání dat a jejich klasifikace – tedy učení s učitelem) musí být systém kompletně odstíněn od "rušivých" vlivů. Zde se má na mysli situace, kdy provedeme nevědomě nesprávnou klasifikaci. Tato situace může nastat v momentě, kdy ohodnotíme získaný objekt jako normální, ačkoliv náš systém zaznamenal (bez našeho vědomí) útočné jednání v podobě např. skenování portů. Odstínění provedeme tím, že postavíme sběrný systém za firewall do prostředí, kde kvalitu sběru dat nemůže ohrozit útočné nebo obecně anomální chování. Pokud je zachováno ideální prostředí pro tuto činnost, je možné přistoupit k samotné klasifikaci. Tuto můžeme provádět buď **automaticky** (s předem přednastavenou hodnotou klasifikátoru) nebo **asistovaně**, kde na základě síťových znalostí zatřídíme objekt do skupiny normálního nebo anomálního síťového chování.

Pro zajištění přístupu k síťovému rozhraní je třeba získat práva administrátora systému (uživatel *root*), provedeme příkazem *su* a vstoupíme do adresáře se skripty:

```
vladimir@server:~$ su
Enter password: ************
root@server:# cd ~/RSES/
root@server:~/RSES/#
```

Nejprve nahlédněme na příklad **asistovaného ručního ohodnocování**. Spuštěním skriptu *collect_and_clasify.sh* s parametrem názvu souboru s výstupním datovým modelem budeme provádět ruční klasifikaci objektů. Příklad:

```
root@server:~/RSES/#./collect_and_clasify.sh model
Citac: 1
Charakteristika: 1:0:0
Klasifikuj [0,1,2,..], pro ukonceni napis "done"
```

Nyní je třeba provést klasifikaci získané charakteristiky. Toto lze provést spolehlivě tehdy, známe-li přesně síťový provoz (pokud jsme jej vyvolali my sami nebo jsme jej pozorovali pomocí nějakého nástroje pro analýzu síťového provozu a dokážeme rozhodnout o jeho bez-/vadnosti). V tomto případě jsou jednotlivé atributy odděleny dvojtečkou a znamenají v pořadí: počet navázání spojení (TCP/SYN), počet odmítnutých spojení (TCP/RST) a počet ICMP zpráv o nedosažitelném portu (ICMP/Port Unreachable). Tento síťový balíček označíme nulou, neb se nejedná o anomální typ provozu. Takto pokračujeme tak dlouho, dokud nepostihneme valnou většinu očekávaného síťového chování. Z důvodu

29 ANSI - American National Standards Institute

časové náročnosti a pracnosti tohoto postupu doporučuji využít automatické klasifikace.

Automatická klasifikace probíhá následujícím způsobem. Skript je uzavřen ve smyčce – získává data ze sítě a automaticky jim přiřazuje rozhodovací hodnotu (klasifikátor) stanovenou uživatelem. Tuto techniku doporučujeme použít, protože potenciálně umožňuje pokrýt širokou typologi síťového provozu. Doporučujeme též tento klasifikační postup použít po delší měřené období, např. po dobu jednoho dne. Získáme tím velké množství pravidel (s roustoucí saturací sítě získáme větší počet objektů). Množství objektů ve své podstatě má své opodstatnění v tom, že dokážeme postihnout široké spektrum provozu. Dosti pravděpodobně ale bude docházet k ekvivalenci tříd (nerozlišitelnost objektů) a před samotným vytvářením pravidel (rough rules) je dobré provést redukci dat (např. odstraněním duplicitních řádků prostřednictvím vlastního skriptu). Automatickou klasifikaci provedeme následovně:

```
root@server:~/RSES/#./collect_and_clasify.sh model auto 0
Citac: 1
Charakteristiky: 0:0:0
Klasifikuj [0,1,2,..], pro ukonceni napis "done"
1:0:0:0
Citac: 2
Charakteristiky: 0:0:0
Klasifikuj [0,1,2,..], pro ukonceni napis "done"
. . .
```

V příkladu nahoře jsme použili tři parametry: *model* (soubor, který bude obsahovat seznam objektů), *auto* (direktiva pro spuštění automatické klasifikace), *0* (klasifikujeme hodnotou nula). Výsledkem bude soubor s objekty, kde všechny budou klasifikovány nulou (můžeme interpretovat jako „není útočné/anomální chování"). Paralelně s během tohoto skriptu pak provedeme simulaci normálního síťového chování. Simulaci provedeme dle očekávaného chování na síti. Skript ukončíme stisknutím kombinace kláves CTRL+C. Takto máme sestavenu tu část modelu, která obsahuje jen objekty charakterizující normální síťové chování. Nyní je třeba vytvořit část popisující anomální chování. Toto provedeme změnou posledního parametru skriptu *collect_and_clasify.sh* a názvu výstupního souboru:

```
root@server:~/RSES/#./collect_and_clasify.sh model_2 auto 1
```

V tuto chvíli provedeme simulaci anomálního chování na síti. Tzn. že při běžném provozu budeme náhodně spouštět v různých intervalech a nastaveních aplikace, které generují nežádoucí datový tok (např. skenování portů, různé typy ICMP útoků a podobně). Je též dobré simulovat tento typ chování jak za běžného provozu, tak v případě „tiché" sítě (nezatížené žádným jiným provozem).

Máme-li obě části modelu vytvořené, syntetizujeme oba soubory do jednoho pomocí příkazu:

```
root@server:~/RSES/# cat model model_2 > model_complete
```

Tímto zajistíme spojení obou souborů v jeden soubor *model_complete*. Tento pak obsahuje objekty popisující normální síťové chování, tak anomální. Soubor tedy vypadá např. takto (tučně zvýrazněné číslice značí ohodnocení):

```
root@server:~/RSES/# cat model_complete.clf
2:0:0:19:0:0:0:4:0
3:0:0:18:0:0:0:8:0
2:0:0:12:0:0:0:9:0
. . .
1:0:0:21:0:0:0:29:1
1:0:0:23:0:0:0:28:1
0:0:0:26:0:0:0:28:1
```

Nyní provedeme zformátování souboru s objekty do tvaru, kterému bude rozumět některý z inferenčních nástrojů. V našem případě jsem použil (pro názornost) program RSES, proto je třeba v konfiguračním souboru *rough.conf* nastavit proměnnou *OUTPUT_TYPE* na *rses*.

Zformátování provedeme příkazem:

```
root@server:~#/root/RSES/rough_format.sh model_complete.clf
```

Výstupem bude soubor *model_complete.clf.sys*. Obsah tohoto souboru je již připraven ke zpracování programem RSES. Ukázka obsahu:

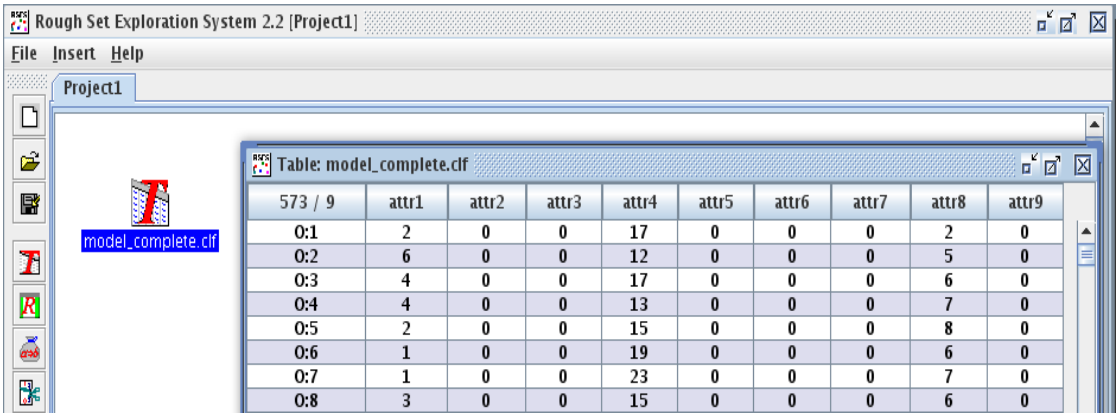
```
NAME: model_complete.clf
ATTRIBUTES: 9
OBJECTS: 507
2 0 0 19 0 0 0 4 0
3 0 0 18 0 0 0 8 0
2 0 0 12 0 0 0 9 0
. . .
2 0 0 16 0 0 1 30 1
1 0 0 21 0 0 0 29 1
0 0 0 28 0 0 0 27 1
```

7.5 Sestavení pravidel pomocí RSES

Získaná data v předchozí kapitole nyní využijeme pro vytvoření pravidel teorie hrubých množin. Tato pravidla následně budou využita pro klasifikaci předkládaného síťového provozu.

V prvých krocích je třeba naimportovat soubor objektů (informační systém) do aplikace RSES:

1. Spustíme aplikaci RSES a vytvoříme prázdný projekt.
2. Informační systém je reprezentován v teorii hrubých množin jako tabulka. Klikneme proto na ikonu reprezentující přidání tabulky, případně volíme akci Vložení tabulky (Insert table) z menu, které se zobrazí po stisknutí pravého tlačítka myši v prázdné oblasti projektu. Tím se na ploše projektu zobrazí ikona s popisem NewTable1.
3. Naimportujeme data (soubor s objekty) do tabulky tak, že klikneme pravým tlačítkem na nově vytvořenou ikonu NewTable1 a v menu vybereme Load (načíst).
4. Otevře se okno se soubory v adresáři, ze kterého byla aplikace RSES spuštěna. Vybereme soubor s daty (*model_complete.tab*). Soubor se načte do aplikace RSES.

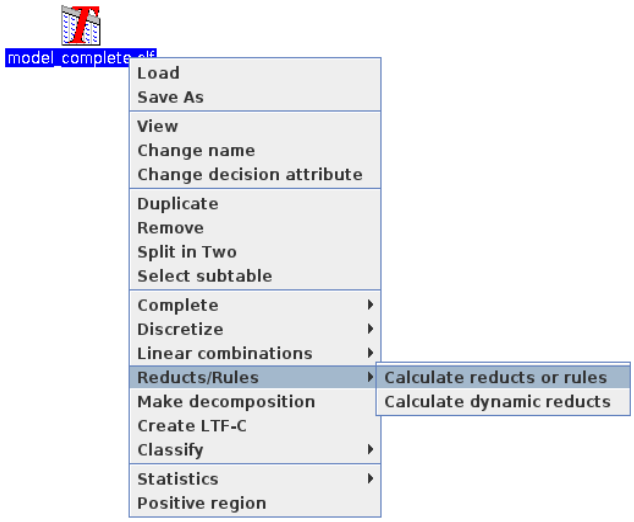


573 / 9	attr1	attr2	attr3	attr4	attr5	attr6	attr7	attr8	attr9
0:1	2	0	0	17	0	0	0	2	0
0:2	6	0	0	12	0	0	0	5	0
0:3	4	0	0	17	0	0	0	6	0
0:4	4	0	0	13	0	0	0	7	0
0:5	2	0	0	15	0	0	0	8	0
0:6	1	0	0	19	0	0	0	6	0
0:7	1	0	0	23	0	0	0	7	0
0:8	3	0	0	15	0	0	0	6	0

Obrázek 12: Naimportovaný informační systém do RSES [Zdroj: vlastní]

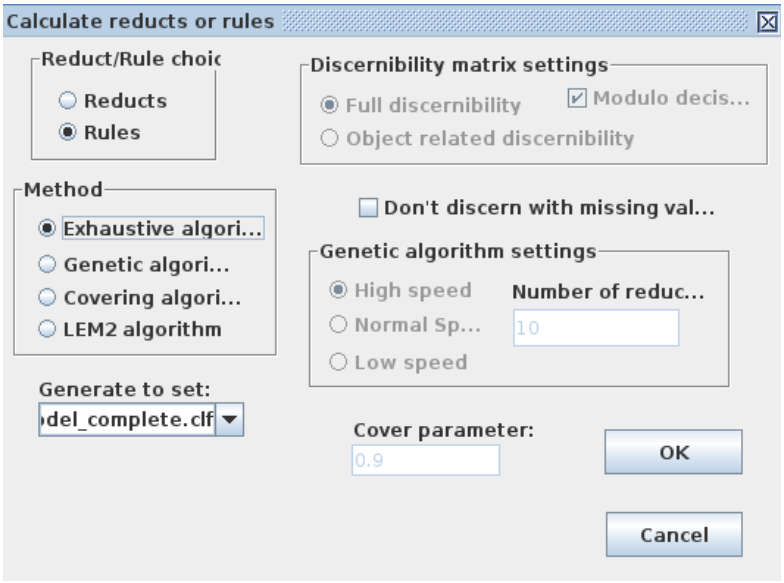
Data tabulky (nyní nově automaticky pojmenované *model_complete.clf*) lze zobrazit dvojím poklikáním na ikonu s naimportovaným informačním systémem. V našem demonstrativním případě naimportovaný model obsahuje 573 objektů a 9 měřených atributů, je patrné z obrázku níže:

V dalším fázi je nyní možno vygenerovat soubor pravidel z tohoto naimportovaného systému. Tuto operaci provedeme tak, že stiskem pravého tlačítka na ikoně tabulky *model_complete.clf* vyvoláme následující nabídku:



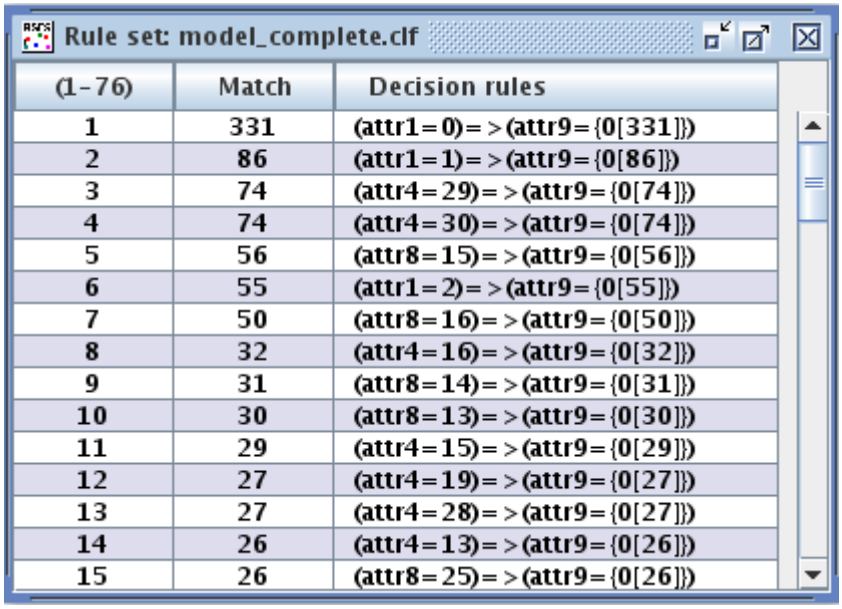
Obrázek 13: Nabídka pro manipulaci s tabulkou [Zdroj: vlastní]

Dále vybereme *Reducts/Rules* a v podnabídce *Calculate reducts or rules*. Před samotným procesem generování pravidel je třeba provést přednastavení, kde je nutné definovat metodu pro genezi pravidel a zároveň množinu (informační systém), ze které se pravidla následně vyextrahují.



Obrázek 14: Příprava pro generování pravidel síťového modelu [Zdroj: vlastní]

V tomto případě jsme zvolili Exhaustive algoritmus (popsán v kapitole 7.1). Dále stiskem tlačítka „OK“ provedeme vygenerování pravidel. Výsledkem je nová ikona připojena k tabulce s daty reprezentující soubor pravidel. Analogickým poklikáním levého tlačítka myši je možné zobrazit pravidla:



(1-76)	Match	Decision rules
1	331	(attr1=0)=>(attr9={0[331]})
2	86	(attr1=1)=>(attr9={0[86]})
3	74	(attr4=29)=>(attr9={0[74]})
4	74	(attr4=30)=>(attr9={0[74]})
5	56	(attr8=15)=>(attr9={0[56]})
6	55	(attr1=2)=>(attr9={0[55]})
7	50	(attr8=16)=>(attr9={0[50]})
8	32	(attr4=16)=>(attr9={0[32]})
9	31	(attr8=14)=>(attr9={0[31]})
10	30	(attr8=13)=>(attr9={0[30]})
11	29	(attr4=15)=>(attr9={0[29]})
12	27	(attr4=19)=>(attr9={0[27]})
13	27	(attr4=28)=>(attr9={0[27]})
14	26	(attr4=13)=>(attr9={0[26]})
15	26	(attr8=25)=>(attr9={0[26]})

Obrázek 15: Vygenerovaná pravidla z tabulky model_complete.clf [Zdroj: vlastní]

Z výše uvedeného snímku (Obrázek 15) je patrno, že z množiny dat bylo sestaveno 76 pravidel.

7.6 Testování modelu v RSES

V kapitole 7.5 byl vytvořen model pravidel chování systému, který popisuje jak normální tak anomální provoz v síti. Anomální provoz byl řešen sekvenčním spuštěním programu Nmap (více o Nmap v kapitole 3.4) s několika parametry (-sS, -sF, -sU a bez parametru)

proti vybranému vzdálenému serveru. Normální provoz byl vytvořen za pomoci skriptu *http_traffic.sh* (více o skriptu v kapitole 6.6) a dodatečného ručního prohlížení náhodných webových stránek. Ověření modelu lze provést tak, že získáme vzorek dat ze sítě a ten předložíme programu RSES k evaluaci. Vzorek dat (jeden objekt nebo více objektů) je třeba pořídit se stejnými parametry (parametry uvedené v konfiguraci *rough.conf*, zejména velikost *PACKAGE_SIZE*), jakými byl vytvořen model, v jiném případě dostaneme špatné výsledky.

Otestovat validitu pravidel lze poměrně snadno aplikací popsaných skriptů v kapitolách 6.1 až 6.4. Ukázka sběru vzorku k otestování:

```
root@server:/tmp# collect_and_clasify.sh test auto 0
Citac: 1
Charakteristiky: 3:0:0:8:0:0:0:1
Klasifikuj [0,1,2,..], pro ukonceni napis "done"
3:0:0:8:0:0:0:1:0
Citac: 2
Charakteristiky: 0:0:0:9:0:0:0:0
Klasifikuj [0,1,2,..], pro ukonceni napis "done"
0:0:0:9:0:0:0:0:0
Citac: 3
. . .
```

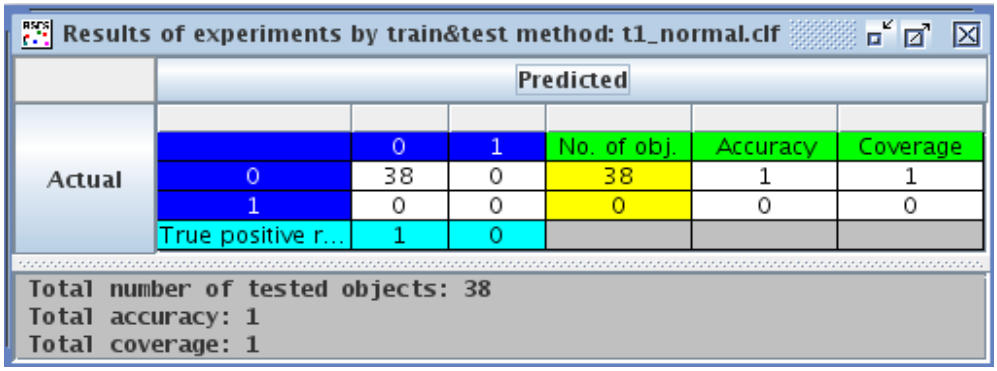
Ve výše uvedeném příkladu jsme zavolali skript *collect_and_clasify.sh* s parametrem výstupního souboru *test.clf* (přípona *.clf* informuje, že obsahuje oklasifikovaná data) a nechali jsme jej automaticky klasifikovat veškerý nasnímaný provoz hodnotou 0 (parametr *auto 0*). Skript ukončíme stisknutím CTRL+C a provedeme přeformátování dat do podoby srozumitelné programu RSES:

```
root@server:/tmp# /root/RSES/rough_format.sh test.clf
```

Výstupem bude soubor *test.clf.sys*, který je již připraven pro import do RSES v podobě testovací množiny dat.

7.6.1 Normálního provoz

V této části předkládáme aparátu hrubých množin soubor dat *t1_normal*, který obsahuje objekty, které reprezentují běžný provoz prohlížeče.



Obrázek 16: Klasifikace normálního síťového provozu [Zdroj: vlastní]

Výsledkem je, že z 38-mi objektů bylo 38 klasifikováno jako normální síťový provoz.

7.6.2 Explicitní anomálie

Testování anomálního provozu provedeme obdobně jako v předchozím příkladu s tím rozdílem, že snímaný provoz bude obsahovat nežádoucí skenování portů.

Results of experiments by train&test method: t3_synscan.clf

		Predicted				
		0	1	No. of obj.	Accuracy	Coverage
Actual	0	0	20	20	0	1
	1	0	0	0	0	0
	True positive r...	0	0			

Total number of tested objects: 20
Total accuracy: 0
Total coverage: 1

Obrázek 17: Ukázka klasifikace skenování portů [Zdroj: vlastní]

Skenování portů bylo součástí trénovací množiny dat *model_complete.clf.sys*, proto námi vegenerovaná pravidla zaznamenala nežádoucí chování ve všech případech a ani v jednom objektu neprovedla chybnou klasifikaci (žádný objekt nebyl označen za normální, všech 20 je pokládáno za anomální). **Důležitá poznámka**: Může být matoucí, že v diagramu zobrazující výsledky je uvedena celková přesnost klasifikace (Total accuracy) nulová. Důvodem je, že při sběru dat ze sítě pomocí skriptu *collect_and_clasify.sh* jsme získané vzorky automaticky klasifikovali hodnotou 0 (normální provoz), ačkoliv jsme záměrně na pozadí spouštěli skenování portů.

7.6.3 Kombinovaný provoz

V následujícím případě jsme nasbírali 12 objektů, kde jedna část snímku tvoří anomální a druhá normální síťové chování.

Results of experiments by train&test method: t4_combi1.clf

		Predicted				
		0	1	No. of obj.	Accuracy	Coverage
Actual	0	10	2	12	0.833	1
	1	0	0	0	0	0
	True positive r...	1	0			

Total number of tested objects: 12
Total accuracy: 0.833
Total coverage: 1

Obrázek 18: Kombinované síťové chování [Zdroj: vlastní]

Oba druhy síťového provozu jsme generovali současně. Tzn., že jsme provedli načtení náhodných webových stránek a posléze vygenerovali provoz typu SYN sken pomocí programu Nmap.

7.7 Výsledky algoritmu LEM2

Pro otestování algoritmu LEM2 jsme založili v RSES nový projekt, naimportovali informační systém a vygenerovali pravidla algoritmem LEM2 (poměr pokrytí 0.9). Informační systém je identický jako v případě testování Exhaustive algoritmu v kapitolách 7.6.1 až 7.6.3. Testovací množiny jsou též shodné.

Rule set: model_complete.clf

(1-8)	Match	Decision rules
1	321	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=0)=>(attr9={0[321]})
2	75	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=1)=>(attr9={0[75]})
3	47	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=2)=>(attr9={0[47]})
4	22	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=3)=>(attr9={0[22]})
5	20	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr4=0)=>(attr9={1[20]})
6	18	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=4)=>(attr9={0[18]})
7	12	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=6)=>(attr9={0[12]})
8	12	(attr3=0)&(attr5=0)&(attr6=0)&(attr7=0)&(attr2=0)&(attr1=5)=>(attr9={0[12]})

Obrázek 19: Pravidla vygenerovaná pomocí algoritmu LEM2 [Zdroj: vlastní]

Výsledky algoritmu LEM2 jsou identické s tím rozdílem, že LEM2 algoritmus svou podstatou nepokrývá celý prostor objetů (tak jako Exhaustive), ale vždy jen nějakou část (stanovenou při generování pravidel stupněm pokrytí). To s sebou nese neporkytí veškerých objektů – některé objekty nejsou zahrnuty do pravidel (např. víceznačné). V následujícím obrázku (Obrázek 20) je klasifikace normálního provozu. Je patrné, že klasifikátor byl schopen rozpoznat všechny objekty jako normální a ani jeden stanovit jako anomální:

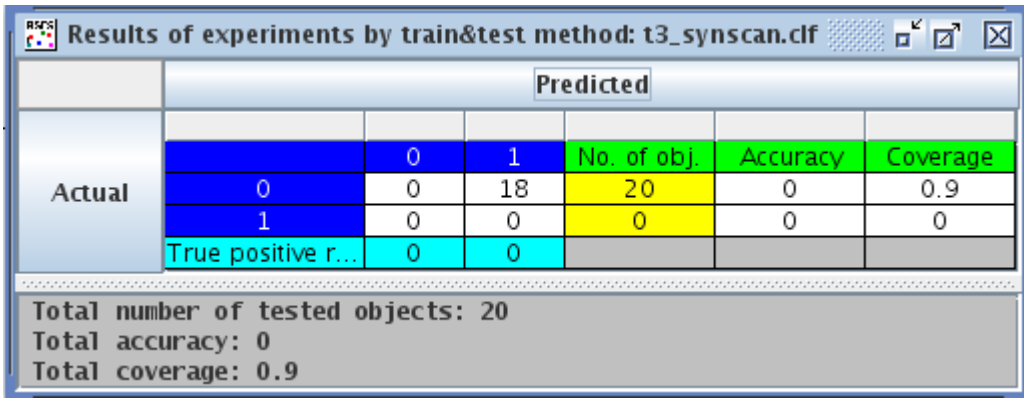
Results of experiments by train&test method: t1_normal.clf

		Predicted				
		0	1	No. of obj.	Accuracy	Coverage
Actual	0	36	0	38	1	0.947
	1	0	0	0	0	0
	True positive r...	1	0			

Total number of tested objects: 38
Total accuracy: 1
Total coverage: 0.947

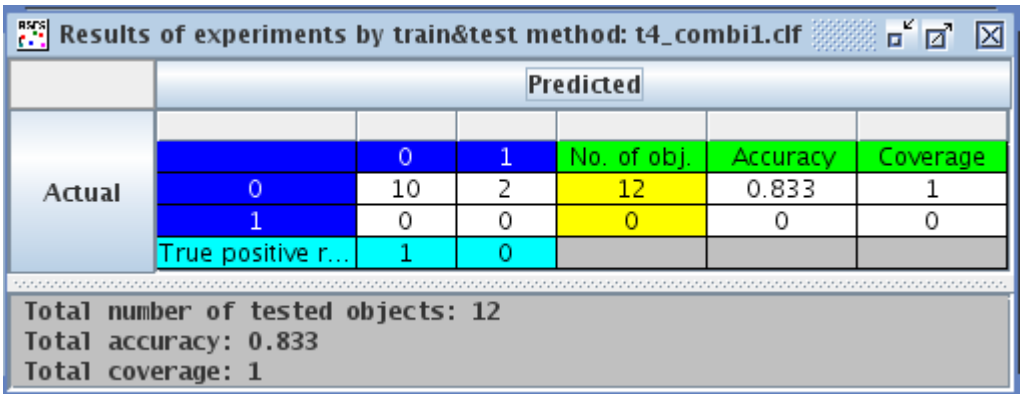
Obrázek 20: Normální síťový provoz nad pravidly LEM2 [Zdroj: vlastní]

V následujícím experimentu jsme použili testovací datovou množinu, která obsahovala všechny objekty, které jsou součástí vertikálního SYN skenu a výsledek je opět naprosto přesný. Ani jeden objekt není klasifikován jako normální, všechny ostatní (počet 18) patří do skupiny anomálních:



Obrázek 21: Skenování portů nad pravidly LEM2 [Zdroj: vlastní]

V posledním testování jsme použili datovou množinu s kombinovaným provozem: běžný webový provoz ukončený SYN skenem:



Obrázek 22: Kombinovaný síťový provoz nad LEM2 [Zdroj: vlastní]

Dva objekty byly rozpoznány jako anomálie, ostatních deset bylo klasifikováno jako normální síťové chování.

8 Závěr

Cílem práce bylo zhodnotit možnost využití teorie hrubých množin pro detekci síťových anomálií. Aparát hrubých množin se ukázal jako velice vhodný pro tuto problematiku z hlediska jednoduchosti implementace a schopnosti úspěšně rozpoznávat odchylky od definovaného důvěryhodnostního modelu.

Navrhl jsem **paketově orientovaný** způsob klasifikace síťového provozu a navrhl jsem algoritmus, který je schopen ze skupiny paketů vytvořit objekt. Tento objekt je určen ke klasifikaci. Vytvořil jsem důvěryhodnostní model síťového provozu, který se skládal z objektů normálního a anomálního síťového provozu. Tento model pokrýval normální síťové chování, tedy: sledování webových stránek, vyhledávání na Internetu, práce s elektronickou poštou. Dále jsem do modelu zařadil anomálie definované několika druhy skenování portů a útokem SYN Flood. Z těchto objektů jsem pak sestavil pomocí vybraného algoritmu sadu pravidel popisující oba dva druhy provozu. Následovalo testování pravidel. Probíhalo tak, že jsem programu RSES sekvenčně předkládal jednu množinu nasbíraných testovacích vzorků za druhou. Program RSES pak aplikoval sestavená pravidla (důvěryhodnostní model) na tyto množiny. Výsledkem bylo zařazení předkládaných objektů do dvou tříd: normální nebo anomální. Klasifikátor programu RSES ani v jedné z množin objektů neprovedl chybné zařazení. Důvod těchto dobrých výsledků spatřuji v kvalitním vypracování pravidel, která popisují veškerý očekávaný druh provozu. Je nutné zmínit, že mnou navržený způsob diskriminace vzorků (normální, anomální) se ukázal jako velmi vhodný a účinný. Výsledek je takový, že dokážeme na základě důvěryhodnostního modelu jednotlivé druhy síťového provozu přesně rozlišit. Máme též vyřešenou situaci, kdy normální a anomální chování běží současně. V takovém případě mohou nastat při klasifikaci dvě situace: objekt bude klasifikován jako anomální nebo jako „nezařaditelný“. Všechny objekty, které jsou tzv. nezařaditelné, je třeba přezkoumat (*a priory* vnímat jako anomální). Takový objekt je po posouzení (operátorem, administrátorem sítě) zařazen do skupiny normálních objektů nebo anomálních a pravidla jsou přegenerována. Pravděpodobnost výskytu nezařaditelných objektů jsem se snažil snížit na minimum tím, že jsem upravoval velikost (počet datagramů) vzorku. Příliš široký vzorek pak s sebou nese větší preciznost v klasifikaci, ale pokud není úměrně rozšířen pravidlový prostor, který postihne širší spektrum síťového chování, má potenciálně za následek zvýšený počet „falešných poplachů“ (false positive alerts).

Vhodnost nasazení tohoto systému pak obzvláště spatřuji na koncové stanice, kde je síťový provoz do jisté míry predikovatelný, kde jsme skrze pravidla hrubých množin schopni síťové chování popsat. Nepochybně by bylo možné tento preventivní systém nasadit na pracovní stanice v bankách a v dalších institucích, kde se využívá omezený počet programových prostředků - kde je síťové chování relativně snadné popsat. Mnou navržený algoritmus diskriminace objektů nepracuje s časem, ale s počtem paketů ve vzorku. Vyhnul jsme se tím možnosti špatné klasifikace z důvodu rozdílného stylu práce uživatele.

Dovedu si představit též nasazení na síťové servery s tím, že by bylo třeba upravit algoritmus sestavování objektů a též zároveň množinu atributů, které diskriminují provoz. V tomto případě by se (patrně) více hodil tokově orientovaný způsob hodnocení (analýza protokolů sFlow, NetFlow apod.). Nevylučuji ovšem, že paketově orientovaný je též nasaditelný.

Použitá literatura

[1] Dostálek L., Kabelová A.: Velký průvodce protokoly TCP/IP a systémem DNS. Computer Press, 2000.

[2] IP Protokol [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.cpress.cz/knihy/tcp-ip-bezp/CD-0x/5.html>.

[3] Kállay F., Peniak P.: Počítačové sítě a jejich aplikace LAN/MAN/WAN. Grada Publishing a.s., 2003.

[4] Protokol TCP [online]. [cit. 2009-08-06]. Dostupné z WWW: <Zdroj:http://www.cpress.cz/knihy/tcp-ip-bezp/CD-0x/9.html>.

[5] Protokol UDP [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.cpress.cz/knihy/tcp-ip-bezp/CD-1x/10.html>.

[6] Protokol ICMP [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/ICMP>.

[7] Internet Control Message Protocol [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.ietf.org/rfc/rfc792.txt>.

[8] ICMP Type Numbers [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.iana.org/assignments/icmp-parameters>.

[9] IDS - Intrucion Detection System na VIC ČVUT [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://ids.vc.cvut.cz/info.php>.

[10] Šifry [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.karlin.mff.cuni.cz/~tuma/ciphers08/sifry5.ppt>.

[11] Bezpečnost IS/IT [online]. [cit. 2009-08-06]. Dostupné z WWW: <https://akela.mendelu.cz/~lidak/share/snimky-bis/prednaska7.ppt>.

[12] Characteristics of Network Traffic Flow Anomalies [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.imconf.net/imw-2001/imw2001-papers/47.pdf>.

[13] Rozšíření NetFlow kolektoru NfSen o detekci síťových anomálií [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://is.muni.cz/th/72577/fi_m/xelich-dp.pdf>.

[14] Bezpečnost IS/IT [online]. [cit. 2009-08-06]. Dostupné z WWW: <https://akela.mendelu.cz/~lidak/bis/7tech.htm>.

[15] TOP 5 Intrusion Detection Systems [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://sectools.org/ids.html>.

[16] Snort Users Manual [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.snort.org/assets/82/snort_manual.pdf>.

[17] NetFlow Overview [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.cisco.com/en/US/docs/ios/12_1/switch/configuration/guide/xcdnfov.html>.

[18] Architecture for IP Flow Information Export [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://tools.ietf.org/html/rfc5470>.

[19] Packet Sampling Basics [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.sflow.org/packetSamplingBasics/index.htm>.

[20] sFlow version 5 [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.sflow.org/sflow_version_5.txt>.

[21] From NetFlow to IPFIX (The evolution of IP flow information export) [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.cert.org/netsa/publications/nanog41-ipfix.pdfhttp://www.cert.org/netsa/publications/nanog41-ipfix.pdf>.

[22] Traffic Monitoring using sFlow [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.sflow.org/sFlowOverview.pdf>.

[23] NetFlow [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Netflow>.

[24] Cisco Systems NetFlow Services Export Version 9 [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://www.faqs.org/rfcs/rfc3954.html>.

[25] Detection and Characterization of Port Scan Attacks [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://cseweb.ucsd.edu/~clbailey/PortScans.pdf>.

[26] Stroje a množiny Zdzislawa Pawlaka [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://math.fce.vutbr.cz/~pribyl/workshop_2006/prispevky/Novotny.pdf>.

[27] Rough Sets: A tutorial [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://folli.loria.fr/cds/1999/library/pdf/skowron.pdf>.

[28] Vaníček J., Papík M., Pergl R., Vaníček T.: Teoretické základy informatiky. Kernberg Publishing, s.r.o., 2007.

[29] Zpracování dat v prostředí GIS s využitím teorie Rough set [online]. [cit. 2009-08-06]. Dostupné z WWW:

<http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2000/Sbornik/Smutny/Referat.htm>.

[30] Petr P.: Data Mining Díl I.. Univerzita Pardubice, 2008.

[31] Algoritmy a datové struktury [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://wiki.matfyz.cz/wiki/Bakalářská_státnice_-_Informatika_-_Základy_informatiky_-_ISPS_-_Algoritmy_a_datové_struktury>.

[32] Meloun M., Militký J., Hill M.: Počítačová analýza vícerozměrných dat v příkladech. Academia, 2005.

[33] Souček E.: Základy pravděpodobnosti a statistiky. Univerzita Pardubice, 2004.

[34] Using Rough Sets and Genetic Algorithm to Build Decision Tree [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://infos2007.fci.cu.edu.eg/Computational%20Intelligence/07172.pdf>.

[35] Covering Algorithms, continuum percolation and the geometry of wireless networks [online]. [cit. 2009-08-06]. Dostupné z WWW: <www.cs.vu.nl/~rmeester/preprints/covering.ps>.

[36] A Comparison of Three Strategies to Rule Induction from Data with Numerical Attributes [online]. [cit. 2009-08-06]. Dostupné z WWW: < http://lightning.eecs.ku.edu/c82-rskd.pdf>.

[37] RSES 2.2 User's Guide [online]. [cit. 2009-08-06]. Dostupné z WWW: <http://logic.mimuw.edu.pl/~rses/RSES_doc_eng.pdf>.

Seznam užitých zkratek

- ANSI - American National Standards Institute
- ATM - Asynchronous Transfer Mode
- BPF - Berkley Packet Filter
- BSD - Berkley Software Distribution
- DDoS - Distributed Denial of Service
- DoS - Denial of Service
- GUI - Graphical User Interface
- HTTP - HyperText Transfer Protol
- ICMP - Internet Control Message Protocol
- IETF - Internet Engineering Task Force
- IDS – Intrusion Detection System
- IP – Internet Protcol
- ISP – Internet Service Provider
- NBAD – Network Behavior Analysis Detection
- NIDS - Network Intrusion Detection System
- OS – Operating System
- PPP – Point-to-Point
- RSES - Rough Set Exploration system
- SCTP - Stream Control Transmission Protocol
- SMTP - Simple Message Transfer Protocol
- TCP - Transmission Control Protocol
- UDP - User Datagram Protocol

Seznam ilustrací

Seznam tabulek