

UNIVERZITA HRADEC KRÁLOVÉ
FAKULTA INFORMATIKY A MANAGEMENTU

Katedra informačních technologií
Obor: informační management

**Detekce průniků užitím
inteligentního systému
pro podporu rozhodování**

SEMESTRÁLNÍ PRÁCE

Autor: Bc. Josef Kadlec
Obor: IM2-K (1. ročník)
Předmět: SPR
Vyučující: Ing. Karel Mls

Rok 2004

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně a uvedl jsem veškerou použitou literaturu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským.

Je dovoleno kopírovat, šířit a/nebo modifikovat tento dokument za podmínek licence GNU FDL, verze 1.2 nebo vyšších publikovaných nadací Free Software Foundation. Toto dovození je platné pouze pro státy, kde obsah díla daný jeho názvem není v rozporu se zákonem.

Copyright © 2004 Josef Kadlec

V Hradci Králové dne 15. prosince 2004

Josef Kadlec

Anotace

Tato semestrální práce popisuje jakýsi návrh genetického třídícího IDS (Intrusion Detection System), kterým lze aktivně detekovat a automaticky reagovat na incidenty. Tento systém je zkonstruován tak, aby byl schopen monitorovat různé aktivity na síti - hledá změny jako poruchy, chyby, abnormality, zneužití, odchylky, průniky, apod. Počítače jsou monitorovány na více úrovních současně. Mezi tyto úrovně patří uživatelská úroveň, systémová úroveň, úroveň procesů a úroveň paketů, seřazeno od té nejvyšší. Rozhodnutí o akci v případě bezpečnostního incidentu má na starosti genetický (evoluční) třídící systém. Cílem je najít korelaci mezi odchylkami hodnot měřených parametrů od normálního stavu a tím zjistit typ průniku. A následně adekvátně zareagovat. Byly prováděny experimenty na platformě Unix za účelem vytvoření perspektivních rozhodovacích pravidel na základě jednotlivých monitorovaných parametrů.

Obsah

1 Úvod	1
2 Monitorování a filtrování dat	3
2.1 Monitorování parametrů	3
2.2 Stanovení prahových hodnot	4
3 Vytváření modulu pro podporu rozhodování	6
3.1 Vytváření úrovně heuristiky	6
4 Prvotní experimenty a výsledky	13
5 Závěr	15
Literatura	i

Kapitola 1

Úvod

Problém detekce anomálií, průniků a jiných forem počítačového zneužívání může být zobecněn na hledání nepřipustných odchylek (nebo bezpečnostních narušení) s charakteristickými hodnotami měřenými na monitorovaných systémech nebo sítích. Předpokládá se tedy, že aktivity, které vyvolává narušitel se nějak liší od aktivit běžného užívání. Avšak v mnoha případech je velmi obtížné v souvislosti s nějakým bezpečnostním incidentem nějaké rozdíly detekovat.

Abychom mohli správně a efektivně reagovat na incident, je potřeba rozpoznat, co se přesně stalo [3]. Způsobů, jak detekovat útoky bylo nalezeno mnoho [8] [10] [11] [12] a je k dispozici mnoho komerčních IDS (například NetRanger, RealSecure, Omniguard Intruder) i nekomerčních IDS (například Snort, LIDS, Grsecurity, Prelude). Častý model detekce útoků je založen na jakýchsi signaturách. IDS pak vytváří signatury akcí, které se na systému či síti dějí a porovnává je s databází signatur útoků. V případě nalezení shody se přistupuje k dalším krokům v rámci reakce na incidenty. Nevýhodou takového modelu je především to, že databáze signatur musí být pravidelně aktualizována a také samozřejmě to, že takové IDS bude účinné pouze proti známým útokům. V případě, že budou použity nějaké nové metody průniku, bude takové IDS bezmocné. Ovšem každý z modelů detekce nežádoucí činnosti má své silné i slabé stránky. K detekci koordinovaných a sofistikovaných útoků začali vědci využívat umělou inteligenci [6], genetických přístupů [4] [8] a tzv. architektury využívající agenty (angl. agent architectures) [7].

Tato semestrální práce popisuje návrh a implementaci komponenty pro podporu rozhodování založené na třídění k IDS (Intrusion Detection System). Třídící IDS monitoruje aktivity Unixového systému na několika úrov-

ních (od úrovně paketů k uživatelské úrovni) a snaží se najít korelaci mezi odchylkami hodnot měřených parametrů od normálního stavu. Například na uživatelské úrovni se může jednat o neobvyklé uživatelské chování. Na systémové úrovni například vytíženost CPU, paměti, I/O operací, atd. Na úrovni procesů se může jednat o neplatné a neověřené procesy. Na úrovni paketů je pak monitorován především počet, objem a velikost takových paketů souběžně s jejich zdrojovou adresou a typem spojení.

Pro měření těchto parametrů jsou vhodné unixové aplikace typu vmstat, iostat, netstat, mpstat, a další [11]. Po zachycení neobvyklé aktivity jsou informace o tomto incidentu posílány do třídícího (informace jsou rozřazovány do tříd podle charakteru incidentu nebo jeho nebezpečnosti) systému [9], který má za úkol rozhodnout o patřičných aktivitách (bezprostřední reakci na incident).

Kapitola 2

Monitorování a filtrování dat

Detekce průniků a anomálií obecně vyžaduje pravidelné monitorování zkoumaných parametrů (jak síťových tak systémových) a to kvůli sběru informací o normálním chování systému. Při detekci abnormalit jsou některé parametry jsou označeny jako klíčové. Celý problém je založen na tom, že detekování bezpečnostních incidentů je postaveno na hypotéze, že tyto incidenty je možné odhalit zaznamenáním neobvyklých (abnormálních) charakteristik chování systému.

2.1 Monitorování parametrů

Zde popisovaný návrh zpracovává na uživatelské úrovni tyto parametry (označení U1 až Un je záměrné a to z důvodu snazšího zápisu složitějších vazeb):

- U1: Typ uživatele a uživatelská privilegia
- U2: Intervaly přihlašování a odhlašování
- U3: Přístup k souborům a adresářům
- U4: Typy používaného softwaru
- U5: Použité příkazy

Uživatelské parametry jsou sbírány na základě id, které započalo relaci. Takže údaje o uživatelských kontech jsou separovány na jednotlivé relace a to z důvodu, že je pravděpodobné, že se narušitel bude maskovat jako jeden z důvěryhodných uživatelů.

Mezi parametry na systémové úrovni jsou zde řazeny:

- S1: Souhrnné vytížení procesoru jednotlivých uživatelů
- S2: Využití fyzické a virtuální paměti
- S3: Množství volné SWAP paměti
- S4: Množství volné paměti
- S5: Využití I/O a disku

Do skupiny parametrů na úrovni procesů zde řadíme:

- P1: Počet procesů a jejich typy
- P2: Vazba mezi systémy
- P3: Čas od spuštění procesu
- P4: Současný stav procesu (běžící, zastavený, čekající) a ukončené procesy
- P5: Podíl různých časů procesů - například uživatelský čas procesu, systémový čas procesu, nevyužitý čas procesu

Nakonec parametry na úrovni sítě (nebo také paketová úroveň):

- N1: Počet procesů a jejich stav
- N2: Průměrný počet odeslaných a přijatých paketů
- N3: Délka trvání spojení
- N4: Typ připojení (vzdálené, či lokální)
- N5: Použitý protokol a port

To jsou tedy parametry, jejichž hodnoty jsou zaznamenávány a statisticky analyzovány za účelem rozeznání abnormálních aktivit a hypoteticky tím pádem i detekce bezpečnostního incidentu. Zvolení parametrů může být samozřejmě v závislosti na individuální implementaci jiné. Toto rozdělení např. zvolili Dipankar Dasgupta a Fabio A. Gonzalez. Tomáš Pluskal ve své práci [2] zvolil a ověřil trochu jiné parametry.

2.2 Stanovení prahových hodnot

Jak již bylo řečeno, data jednotlivých měřených parametrů jsou pravidelně monitorována po určitou dobu, kdy se systém chová "normálně" (tzn. nedochází k žádným incidentům) za účelem vytvoření jakéhosi statistického modelu tohoto normálního stavu systému. Současně s tím jsou stanoveny jakési prahové hodnoty (limity nebo hranice), které jsou pro různé parame-

desítkově	binárně	slovně
0	00	normální stav
1	01	minimální odchylka
2	10	významná odchylka
3	11	nebezpečná odchylka

Tabulka 2.1: Stupně odchylky od normálu.

try rozdílné. Během monitorování jsou pak aktuální hodnoty jednotlivých parametrů porovnávány s hodnotami modelu normálního chování.

Stanovení prahových hodnot tak, aby byly využitelné k rozeznání útoku je velmi choulostivá záležitost, která vyžaduje pečlivé ladění. Detekce systému by neměla vyvolávat falešný poplach kvůli triviálním jevům, které mohou být detekovány jako abnormální, ale zase by neměla přehlížet reálné možnosti skutečného útoku. Navíc tyto limity musí být aktualizovány v závislosti na legitimních změnách daného systému.

Abychom mohli docílit seriózní detekce průniků, je potřeba přidělit parametrům různé priority - viz. tabulka 2.1.

Kapitola 3

Vytváření modulu pro podporu rozhodování

Je velmi obtížné vytvořit takovou inteligentní komponentu pro podporu rozhodování, kvůli tomu, že celý problém je velmi neurčitý a mnoha smyslný. Řešením může být vytvoření takového systému, který se dokáže adaptovat podle prostředí. Třídící systém je adaptivní samo učící se systém, který dokáže vytvořit sadu pravidel na základě zkoumaného prostředí. Tato pravidla jsou naprogramována jako řetězce o pevně dané délce a jsou vyvinuty užitím genetického prohledávání. Tyto třídiče (vlastně řetězce) jsou vyvinuty na základě bezpečnostní politiky [3]. Ovšem když to převedu do prostředí této práce, tak je bezpečnostní politika vytvořena v době definování "normálního" chování systému. V řetězci (třídíči) představuje jedna část (ta podmínková) stupeň odchylky od normálu (jak již víme, tak může nabývat hodnot 0 až 3) a druhá část (ta reakční) představuje již určitou reakci na incident (ta zde bude nabývat hodnot 0 až 7).

3.1 Vytváření úrovnové heuristiky

Stupeň důležitosti každé úrovně (resp. monitorovaných parametrů) je založen na hypoteticky založen na znalostech a zkušenostech této oblasti. Naším záměrem bude generovat pravidla na základě obecných znalostí expertů této oblasti nebo vlastních zkušeností, které jsme získali zkoumáním problematiky. Toto by nám mělo zajistit akce, které budou nejlépe odpovídat realitě. Samozřejmě si můžeme vytvořit vlastní heuristiku a zajistit si podobné výsledky. Tabulka 3.1 ukazuje příklady takových pravidel.

hypotéza č.	uživ. úr.	syst. úr.	úr. procesů	paketová úr.	reakce
1	0.2	0.0	0.0	0.1	1
2	0.4	0.0	0.0	0.4	3
3	0.0	0.1	0.2	0.8	6

Tabulka 3.1: Váhy jednotlivých parametrů přidělené při nežádoucí činnosti a přidělené reakce.

Matematickou symbolikou vyjádřeno je každá hypotéza n-tice (v našem případě 5ice)

$$(k_1, k_2, k_3, k_4, a)$$

, kde $k_i \in [0.0, 1.0]$ a $a \in \{0, 1, 2, \dots, 7\}$.

Za předpokladu, že vstup pro třídič dostane tzv. zprávu

$$(m_1, m_2, m_3, m_4)$$

, tak tato zpráva vyhovuje hypotéze pouze v případě, že

$$k_1 \leq m_1, k_2 \leq m_2, k_3 \leq m_3, k_4 \leq m_4.$$

To představuje nejnižší hodnoty odchylek v různých úrovních jakožto podmínkovou část výrazu dané hypotézy. Současně pokud máme dvě hypotézy

$$K = (k_1, k_2, k_3, k_4, a) \text{ a } L = (l_1, l_2, l_3, l_4, a)$$

a platí

$$k_i \leq l_i, \quad i = 1, 2, 3, 4.$$

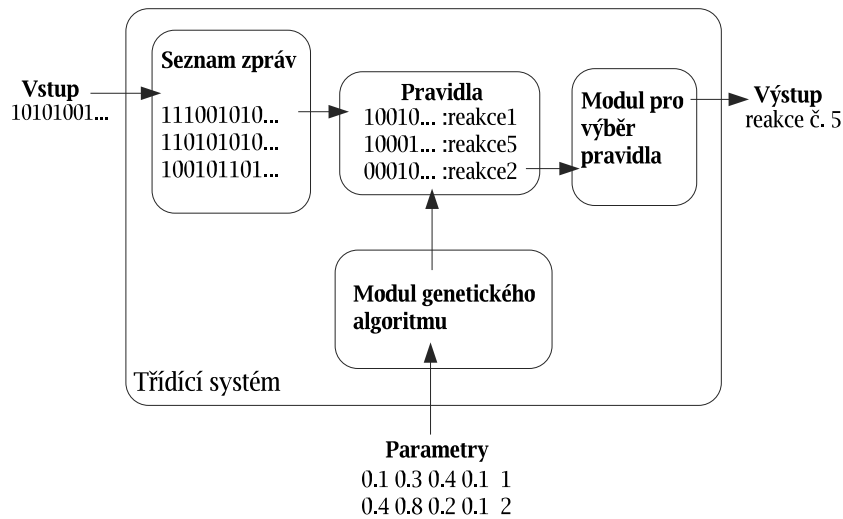
To znamená že hypotéza K je obecnější (nadřazená) než hypotéza L, potom tedy jakákoliv zpráva, která souhlasí s L, bude souhlasit i s K. V rámci zachování konzistence poznatků bude akce hypotézy $L(a_l)$ silnější než hypotézy $K(a_k) : a_l \geq a_k$. Tímto způsobem získáme velmi robustní systém.

Třídící systémy a vytváření pravidel

Třídící systémy jsou dynamické systémy, které vytvářejí pravidla (na základě *podnět:reakce*) nebo třídiče (užitím genetického algoritmu). Třídící systém dostává vstup (zprávy) z monitorovaného prostředí ve formě binárních řetězců. Výstup takového systému představuje rozhodnutí o reakci. Když tedy třídící systém dostane zprávu z monitorovaného prostředí, tak ji zařadí na tzv. nástěnku (angl. message board). Následně je prohledán třídící seznam, aby se našlo pravidlo, které vyhovuje zprávě. Pokud existuje více takových pravidel, tak je vybráno to, které je silnější (důležitější). Pokud zprávě nevyhovuje žádné pravidlo, tak třídící systém vytvoří nové pomocí genetického algoritmu. Takové pravidlo má tvar *podmínka:reakce*.

Podmínková část nabývá hodnot "0", "1" nebo "#". Znaků "#" náleží "0" nebo "1". Reakční část nabývá v našem případě sedmi hodnot v závislosti na typu reakce. Příklad takového pravidla může vypadat takto:

```
#1001101#110#00#01###0001#111110#110#:5
```



Obrázek 3.1: Komponenty třídícího systému pro podporu rozhodování.

Genetický algoritmus

Genetický algoritmus (GA) [14] [16] reprezentuje třídu adaptivních metod pro prohledávání, které si berou příklad z přírodního genetického vývoje. Genetické algoritmy operují na statistickém souboru ucházejících se řešení a aplikují princip silnějšího s účelem vytvářet lepší aproximace správného řešení. Ke správnému aplikování GA je potřeba si zavést tyto čtyři komponenty:

1. Syntax chromozómu - reprezentuje prostor
2. Interpretace chromozómu - dekóduje a kontroluje proveditelnost
3. Vhodná míra chromozómu - rozezná kvalitu řešení
4. Genetické operátory - manipulace s řešeními výměnou a mutací operátorů

GA pracuje se statistickým souborem ucházejících řešení (chromozómů). Vhodnost každého elementu takového řešení (neboli jednoho článku v prostoru) je odhadován odhadovou funkcí, která měří, jak je element výkonný (vzhledem k dané problematice). Meze mohou být v odhadové funkci zohledněny ve formě tzv. pokut (tak jak je známe například z umělých neuronových sítí). Výkonnější elementy jsou odměňovány a naopak nevýkonné elementy jsou sankcionovány nebo vyřazeny. Takže začne se s počáteční populací. Genetický algoritmus vytěží informace ze současné populace a prozkoumá nové elementy z budoucí populace generováním potomků. Tento selektivní množící proces vede s velkou pravděpodobností k nejbližšímu řešení.

Během vytváření sady pravidel se ohodnocuje každé pravidlo, abychom rozeznali vhodnou hodnotu takového pravidla. Ke generování rozdílné sady pravidel se používá tzv. sekvenční doplňující technika.

Fúze dat

Každé pravidlo je vytvořeno pomocí genetického prohledávajícího procesu. V naší implementaci je genetická reprezentace ucházejícího řešení nazývána chromozóm a je složeno z parametrů představující různé hodnoty na různých úrovních měření - viz. tabulka. Cílem je najít korelaci mezi odchylkami hodnot měřených parametrů od normálního stavu a tím zjistit typ průniku.

úr	uživatelská	systemová	procesů	paketová
par	U1 U2 U3 U4 U5	S1 S2 S3 S4 S5	P1 P2 P3 P4 P5	N1 N2 N3 N4 N5
př	00 11 10 11 11	10 01 00 11 10	00 01 10 11 11	10 11 10 01 00

Tabulka 3.2: Schéma chromozómu.

P^1	P^2	P^3	P^4
$P_1^1 \dots P_5^1$	$P_1^2 \dots P_5^2$	$P_1^3 \dots P_5^3$	$P_1^4 \dots P_5^4$
$c_1, c_2 \dots c_{10}$	$c_{11}, c_{12} \dots c_{20}$	$c_{21}, c_{22} \dots c_{30}$	$c_{31}, c_{32} \dots c_{40}$

Tabulka 3.3: Chromozómu symbolicky.

Ohodnocování

Cílem ohodnocování funkce je odhadnout jak hodně každý chromozóm (pravidlo) ovlivní řešení problému. Při tomto hledání jsou sledovány následující body:

- jak dobře pravidlo reflektuje znalost problematiky - odchylka každé úrovně by měla být co nejbližší daným specifikám
- stupeň obecnosti pravidla - pravidlo může být buď hodně specifické (bude obsahovat jen pár "#") nebo obecnější (bude obsahovat více "#")
- diverzita vybrané optimální sady pravidel - tohoto se docílí právě sekvenční doplňující technikou, která bude popsána níže.

Chromozóm bude symbolicky zapsán jako $C = \langle c_1, c_2, \dots, c_{40} \rangle$, kde $c_i \in \{ "0", "1", "#" \}$.

P^i je podíl chromozómu vztahující se k úrovni i ($i = 1$ pro uživatelskou úroveň, $i = 2$ pro systémovou úroveň, atd.). P_j^i představuje podíl chromozómu v souvislosti s parametrem j úrovně i ($i = 1, \dots, 4$). Dále potřebujeme následující definice:

$Min(P_j^i)$: minimální hodnota, kterou může P_j^i nabývat - viz. tabulka 3.4.

P_j^i	$Min(P_j^i)$
00	0
01	1
10	2
11	3
0#	0
#0	0
1#	2
#1	1
##	0

Tabulka 3.4:

A současně

$$MinValue(P^i) = \frac{1}{5} \sum_{j=1}^5 \frac{Min(P_j^i)}{3}$$

$NumWC(P^i)$: počet znaků "#" v P^i .

Hypotéza je tedy tvořena seznamem reálných hodnot $(k_1, k_2, k_3, k_4, k_5)$. Funkce *KnowledgeDist* může být definována jako vzdálenost minimální hodnoty každé úrovně k jedné konkrétní hodnotě.

$$KnowledgeDist(C) = \frac{1}{4} \sum_{i=1}^4 |k_i - MinValue(P^i)|$$

Funkce *Generality* měří na každé úrovni rozdíl mezi počtem použitých znaků "#" a jejich optimálním počtem (*OptNumWC*). *GenCoef* nabývá hodnot 0.0 až 1.0 a určuje důležitost míry všeobecnosti při určování vhodných hodnot.

$$Generality(C) = GenCoef * \frac{1}{4} \sum_{i=1}^4 |OptNumWC - NumWC(P^i)|$$

Způsobnost (vhodnost) je nyní počítána jako rozdíl *KnowledgeDist* a *Generality* od hodnoty 2, za předpokladu, že maximální hodnota způsobnosti bude 2.0.

$$Fitness(C) = 2 - KnowledgeDist(C) - Generality(C)$$

Sekvenční doplňující algoritmus

Tento algoritmus se využívá k vytvoření rozdílné sady pravidel, který vlastně opakovaně spouští GA s různými hodnotami funkce *Fitness*. Dáme tomu, že $B = \langle b_1, b_2, \dots, b_n \rangle$ jsou nejlepší výsledky GA. Podstatou tohoto algoritmu je to, že se zavrhne vhodná hodnota případu v závislosti na vzdálenosti tohoto případu od neoptimálnějšího případu. Řekněme, že f je ten případ, tak $dist(f, b_i)$ bude vzdálenost případu f od neoptimálnější možnosti b_i . f bude tedy modifikováno ($fitness'(f)$ bude modifikované f) takto:

$$fitness'(f) = M_n(f)$$

, kde

$$M_0 = Fitness(f)$$

$$M_i = M_{i-1} * G(f, b_i)$$

$$G(f, b_i) = \begin{cases} (dist(f, b_i)/R)^{alpha} & \text{když } dist(f, b_i) < R \\ 1 & \text{v ostatních případech} \end{cases}$$

Kde R a $alpha$ jsou parametry zde vysvětlovaného algoritmu.

Kapitola 4

Prvotní experimenty a výsledky

Bylo uskutečněno mnoho experimentů s různými daty s cílem testování výkonnosti třídícího systému pro podporu rozhodování. Tato testování byla prováděna v laboratorních podmínkách. Byly použity vysokoúrovňové znalosti problematiky (o kterých už jsme mluvili) a např. Dipankar Dasgupta a Fabio A. Gonzalez založili na následujících hypotézách:

1. Pokud $U \geq 0.2$ nebo $S \geq 0.2$ nebo $P \geq 0.2$ nebo $N \geq 0.2$, pak proved' reakci 1
2. Pokud $N \geq 0.5$ nebo $U \geq 0.5$ nebo $P \geq 0.5$, pak proved' reakci 2
3. Pokud ($U \geq 0.4$ a $S \geq 0.4$) nebo ($P \geq 0.4$ a $N \geq 0.4$), pak proved' reakci 3
4. Pokud $U \geq 0.8$ nebo $S \geq 0.8$ nebo $P \geq 0.8$ nebo $N \geq 0.8$, pak proved' reakci 4
5. Pokud ($U \geq 0.4$ a $P \geq 0.6$) nebo ($S \geq 0.4$ a $N \geq 0.6$), pak proved' reakci 5
6. Pokud ($U \geq 0.5$ a $S \geq 0.5$) nebo ($P \geq 0.5$ a $S \geq 0.6$), pak proved' reakci 6
7. Pokud ($U \geq 0.6$ a $S \geq 0.6$ a $P \geq 0.6$) nebo ($N \geq 0.6$ a $P \geq 0.6$), pak proved' reakci 7
8. Pokud $U \geq 0.7$ a $S \geq 0.7$ a $P \geq 0.7$ a $N \geq 0.7$, pak proved' reakci 8

Účelem je testovat výkonnost vytvořené sady pravidel na libovolném vstupu. Aby toto mohlo být realizováno, bylo vygenerován seznam 40-bitových zpráv (jakožto monitorovaných parametrů) jakožto vstup pro třídící systém. Pro každou akci byl porovnán výstup (reakci) s akcí navrhovanou pro hypotézu. Systém byl testován 10 000 zpráv s různými odchylkami za účelem testování robustnosti a vůbec pokrytí rozsahu detekce. Byly vybrány 4 typy experimentů jejichž testování můžete sledovat v tabulce 4.1.

reakce	1	2	3	4
1	78%	22%	0%	1%
2	2%	96%	0%	2%
3	7%	38%	53%	3%
4	0%	4%	17%	59%

Tabulka 4.1: Výsledky testu výkonnosti vytvořeného systému.

Například první řádka (čili skutečná, správná akce podle dostupných znalostí) znamená, že systém identifikoval reakci 1 na 78% zpráv, reakci 2 na 22%, atd. Zvýrazněné vyšší hodnoty na diagonále indikují, že systém mohl správně vybrat (popř. rozeznat) vhodné akce ve většině případů.

Stále tu mluvíme o reakcích, které budou použity v závislosti na typu vstupní zprávy, ale ještě jsme si žádné nedefinovali. Třídící systém tedy vybere jednu nebo více s níže uvedených akcí, které jsou opět spíše rozhodnutím individuální implementace:

- A0. Žádná reakce (ignorace incidentu)
- A1. Informování administrátora elektronickou cestou (například e-mailem)
- A2. Změna priority uživatelského procesu
- A3. Změna přístupových práv uživatele
- A4. Blokování určité IP adresy nebo odesílatele vůbec
- A5. Odmítnutí navázání vzdáleného spojení
- A6. Ukončení síťového spojení
- A7. Restartování počítače

Kapitola 5

Závěr

Většina současných IDS používají informace z paketové nebo uživatelské úrovně k vykonávání rozhodnutí v případě incidentu. V této práci bylo popsáno IDS, které monitoruje síťovou aktivitu současně na několika úrovních. Takže lze detekovat jak vnitřní zneužívání, tak útoky z vnějšku sítě. Hlavním cílem této práce bylo demonstrovat použití inteligentního třídícího systému pro podporu rozhodování jako robustního nástroje při detekci incidentů.

Literatura

- [1] Dipankar Dasgupta, Fabio A. Gonzalez: An Intelligent Decision Support System for Intrusion Detection and Response
- [2] Tomáš Pluskal: Intrusion detection system based on process behavior rating, 2004, online na Internetu: <http://plusik.pohoda.cz/thesis/>
- [3] Josef Kadlec: Forenzní analýza digitálních dat, 2004, semestrální práce FIM UHK
- [4] Crosbie M., Spafford G.: Applying Genetic Programming to Intrusion Detection, 1997, Pardue University
- [5] Dasgupta D.: Immunity-Based Intrusion Detection System: A General Framework, 1999, National Information Systems Security Conference
- [6] Frank J.: Artificial Intelligence and Intrusion Detection: Current and future directions, 1994, National Information Systems Security Conference
- [7] Balasubramaniyan J., Fernandes J.O.G., Isacoff D., Spafford E., Zamboni D.: An Architecture for Intrusion Detection using Autonomous Agents, 1998, Pardue University
- [8] Me L., Gassata: A Genetic Alghorithm as an Alternative Tool for Security Audit Trail Analysis, 1998, First International Workshop on the Recent Advances in Intrusion Detection in Belgium
- [9] Boer B.: Classifier Systems, A useful approach to machine learning?, 1994, Leiden University
- [10] Mukherjee B., Heberline L.T., Levit K.: Nwtwork Intrusion Detection, 1994 IEEE Network

- [11] Axelsson S., Lindqist U., Gustafson U., Jonsson E.: An Approach to UNIX security Logging, 1996, Technical Report IEEE Network
- [12] Lunt T.F.: Real-time Intrusion Detection, 1990, Technical Report Computer Science Journal
- [13] Debar H., Dacier M., Wepszi A.: A Revised Taxonomy for Intrusion Detection Systems, 1999, Technical Report Computer Science/Mathematics
- [14] Goldberg D.E., Genetic Alghorithm in Search, Optimazation & Machine Learning, 1198, Assison-Wesley
- [15] Back T., Fogel D.B., Michalewicz Z.: Handbook of Evolutionary computation, 1997, Institute of Physics Publishing and Oxfort university press
- [16] Dasgupta D., Michalewicz Z.: Evolutionary Algorithms in Engineering and Applications, 1997, Springer-Verlag

Práce byla vytvořena sázecím jazykem \LaTeX .