

O uso de *clustering* em sistemas de detecção de intrusão

Márcio da Rós Gomes

29 de outubro de 2004

Gostaria de agradecer a meus pais, meus irmãos e minha família pelo apoio constante, em especial durante o meu curso de graduação. Agradeço também a todos os meus professores pelos ensinamentos, em especial aos professores Raul, Sérgio, Rosane e Cláudia Boeres por, além de serem mestres, serem também amigos e conselheiros nas horas difíceis. Agradeço também à Kelly, por ter tido paciência de me agüentar falando desse projeto o tempo todo. Aos amigos Hilário, Mariella, Alexandre e Jociel por todas as batalhas que enfrentamos juntos. Agradeço à Donald E. Knuth e Leslie Lamport por, além da significativa contribuição dada a computação, terem criado o \TeX e o \LaTeX , me permitindo, assim, expor meu trabalho de forma simples e bonita.

Resumo

Neste trabalho é investigado o uso de *clustering* para a construção de sistemas de detecção de intrusão. Para isso, explicamos o que são sistemas de detecção de intrusão e algumas das técnicas utilizadas para sua implementação. Explicamos também o que é *clustering* e como esse pode ser usado para a construção de um sistema de detecção de intrusão. Realizamos testes e apresentamos os resultados dos experimentos.

Sumário

1	Segurança em computação	3
1.1	Introdução	3
1.2	O problema da detecção de intrusão	3
1.3	<i>Secure Computing</i> e <i>Trusted Computing</i>	4
1.4	Objetivos deste trabalho	5
2	Sistema de detecção de Intrusão	6
2.1	Introdução	6
2.2	Terminologia	6
2.3	Descrição de um sistema de Detecção de Intrusão genérico	7
2.3.1	Arquitetura	7
2.4	Taxonomia de Sistemas de Detecção de intrusão	8
2.4.1	Métodos de Detecção	8
2.4.2	Local da fonte de Dados	10
2.4.3	IDSs baseados em <i>Host versus</i> baseados em Rede	11
3	O algoritmo de <i>clustering</i> DBSCAN	13
3.1	Introdução	13
3.2	O algoritmo DBSCAN	13
4	IDSs e <i>clustering</i>	16
4.1	Introdução	16
4.2	<i>Clustering</i> dos dados de Auditoria	16
4.3	Detecção de anomalias a partir dos <i>clusters</i>	20
5	Alguns experimentos	24
5.1	Introdução	24
5.2	Os experimentos	24
5.3	Os resultados	25
5.4	Conclusões	26
6	Trabalhos futuros	30

Lista de Figuras

2.1	Arquitetura de IDS simples	8
2.2	Características de Sistemas de Detecção de Intrusos	9
3.1	Exemplos de clustering - (a) DBSCAN - (b) por particionamento	14
4.1	Como o <i>clustering</i> é efetuado	18
5.1	Semana 1 com relação à semana 1 - falsos alarmes	25
5.2	Semana 1 com relação à semana 1 - limite e valores	26
5.3	Semana 3 com relação à semana 1 - limite e valores	27
5.4	Semana 3 com relação à semana 1 - falsos alarmes	27
5.5	Semana 2 com relação à semana 1 - taxa de detecção	28
5.6	Semana 2 com relação à semana 1 - limite e valores	28
5.7	Semanas 4 e 5 com relação à semana 1 - taxa de detecção	29
5.8	Semanas 4 e 5 com relação à semana 1 - limite e valores	29

Capítulo 1

Segurança em computação

1.1 Introdução

A segurança em computação é um esforço para criar uma plataforma de computação segura, especificada e desenhada de tal forma que o usuário não possa executar ações que não lhe sejam permitidas, mas ainda permitindo que o usuário possa executar as ações que lhe são permitidas. Exemplos de ações acima são acesso, modificação, execução, exclusão e delegação de arquivos ou dispositivos. A segurança em computação está baseada no tripé: integridade, confidencialidade e disponibilidade. Essas três propriedades devem ser mantidas para que um sistema seja seguro. O objetivo deste trabalho é a investigação de uma técnica de sistemas de detecção de intrusão. Tais sistemas são usados para tentar manter a integridade dos sistemas de computação.

1.2 O problema da detecção de intrusão

A preocupação com a integridade dos sistemas de computação vem do custo que a falta desta causaria para uma pessoa, empresa ou país. Hoje, onde sistemas de computação são usados para controlar desde aeroportos ao lançamento de mísseis intercontinentais, passando por sistemas de gerenciamento e de finanças de grandes empresas, essa preocupação torna-se uma constante nos ambientes de gerenciamento de tecnologia da informação. Essa preocupação começou a ser observada no ambiente militar, onde os ambientes computacionais tornaram-se mais comuns [And72, KS74].

A partir da disseminação da internet na década de 90, os sistemas de computação ficaram expostos a milhares de pessoas através da rede, ficando muito vulneráveis a ataques externos. A exploração de falhas de programas para tomar controle de sistemas remotos ou mesmo danificá-los é comum até os dias de hoje. Para tentar contornar ou ao menos amenizar esse problema, foram criados sistemas de detecção de intrusão, que tentam detectar e barrar um ataque ou intrusão, avisando o administrador do ocorrido. Teremos uma visão mais detalhada de sistemas de detecção de intrusão no capítulo 2.

1.3 *Secure Computing e Trusted Computing*

Existem duas abordagens diferentes para a segurança em computação que são usadas até hoje: *Secure Computing* e *Trusted Computing*. *Secure Computing* foca na correção formal ou parcial dos subsistemas de um sistema. Assim, um sistema é considerado seguro se seus componentes são provados que também o são. Esse tipo de abordagem assume que nenhum sistema poderá ser completamente seguro a não ser que haja uma prova formal de que este esteja correto e de acordo com a especificação do sistema.

Por outro lado, em *Trusted Computing* assume-se que é inviável a prova formal de sistemas tão complexos quanto um sistema operacional e são usados mecanismos para a proteção e monitoramento do sistema baseado nessa suposição. Esse é o tipo de técnica adotada na maioria dos atuais sistemas de computação. Geralmente esse tipo de abordagem é piorada com a suposição de que o sistema de computação em si é confiável, bastando então somente a preocupação com riscos externos ¹.

Desse modo, surgiu a necessidade da criação do que é chamado *Trusted Operating Systems*(TOS)[MM04]:

“Trusted operating systems provide the basic security mechanisms and services that allow a computer system to protect, distinguish, and separate classified data. Trusted operating systems have been developed since the early 1980s and began to receive National Security Agency (NSA) evaluation in 1984.”

TOS são uma tentativa de minimizar os riscos de segurança de sistemas que processam dados classificados ². TOS implementam políticas de segurança como MAC ³ e sistemas de auditoria em vários níveis como os definidos em [oD85] e mais tarde em [Org99]. Políticas de segurança são as regras e práticas usadas para controlar como e por quem dados ou dispositivos podem ser acessados, modificados ou distribuídos. Sistemas de auditoria têm o propósito de identificar quem teve acesso a que dado para que um usuário possa ser responsabilizado por seus atos ou para que se possa ter um rastro de quais informações foram acessadas quando houve uma violação.

TOS também têm um ponto interessante que é a avaliação por um órgão que emite uma certificação de que o sistema está em conformidade com a especificação do nível de segurança para o qual está sendo avaliado. Alguns desses órgãos são a NSA ⁴ e mais recentemente uma união de alguns países para a definição de um padrão de avaliação de sistemas - *Common Criteria*. ⁵

¹http://encyclopedia.thefreedictionary.com/computer_insecurity

²dados classificados - refere-se ao sistema de classificação de dados usado em organizações militares americanas, onde dados são classificados em classes (*unclassified*, *confidential*, *secret*, *top secret*)

³MAC(Mandatory Access Control) [Den76] - controle de acesso no qual são definidos sujeitos, objetos e funções de autorização entre esses.

⁴NSA (National Security Agency) - Agência de segurança nacional norte-americana que, entre outras funções, faz pesquisas em sistemas de computação seguros <http://www.nsa.gov>

⁵*Common Criteria* - O *Common Criteria* representa o resultado de esforços para desenvolver critérios de avaliação de segurança de tecnologia da informação que são amplamente úteis na comunidade internacional. <http://www.commoncriteriportal.org>

1.4 Objetivos deste trabalho

Neste trabalho serão investigados sistemas de detecção de intrusão, em especial sistemas de *host* baseados em anomalia de comportamento de usuários e programas. Assim, investigamos o uso de *clustering* dos dados de auditoria do uso normal do sistema de computação para criação de perfis de usuários. Através da comparação de novos dados auditados com o perfil, é possível a detecção de anomalias no uso do sistema por um usuário. Essas anomalias normalmente indicam tentativas de intrusão por parte de intrusos ou por parte de usuários legítimos.

Capítulo 2

Sistema de detecção de Intrusão

2.1 Introdução

Sistemas de detecção de intrusão são aqueles que têm por objetivo descobrir falhas de segurança, falhas em que houve tentativas de exploração ou vulnerabilidades que poderiam levar a potenciais falhas. Podem ser baseados em *host*, onde dados internos coletados pelo sistema operacional e de programas são usados para efetuar a detecção, ou baseados em rede, quando funcionam capturando pacotes de rede que trafegam para investigar seu conteúdo e detectar algum ataque. Abaixo falaremos com mais detalhes sobre esses sistemas e como funcionam.

2.2 Terminologia

- IDS - Sigla usada para representar *Intrusion Detection System* - Sistema de Detecção de Intrusão. O plural será representado por IDSs;
- Sistema ou sistema alvo - é aqui usado para denotar um sistema sob monitoramento. Esse sistema pode ser uma estação de trabalho, um servidor, um *firewall* ou equipamento de rede;
- *Audit* - usado para fazer referência ao resultado de uma auditoria feita em um sistema. O resultado da auditoria em um sistema geralmente é composto por alguns dados como: informações de login/logout, rastro (*trace*) da execução de programas ou dados fornecidos por *syslog*;
- Explorar - usado no contexto da exploração de uma falha em sistema, ou seja, o uso de uma falha em um sistema para obter privilégios que podem potencialmente ser usados para causar algum dano ao sistema ou obter acesso a dados nele contidos;
- Ataque - ação conduzida por um adversário, o intruso, contra outro adversário, a vítima. O intruso conduz um ataque com um objetivo específico em mente. Na perspectiva de um administrador de sistemas, um ataque

é um conjunto de um ou mais eventos que podem ter uma ou mais consequências de segurança. Na perspectiva de um intruso, um ataque é um mecanismo de atingir um objetivo [ACF⁺00];

- Intrusão - usada não só no sentido de ataque, mas também de ataque bem sucedido;
- Vulnerabilidade - uma característica ou combinação de características de um sistema que permite que um adversário ponha o sistema em um estado que é contrário aos desejos das pessoas responsáveis pelo sistema e aumenta a probabilidade ou magnitude de um comportamento indesejável no sistema [ACF⁺00];
- Assinatura - conjunto de informações já previamente levantadas sobre um ataque conhecido, como por exemplo, a sequência de caracteres usado em uma certa requisição para que uma falha seja explorada, ou os passos seguidos para que uma falha seja explorada.

2.3 Descrição de um sistema de Detecção de Intrusão genérico

2.3.1 Arquitetura

Como descrito em [DDW99], um sistema de detecção de intrusão é aquele que obtém informações sobre o ambiente e as usa para criar um diagnóstico de segurança sobre este. Seu objetivo é descobrir falhas de segurança, falhas em que houve tentativas de exploração, ou vulnerabilidades que poderiam levar a potenciais falhas. Um sistema de detecção de intrusão pode ser logicamente distribuído em três componentes: sensores, analisadores, e uma interface com o usuário [ACF⁺00, DDW99]. A figura 2.1 mostra uma arquitetura simples de uma IDS como proposta pelo idwg¹. Seus componentes são:

- Sensores - São responsáveis por coletar dados. Os dados de entrada para os sensores podem ser quaisquer tipos de evidências de uma intrusão. Alguns exemplos de entrada são arquivos de log, registros de chamadas de sistema ou pacotes de rede. Os sensores repassam as informações relevantes para os analisadores;
- Analisadores - Recebem dados de um ou mais sensores ou outros analisadores. O analisador decide se os dados são evidências de um ataque e emite uma notificação de que uma intrusão ocorreu. A saída, além da conclusão do analisador, pode incluir as evidências que deram suporte à decisão tomada;
- Interface com o usuário - Permite mostrar a saída do sistema e controlar o seu comportamento.

¹IETF Intrusion Detection Working Group - <http://www.ietf.org/html.charters/idwg-charter.html>

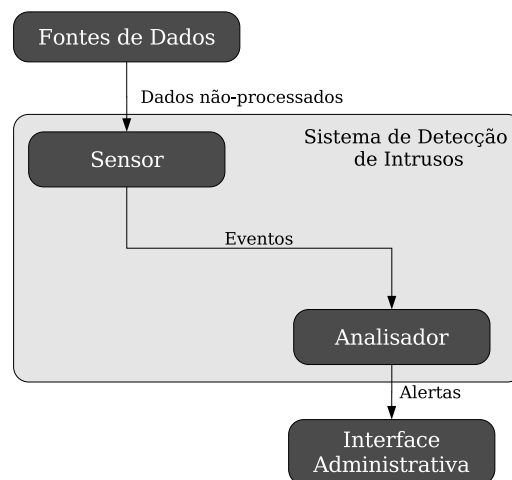


Figura 2.1: Arquitetura de IDS simples

2.4 Taxonomia de Sistemas de Detecção de intrusão

Como ilustrado em [DDW99], sistemas de detecção de intrusão podem ser classificados segundo cinco características:

- Método de Detecção;
- Comportamento ao detectar;
- Local da fonte de dados auditados;
- Paradigma de detecção;
- Frequência de uso.

A figura 2.2 ilustra a hierarquia dos sistemas segundo [DDW99]. Na seção 2.4.1 detalharemos alguns dos métodos de detecção e aspectos quanto ao local da fonte dos dados.

2.4.1 Métodos de Detecção

Os métodos de detecção podem ser classificados como baseados em comportamento ou baseados em conhecimento. Os primeiros tentam traçar um comportamento normal do sistema e a partir desse detectar desvios no comportamento de programas ou usuários. Esses métodos são também conhecidos como *Anomaly Detection*. Já os métodos baseados em conhecimento, também chamados de *Misuse Detection*, usam assinaturas de ataques para detectar que houve ou há um ataque em andamento [LX01].

Misuse Detection

O princípio das técnicas baseadas em conhecimento é o uso de informações sobre ataques e vulnerabilidades já conhecidos em sistemas para a detecção de

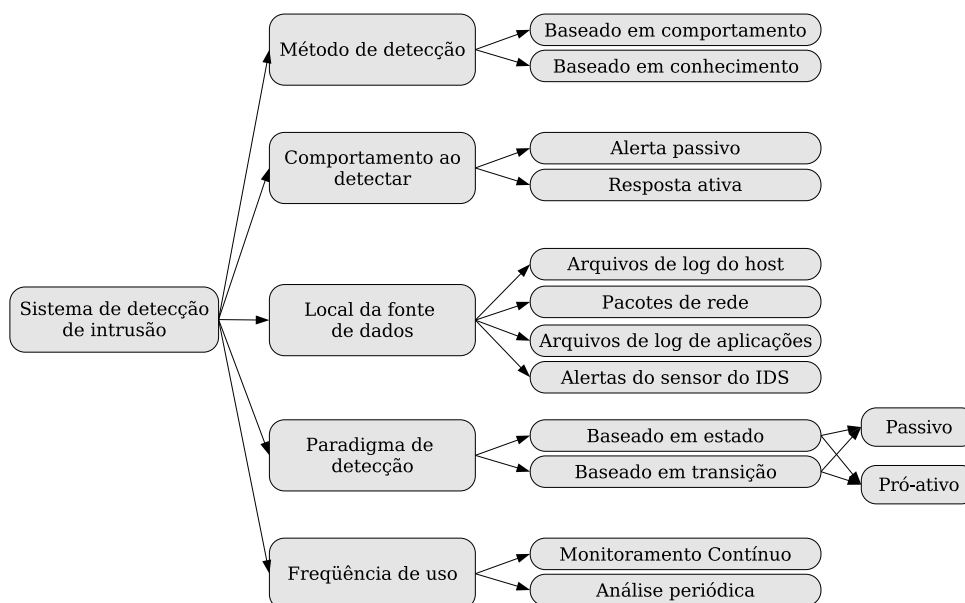


Figura 2.2: Características de Sistemas de Detecção de Intrusos

ataques. O funcionamento de sistemas baseados em conhecimento é relativamente simples. Assinaturas de ataques são armazenadas no sistema de detecção e a partir de dados coletados, geralmente do tráfego de rede, pode-se verificar a intenção ou presença de um ataque. Ou seja, qualquer ação que não esteja explicitamente definida como sendo um ataque não será detectada [DDW99].

O analisador deve efetuar um casamento de padrões entre os dados de ataques já conhecidos e os dados que estão trafegando na rede ou de outra fonte de dados. Assim, para que o sistema possa detectar todos os ataques, ele deve ter conhecimento de todos os que já ocorreram [ACF⁺00].

Anomaly Detection

Métodos de detecção baseados em comportamento usam informações passadas do uso de programas e recursos computacionais para traçar o comportamento de programas. A partir do modelo de comportamento normal do sistema é feita a detecção de qualquer comportamento que não seja o esperado. Assim, pode ser gerado um alarme quando um desvio de comportamento é encontrado.

Esse modelo pode ser criado a partir de várias informações sobre o ambiente e o que acontece nesse. Algumas das fontes de dados desse tipo de técnica são dados auditados no sistema operacional, como chamadas de sistema, uso de cpu/memória/disco e também os *logs* do sistema [LS98, DDW99]. Podem ser usadas também informações sobre o tráfego de rede como o registro de conexões ou tentativas de conexões e a duração destas [LS00]. O IDS passa por uma fase de treinamento inicial para que seja criado um modelo do comportamento do sistema. Essa fase é importante pois define o que será detectado como anomalia no futuro.

Métodos baseados em *Anomaly Detection* têm a vantagem de poder detectar intrusões sem o prévio conhecimento da falha que o intruso esteja tentando explorar. Eles podem detectar novos tipos de ataques, mas também são úteis quando é usado abuso de poder para poder ter acesso a informações privilegiadas ou causar algum dano, pois estes não envolvem a exploração de qualquer falha no sistema mas apresentam um comportamento anômalo.

Anomaly Detection versus Misuse Detection

Sistemas baseados em *Misuse Detection* são efetivos na detecção de ataques conhecidos. Porém, é necessário que se mantenha a base de dados de assinaturas em dia com os ataques que já são conhecidos. Outro problema enfrentado por esse tipo de técnica é, também, o crescente número de novas vulnerabilidades [CER04] e a variabilidade das assinaturas, ou seja, com uma ligeira mudança no código usado por um atacante, esse passa a não ser mais detectado pelo sistema de detecção de intrusões. Esses fatores levam à necessidade de um sistema onde o tempo entre a descoberta de um novo ataque e a criação de uma assinatura para esses seja o mínimo possível, o que nem sempre é viável.

Sistemas baseados em comportamento, apesar de não necessitarem de uma atualização constante, podem causar um alta taxa de falsos alarmes devido a algum comportamento que não foi explicitado no momento do treinamento. Há também o problema da mudança de comportamento, durante o tempo, de algum elemento analisado, o que não significa necessariamente que o sistema está sendo atacado. O lado bom desses sistemas é que eles são capazes de detectar tentativas de exploração de falhas ainda não documentadas. Para minimizar falsos alarmes podem ser feitas atualizações no comportamento. Outro problema é que atualizações no comportamento podem ser usadas por intrusos para tentar modificar o que o sistema considera como normal e passar a não detectar suas tentativas de intrusão [DDW99, LS00].

2.4.2 Local da fonte de Dados

A taxonomia de IDSs vem sendo pesquisada há alguns anos e, nessas taxonomias, uma das classificações feitas é quanto ao local da fonte de dados. Alguns locais citados em [DDW99] são listados abaixo.

Fontes de dados de *host*

Dados auditados no sistema são a única forma de adquirir informações sobre o que acontece em um *host*. Entre as informações possíveis de ser coletadas no sistema, podemos citar:

- Fontes do Sistema Operacional - informações sobre o que é executado em algum momento no sistema, usando comandos como *ps*, *pstat*;
- Contabilidade - pode-se usar estatística sobre uso de memória/*CPU*/disco como fonte de dados;
- *syslog* - *syslog* é um servidor de log usado por outras aplicações e pelo próprio sistema operacional para relatar erros ou dados informativos sobre o atual estado dos programas;

- Sistema de auditoria C2 ² - . Esses sistemas impõem um fino controle dos usuários, fazendo com que usuários sejam individualmente responsabilizados por seus atos através de informações refinadas como o que é executado por quem, o que foi acessado, se obteve sucesso e quando isso ocorreu. Esse tipo de auditoria é a primeira fonte de informações usada pelo IDSs baseados em *host*.

Fonte de dados de aplicativos

O uso de arquivos de log de aplicativos pode ser bastante útil na detecção de ataques, algumas vantagens são:

- Exatidão - os logs de aplicativos fornecem informações exatas e concisas sobre o que acontece no sistema, não necessitando a interpretação por parte do sistema do que aconteceu, pois a interpretação do evento já foi dada pelo programador quando criou a mensagem do log;
- *Completeness* - Não existe a necessidade de remontar pacotes ou reconstruir o caminho seguido por um programa enquanto ele executava, como nos dados de rede e de auditoria. Implicando em menos processamento;
- Desempenho - Como um aplicativo só gera informações estritamente relevantes para questões de segurança, o *overhead* necessário para fazer processamento das informações é muito menor.

Fonte de dados de rede

Dados da rede são a principal fonte de dados dos IDSs de Rede e, entre os dados usados por esses, tem-se:

- SNMP ³ - informações usadas para gerenciamento da rede podem também ser usadas para detectar intrusões;
- Pacotes de Rede - muitos dos ataques efetuados atualmente ocorrem através da rede, como por exemplo *Denial of Service*⁴. Sendo assim, uma análise dos pacotes de rede pode ser usada para detectar ataques.

2.4.3 IDSs baseados em *Host versus* baseados em Rede

IDSs baseados em *host* foram os que surgiram primeiro, pois eram executados em *mainframes*, onde vários usuários executavam suas tarefas. Com o *downsizing*⁵, os sistemas passaram a ser mais dependentes da rede e, por conseguinte, os ataques também passaram a se concentrar nessa. Assim surgiram os IDS baseados em rede, pois esses teriam uma certa onipresença, pois todos os sistemas que dependem da rede poderiam ser monitorados ao mesmo tempo. Porém, IDSs baseados em Redes tem ação restrita a casos de ataques já conhecidos e casos

²Sistema de auditoria C2 - sistemas de auditoria que são classificados como sendo do nível C2 como descrito em [oD85]

³SNMP - Simple Network Management Protocol (RFC 1157)

⁴Denial of Service - Ataque de negação de Serviço - ataque no qual o sistema alvo recebe tantas conexões falsas simultâneas que acaba impossibilitado de atender as conexões legítimas

⁵downsizing - Processo de mover uma aplicação/programa de um *mainframe* para um sistema mais barato, tipicamente um sistema cliente-servidor.

nos quais o mesmo ocorre pela rede. Apesar disso, são os mais populares hoje devido à fácil implementação e configuração. Outra desvantagem desses é que o processamento fica centralizado em um sistema, podendo sofrer sobrecarga caso haja um aumento muito grande de tráfego na rede.

Por outro lado, IDSs baseados em *host* são capazes de obter informações mais detalhadas sobre o sistema operacional e os aplicativos que são executados, como detalhado na seção 2.4.2. Desse modo, esse IDS pode tirar conclusões sobre os resultados de uma intrusão. Porém, essas conclusões e os alarmes emitidos devem ser emitidos o quanto antes, pois o próprio sistema de detecção pode ser prejudicado na intrusão. Uma abordagem comum é o uso de sistemas de detecção híbridos, que combinam IDSs de rede e de *host* e esses colaboram para a decisão da emissão de um alarme.

Capítulo 3

O algoritmo de *clustering* DBSCAN

3.1 Introdução

Clustering pode ser entendido como um agrupamento de objetos com características comuns e a separação daqueles com características que os diferenciam. Segundo Guha *et al.* [GRS98], *clustering* pode ser definido como: dados n objetos num espaço métrico de d dimensões, esses objetos são particionados em k grupos (*clusters*), tais que os objetos em um grupo são mais similares entre si do que em relação a objetos de outros grupos. Tais objetos geralmente são representados por um vetor com as d características. Desse modo, deve ser definida uma função de similaridade que aponte o quão semelhante um objeto é de outro. Por exemplo, se tomarmos pontos no espaço bidimensional como objetos, uma das funções de similaridade que pode ser usada é a distância euclidiana.

3.2 O algoritmo DBSCAN

Os algoritmos de *clustering* podem ser classificados em [cdC01]:

Métodos por particionamento;

Métodos hierárquicos;

Métodos baseados em densidade;

Métodos baseados em grade;

Métodos baseados em modelos.

Os métodos mais usados são os por particionamento e os hierárquicos. O algoritmo DBSCAN é baseado em densidade. Os algoritmos baseados em densidade têm a característica de terem bom desempenho para encontrar *clusters* de forma arbitrária e para a identificação de ruído ¹.

¹ruído - aqueles objetos que não puderam fazer parte de nenhum grupo (*cluster*)

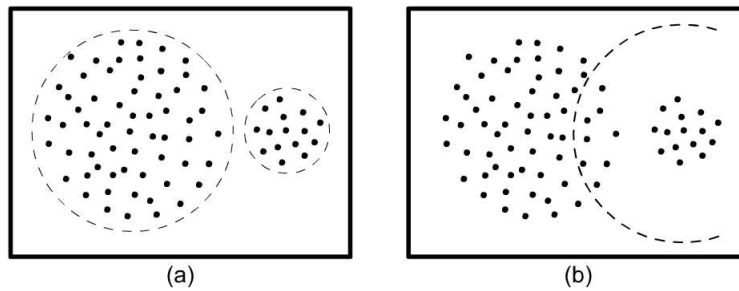


Figura 3.1: Exemplos de clustering - (a) DBSCAN - (b) por particionamento

O algoritmo DBSCAN usa dois parâmetros: *MinPts* e *Eps*. *Eps* significa o raio da vizinhança onde deve haver um número mínimo de pontos (objetos) *MinPts* para que seja formado um *cluster*. O algoritmo 1 é o mesmo descrito em [EKSX96].

Na figura 3.1 temos um exemplo da aplicação de *clustering*, onde no item (a) usa-se o algoritmo DBSCAN para agrupar objetos bidimensionais, usando a distância euclidiana como função de similaridade e a contagem dos objetos para achar os pontos da vizinhança-*Eps* [J04]. No item (b) é mostrado o resultado de um algoritmo por particionamento para a mesma base de dados.

Algorithm 1 DBSCAN(SetOfPoints, Eps, MinPts)

{Todos os itens em SetOfPoints são inicialmente não-classificados}

```
1: procedure DBSCAN(SetOfPoints, Eps, MinPts)
2:   ClusterId = nestId(NOISE)
3:   for i from 1 to SetOfPoints.size do
4:     Point = SetOfPoints.Get(i)
5:     if Point.CId == UNCLASSIFIED then
6:       if EXPANDCLUSTER(SetOfPoints, Point, ClusterId, Eps, MinPts)
           then
7:         ClusterId = nextId(ClusterId)
8:       end if
9:     end if
10:  end for
11: end procedure
12: function EXPANDCLUSTER(SetOfPoints, Point, CId, Eps,
    MinPts):Boolean
13:   seeds = SetOfPoints.regionQuery(Point,Eps)
14:   if seeds.size < MinPts then                                     ▷ no core point
15:     SetOfPoints.changeCId(Point,NOISE)
16:     return False
17:   else                                                           ▷ all points in seeds are density-reachable from Point
18:     SetOfPoints.changeCIds(seeds,CId)
19:     seeds.delete(Point)
20:     while seeds != Empty do
21:       currentP = seeds.first()
22:       result = SetOfPoints.regionQuery(currentP,Eps)
23:       if result.size >= MinPts then
24:         for i from 1 to result.size do
25:           resultP = result.get(i)
26:           if resultP.CId in {UNCLASSIFIED, NOISE} then
27:             if resultP.CId == UNCLASSIFIED then
28:               seeds.append(resultP)
29:             end if
30:             SetOfPoints.changeCId(resultP,CId)
31:           end if                                     ▷ UNCLASSIFIED or NOISE
32:         end for
33:       end if                                     ▷ result.size >= MinPts
34:       seeds.delete(currentP)
35:     end while                                     ▷ seeds != Empty
36:     return True
37:   end if
38: end function
```

Capítulo 4

IDSs e *clustering*

4.1 Introdução

Aqui descreveremos o uso de *clustering* como proposta para a criação de IDSs de *host* baseados em anomalia. Esse uso foi descrito em [OL03] e aqui tentaremos reproduzir alguns dos resultados obtidos e fazer investigações sobre o uso desse tipo de algoritmo na detecção de intrusões.

Programas, ao serem executados em um computador, geram eventos que podem ser coletados por um sistema de auditoria, como os do tipo C2[OD85] providos por alguns sistemas operacionais. Alguns desses eventos são o uso de *CPU*, chamadas de sistema, abertura/fechamento de arquivos. Esses eventos podem ser usados para caracterizar o comportamento de um usuário ou programa. Assim, traçado o comportamento normal de um usuário a partir dos eventos gerados pelas atividades passadas, pode-se detectar quando há algum tipo de anomalia no seu comportamento.

A proposta feita em [OL03] é usar um algoritmo de *clustering* sobre dados do comportamento normal de um usuário e assim poder criar um perfil que mais tarde poderá ser comparado com uma nova atividade para descobrir se essa é anômala.

4.2 *Clustering* dos dados de Auditoria

Para modelar o comportamento de um usuário, suas atividades são divididas em transações. Transações são definidas como sendo um conjunto de atividades executadas em seqüência por um usuário. Uma transação é considerada terminada após um certo tempo de inatividade. Para cada transação, consideramos quais sinais (eventos) e quais as taxas com que esses ocorreram nas atividades dessa transação. Desse modo, para cada transação, temos os sinais que nela ocorreram e a quantidade de vezes que cada um desses sinais ocorreu.

Dado um conjunto de sinais e um conjunto de transações TD das transações passadas de um usuário, seja TD^k um conjunto de transações, cada uma das quais contém o conjunto dos valores do sinal k na transação correspondente em TD . Denotamos por $|TD^k|$ o número total dessas transações e por D^k o conjunto de todos dos valores em TD^k . Ou seja:

- TD - conjunto de transações passadas de um usuário.

- TD^k - conjunto de transações, cada uma das quais com os valores assumidos pelo sinal k na transação correspondente em TD .
- $|TD^k|$ - número de transações em TD^k .
- D^k - conjunto formado pela união de todos os valores assumidos pelo sinal k em todas as transações em TD^k .

Seja cada item (transação) de TD representado por $\{s_1 : c_1, s_2 : c_2, \dots, s_n : c_n\}$, sendo s o sinal e c a quantidade de vezes que esse sinal ocorre na atividade do usuário. Se, por exemplo, temos as seguintes transações em TD :

$$T_1 = \{\{1 : 5, 2 : 7, 3 : 8\}, \{1 : 2, 2 : 7, 3 : 1\}\}$$

$$T_2 = \{\{1 : 3, 2 : 5, 3 : 7\}, \{1 : 3, 6 : 5, 4 : 2\}, \{1 : 4, 2 : 4, 6 : 2\}, \{1 : 2\}\}$$

Assim, os TD^k s seriam:

$$TD^1 = \{\{5, 2\}, \{3, 4, 2\}\}$$

$$TD^2 = \{\{7\}, \{5, 4\}\}$$

$$TD^3 = \{\{8, 1\}, \{7\}\}$$

$$TD^4 = \{\{2\}\}$$

$$TD^6 = \{\{5\}, \{2\}\}$$

E os D^k s seriam:

$$D^1 = \{2, 3, 4, 5\}$$

$$D^2 = \{4, 5, 7\}$$

$$D^3 = \{1, 7, 8\}$$

$$D^4 = \{2\}$$

$$D^6 = \{2, 5\}$$

O processo de *clustering* é efetuado separadamente sobre cada sinal. Assim, sobre cada D^k de um conjunto de transações normais de um usuário é aplicado um algoritmo de *clustering* similar ao DBSCAN para traçar o perfil deste.

Como descrito em [OL03], esse método modela o conjunto de transações nos dados auditados por dois eixos diferentes: um eixo transacional e um eixo de dados. O eixo de dados define um espaço de dados do domínio do problema e modela a similaridade dos dados em termos de cada sinal. De outro lado, o eixo transacional consiste de duas propriedades: inter-transacional e intra-transacional. Para cada sinal do eixo de dados, a propriedade inter-transacional modela o suporte de transação ¹ de dados similares em um conjunto de transações enquanto a propriedade intra-transacional modela a taxa de repetição de dados similares em cada transação.

No algoritmo de *clustering* DBSCAN, que é baseado em densidade, os *clusters* expandem baseados em dois parâmetros: um raio *Eps* e uma quantidade

¹suporte de transação - número de transações nas quais um conjunto de itens de D^k ocorre sobre o total de transações

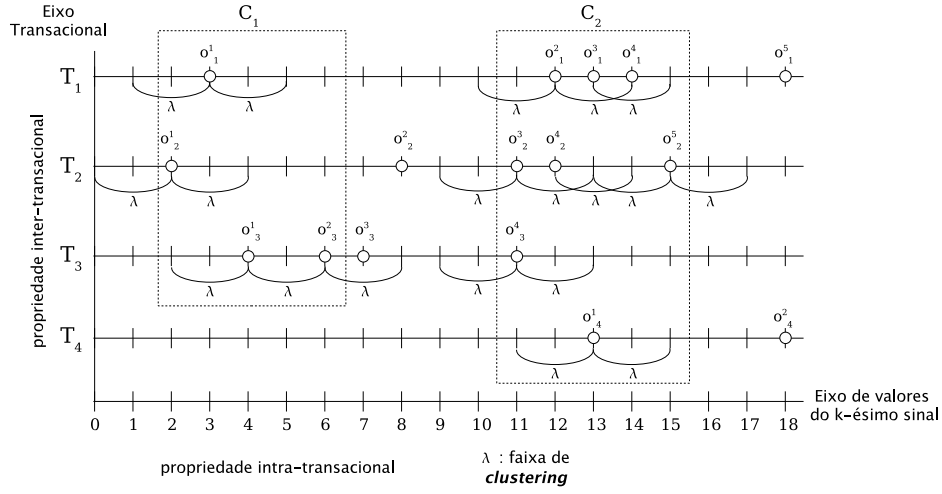


Figura 4.1: Como o *clustering* é efetuado

$MinPts$ mínima de pontos que deve haver na vizinhança de raio Eps para que o *cluster* expanda (ver capítulo 3).

Aqui, o *clustering* é aplicado sobre itens unidimensionais (os itens de D^k). O parâmetro Eps é substituído pela distância λ em D^k , que é usada como medida de similaridade entre os itens que o *clustering* será aplicado. Para que se possa extrair o que há de comum entre as transações, o parâmetro $MinPts$ foi substituído por $minsup$, que significa o mínimo suporte de transação que o grupo de itens da vizinhança λ do item deve ter para que aquele grupo possa ser considerado um *cluster* ou para que um *cluster* existente possa ser expandido.

A figura 4.1 ilustra como o método funciona. Suponha que sejam $\lambda = 2$ e $minsup = 0.75$ e que o TD^i do i -ésimo sinal seja:

$$TD^i = \{\{3, 12, 13, 14, 18\}, \{2, 8, 11, 12, 15\}, \{4, 6, 7, 11\}, \{13, 18\}\}$$

$$\text{Então o } D^i = \{2, 3, 4, 6, 7, 8, 11, 12, 13, 14, 15, 18\}$$

O *clustering* começa pelo objeto o_2^1 que é o menor valor em D^i . Como o_1^1 , o_2^1 e o_3^1 estão à distância λ de o_2^1 eles formam um grupo candidato a se tornar um *cluster*. Como os objetos desse grupo aparecem em 3 transações (1, 2 e 3) e o total de transações é 4, o suporte de transação desse grupo é $3/4 = 0.75 = minsup$. Assim, é formado o *cluster* C_1 com esse grupo. O *cluster* C_1 pode ser expandido se algum de seus objetos também pode formar um *cluster*. Como o suporte de transação dos objetos no grupo da vizinhança de o_3^1 é também igual a $minsup$, o *cluster* C_1 é expandido para o_2^3 . Entretanto, como os objetos do grupo de o_2^3 tem um suporte de transação menor que $minsup$, a expansão do *cluster* pára e os objetos restantes no *cluster* C_1 são examinados para investigar a possibilidade de expansão. Desse modo, para o TD^i acima são formados dois *clusters* C_1 e C_2 como ilustrado na figura 4.1.

No algoritmo DBSCAN, para um dado raio Eps , a vizinhança $N_{Eps}(p)$ de um objeto p é definido por $N_{Eps}(p) = \{q \in D | dist(p, q) \leq Eps\}$ onde $dist(p, q)$ denota a distância euclidiana entre p e q . Similarmente, a vizinhança λ de um

objeto o_i^j para o k -ésimo sinal é definida como $N_\lambda(o_i^j) = \{o_m^j \in D^k \mid |o_i^j - o_m^j| \leq \lambda, 1 \leq m \leq |TD^k|\}$.

Um objeto é considerado *core value* quando o grupo de objetos em sua vizinhança λ tem um suporte de transação maior ou igual a um suporte mínimo *minsup*. Na figura 4.1, os *core values* de C_1 são 2, 3, e 4. Assim, o mínimo *core value* de C_1 é 2 e o máximo 4.

Algorithm 2 clustering(D^k , *minsup*, λ)

D^k : conjunto de todos os valores nas transações em TD^k do k -ésimo sinal
minsup: suporte mínimo de transação de um grupo de dados para que um *cluster* seja formado.

{Todos os itens em D^k são inicialmente não-classificados}

```

1: ordene  $D^k$  em ordem crescente
2: while  $\exists$  o menor objeto não-classificado  $o_i^j$  em  $D^k$  do
3:   recupere  $N_\lambda(o_i^j)$ 
4:   if support( $N_\lambda(o_i^j)$ ) < minsup then
5:     marque o estado de  $o_i^j$  como ruído
6:   else
7:     curobject =  $o_i^j$ 
8:     marque o estado de todos os objetos em  $N_\lambda(\textit{curobject})$  com um novo
       cluster-id cid
9:     curclusterid = cid
10:    escolha  $\hat{o}$  com máximo core value em  $N_\lambda(\textit{curobject})$  onde
       support( $N_\lambda(\textit{curobject})$ )  $\geq$  minsup
11:    while  $\hat{o} > \textit{curobject}$  do
12:      curobject =  $\hat{o}$ 
13:      marque o estado de todos os objetos em  $N_\lambda(\textit{curobject})$  com cur-
        clusterid
14:      escolha  $\hat{o}$  com máximo core value de  $N_\lambda(\textit{curobject})$ 
15:    end while
16:  end if
17: end while

```

Ao final do processo de *clustering* pelo algoritmo 2, os objetos de D^k estão classificados como ruído (não pertencem a nenhum *cluster*) ou como pertencentes a algum *cluster* em particular. O algoritmo 2 descobre um conjunto de *clusters* em D^k , dados os parâmetros de suporte mínimo *minsup* e de vizinhança λ . Quando o suporte de um grupo de dados na vizinhança de um certo objeto tem um suporte maior ou igual a *minsup*, esse conjunto torna-se um *cluster*. Tal *cluster* pode ser expandido com a junção de outros *clusters* adjacentes.

Para que se possa analisar o conhecimento sumarizado pelos *clusters*, algumas propriedades desses são definidas em [OL03]. Dado um *cluster* C^k , $O_{C^k} \subseteq D^k$ representa os objetos de D^k que estão na faixa do *cluster* C^k . As propriedades são **min**, **max**, **tcount**, **center**, **cdev**, **ratio**, **rdev**:

1. $\min(C^k)$ e $\max(C^k)$: $\min(C^k)$ e $\max(C^k)$ são representados pelos valores mínimo e máximo dos itens de O_{C^k}
2. $\textit{tcount}(C^k)$: número de transações diferentes nas quais os itens de O_{C^k} aparecem.

3. $center(C^k)$ e $cdev(C^k)$: $center(C^k)$ representa o valor central do *cluster* C^k . Ele é representado pela média das médias dos valores de O_{C^k} que aparecem em cada transação. Sendo $avg_i(C^k)$ a média dos valores de O_{C^k} que aparecem na transação $T_i \in TD^k$, o centro do *cluster* C^k é representado por:

$$center(C^k) = \frac{\sum_{i=1}^{|TD^k|} avg_i(C^k)}{tcount(C^k)}$$

O desvio padrão do centro do *cluster* C é dado por $cdev(C^k)$

4. $ratio(C^k)$ e $rdev(C^k)$: Denotamos por $r_i(C^k)$ a taxa de repetição individual de uma transação $T_i \in TD^k$. Ela é definida por $r_i(C^k) = |S_i|/|T_i|$, onde S_i corresponde ao número de objetos de T_i que estão no *cluster* C^k . A taxa $ratio(C^k)$ de repetição é dada por:

$$ratio(C^k) = \frac{\sum_{i=1}^{|TD^k|} r_i(C^k)}{tcount(C^k)}$$

O desvio padrão da taxa de repetição do *cluster* C é dado por $rdev(C^k)$

4.3 Detecção de anomalias a partir dos *clusters*

Para que se possa detectar anomalias a partir de uma nova transação e um perfil gerado pelo *clustering*, é necessário que possamos medir o quão longe uma nova transação está do perfil do usuário. Para isso, são definidas algumas medidas para que se possa extrair o conhecimento sumarizado no perfil. Medidas externas são aquelas que serão usadas para avaliar as diferenças entre o ruído existente no perfil e o ruído da nova transação com relação aos *clusters* existentes no perfil. Medidas internas serão usadas para comparar o conteúdo dos *clusters* do perfil com os itens de uma nova transação que se encaixam nos *clusters* do perfil. Assim, são definidas estatísticas que sumarizam o conhecimento do perfil. O sumário externo do perfil é representado por duas propriedades: a taxa de repetição externa *extratio* e seu desvio padrão *extratioddev* e também a sua distância externa *extdist* e seu desvio padrão *extdistdev*.

Sendo o suporte de transação de um *cluster* C (número de transações diferentes nas quais os itens de C ocorrem sobre o total de transações) representado por $support(C)$. A taxa de repetição externa individual de uma transação é a razão entre o número de objetos que são ruído sobre o total de objetos na transação. Seja o número de *clusters* do k -ésimo sinal denotado por m , a taxa de repetição externa é dada por:

$$extratio^k = \frac{1}{|TD^k|} \cdot \sum_{i=1}^{|TD^k|} \{1 - \sum_{j=1}^m r_i(C_j^k)\} = 1 - \sum_{j=1}^m ratio(C_j^k) \cdot support(C_j^k)$$

A distância externa de um ruído é dada pela distância desse ruído ao *cluster* mais próximo no perfil. Seja ε um item de ruído e C um *cluster*. Para se achar o *cluster* C mais próximo a ε , usa-se o fator de proximidade $f_C(\varepsilon)$, sendo $f_C(\varepsilon)$ diretamente proporcional ao $support(C)$ e inversamente à distância de ε à borda

mais próxima de C. Assim, quanto maior $f_C(\varepsilon)$, mais próxima ε está de C. $f_C(\varepsilon)$ é dada por:

$$f_C(\varepsilon) = \frac{\text{support}(C)}{|\varepsilon - (\min(C) \text{ ou } \max(C))|}$$

Desse modo, achado o *cluster* C mais próximo de ε , a distância de ε a C é:

$$\text{dist}(P^k, o_i^j) = \begin{cases} |o_i^j - \min(C^k)| & \text{se } o_i^j < \min(C^k) \\ |o_i^j - \max(C^k)| & \text{se } o_i^j > \max(C^k) \end{cases}$$

Seja E_i^k o conjunto dos itens que são ruídos na transação $T_i \in TD^k$. Desse modo, a distância externa $ex_d(P^k, T_i)$ de uma transação T_i para o perfil P^k criado a partir do *clustering* dos valores do sinal k é dado por:

$$ex_d(P^k, T_i) = \frac{1}{|E_i^k|} \sum_{j=i}^{|E_i^k|} \text{dist}(P^k, o_i^j) \quad \text{onde } o_i^j \in E_i^k$$

Assim, o sumário da distância externa $extdist^k$ é a média das distâncias externas individuais:

$$extdist^k = \frac{\sum_{i=1}^{|TD^k|} ex_d(P^k, T_i)}{|TD^k|}$$

Como dito anteriormente, são definidas medidas internas e externas para extrair o conhecimento do perfil de um usuário. Portanto, para uma nova transação, são definidas 4 medidas de anormalidade que correspondem às diferenças entre a nova transação e o perfil: diferença interna, razão de repetição interna, diferença externa e razão de repetição externa. Denotamos por $MS = \{ID, IR, ED, ER\}$ o conjunto das medidas citadas anteriormente.

Para que uma nova transação T_v seja comparada com o perfil de um usuário, os itens de T_v que estão na faixa de um *cluster* C^k do perfil do usuário são usados para se medir as diferenças internas. Assim, seja S_v o conjunto dos itens de T_v que estão na faixa do *cluster* C^k . A diferença da distância interna de S_v a um *cluster* específico C^k é definida como a diferença entre a média dos valores de S_v - $avg_v(C^k)$ - e o centro do *cluster* C^k . Como o desvio padrão do centro de cada *cluster* é diferente, essa distância deve ser normalizada pelo desvio padrão, pois assim a distância a um *cluster* específico não influencia a medida total. Do mesmo modo, a diferença interna de razão de repetição deve ser normalizada pelo desvio padrão da razão de repetição $rdev(C^k)$. Portanto, a diferença interna de cada *cluster* C^k a uma nova transação T_v e sua diferença de repetição interna são definidas como:

$$\begin{aligned} \text{ind_diff}_{ID}(C^k, T_v) &= \frac{|\text{center}(C^k) - avg_v(C^k)|}{\gamma \cdot cdev(C^k)} \\ \text{ind_diff}_{IR}(C^k, T_v) &= \frac{|\text{ratio}(C^k) - r_v(C^k)|}{\gamma \cdot rdev(C^k)} \end{aligned}$$

O fator de normalização γ é usado para controlar o efeito da diferença interna. Com um γ pequeno, o efeito da diferença interna é aumentada, de

modo que a diferença interna pode ser modelada mais precisamente. Quando $cdev(C^k)$ ou $rdev(C^k)$ é muito pequeno, a diferença correspondente pode ser muito grande. Para evitar isso, é definido um valor máximo para tais diferenças. Assim, definem-se as diferenças internas de um perfil P^k a uma nova transação T_v como sendo o somatório das diferenças internas de cada *cluster* de P^k à T_v . Porém, como cada *cluster* tem um suporte de transação diferente, devem ser dados pesos diferentes às medidas de cada *cluster*, multiplicando as diferenças pelo suporte de transação de cada *cluster*. Sendo m o número de *clusters* em P^k , definimos as distâncias como:

$$\begin{aligned} \text{diff}_{ID}(P^k, T_v) &= \sum_{i=1}^m \text{ind_diff}_{ID}(C^k, T_v) \cdot \text{support}(C^k) \\ \text{diff}_{IR}(P^k, T_v) &= \sum_{i=1}^m \text{ind_diff}_{IR}(C^k, T_v) \cdot \text{support}(C^k) \end{aligned}$$

As diferenças acima são dadas em relação a um sinal k específico. Para que se possa analisar o nível de anormalidade de uma nova transação com relação a todos os sinais, definem-se as diferenças totais de uma nova transação T_v ao conjunto de perfis de todos os n sinais $\wp = \{P^1, P^2, \dots, P^n\}$ como sendo:

$$\begin{aligned} \text{overall}_{ID}(\wp, T_v) &= \frac{1}{n} \sum_{k=1}^n \text{diff}_{ID}(P^k, T_v) \\ \text{overall}_{IR}(\wp, T_v) &= \frac{1}{n} \sum_{k=1}^n \text{diff}_{IR}(P^k, T_v) \end{aligned}$$

De modo análogo, podemos definir as distâncias externas em relação ao k -ésimo sinal. Define-se E_v^k como sendo o conjunto dos itens da transação T_v que são ruídos em relação ao perfil P^k . Assim, de forma similar às distâncias internas de um perfil do sinal k a uma nova transação T_v , as distâncias externas são:

$$\begin{aligned} \text{diff}_{ED}(P^k, T_v) &= \frac{|\text{extdist}^k - \text{ex_d}(P^k, T_v)|}{\gamma \cdot \text{extdist_dev}^k} \\ \text{diff}_{ER}(P^k, T_v) &= \frac{|\text{extratio}^k - |E_v^k|/|T_v||}{\gamma \cdot \text{extratio_dev}^k} \end{aligned}$$

Do mesmo modo que as diferenças internas, as distâncias externas são limitadas superiormente por um valor máximo definido pelo usuário e também controladas pelo fator de normalização γ . As diferenças externas totais em relação ao conjunto de perfis de todos os sinais são definidas como:

$$\begin{aligned} \text{overall}_{ED}(\wp, T_v) &= \frac{1}{n} \sum_{k=1}^n \text{diff}_{ED}(P^k, T_v) \\ \text{overall}_{ER}(\wp, T_v) &= \frac{1}{n} \sum_{k=1}^n \text{diff}_{ER}(P^k, T_v) \end{aligned}$$

Para se descobrir a taxa de anormalidade de uma nova transação T_v em relação ao conjunto de atividades passadas de um usuário, as transações são classificadas como normal ou anormal de acordo com a média de anormalidade $\Phi_\mu(TD)$ e o desvio padrão da média sd_μ . Sendo $\mu \in MS$ e TD o conjunto de atividades normais passadas de um usuário, a média $\Phi_\mu(TD)$ e seu desvio padrão sd_μ são:

$$\Phi_\mu(TD) = \frac{1}{|TD|} \sum_{i=1}^{|TD|} \text{overall}_\mu(\wp, T_i)$$

$$sd_\mu = \sqrt{\frac{1}{|TD|} \cdot \sum_{i=1}^{|TD|} (\text{overall}_\mu(\wp, T_i) - \Phi_\mu(TD))^2}$$

Desse modo, uma nova transação T_v de um usuário é classificada como:

- normal se $0 \leq \text{overall}_\mu(\wp, T_v) \leq \Phi_\mu(TD) + sd_\mu \cdot \xi$
- anormal se $\text{overall}_\mu(\wp, T_v) > \Phi_\mu(TD) + sd_\mu \cdot \xi$

O fator de detecção ξ definido pelo usuário determina quão estrita a classificação de uma nova transação será efetuada. Quando ele diminui, as transações são analisadas mais estritamente.

Capítulo 5

Alguns experimentos

5.1 Introdução

Para investigar o desempenho e funcionamento do método proposto, foram implementados os algoritmos e usados os dados do laboratório Lincoln¹, descritos abaixo:

The Information Systems Technology Group (IST) of MIT Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA ITO) and Air Force Research Laboratory (AFRL/SNHS) sponsorship, has collected and distributed the first standard corpora for evaluation of computer network intrusion detection systems. We have also coordinated, with the Air Force Research Laboratory, the first formal, repeatable, and statistically-significant evaluations of intrusion detection systems. Such evaluation efforts have been carried out in 1998 and 1999.

Aqui foram usados os dados do ano de 1999, compostos de 5 semanas de monitoramento. Os dados usados foram coletados usando o Solaris BSM² do Solaris 2.6. O BSM é amplamente usado como ferramenta de monitoramento e é certificado como sendo do nível C2. Os sinais considerados nesses testes foram somente as chamadas de sistema. Assim, para a execução de um programas são consideradas quais chamadas de sistemas foram feitas e quantas vezes elas ocorreram.

5.2 Os experimentos

Os dados são compostos de 5 semanas de monitoramento. A primeira e terceira semanas são livres de ataques para fins de treinamento de sistemas baseados em anomalia. A segunda semana tem ataques e foi usada no teste de formatos de arquivos e simulações. A quarta e quinta semanas são os testes realmente efetuados na avaliação feita em 1999.

¹<http://www.ll.mit.edu/IST/ideval/>

²BSM - SunSHIELD Basic Security Module - <http://docs.sun.com/db/doc/802-5757?q=BSM>

Em todos os testes feitos, usamos a primeira semana para a criação do perfil do usuário. O usuário aqui analisado foi somente o de *user id* 0 (root). Para avaliar a taxa de falsos alarmes, usamos também a primeira semana como teste, comparando as transações da primeira semana com o perfil criado a partir dessa. Assim, comparando transações sem ataques com o perfil do usuário, foi possível avaliar a taxa de falsos alarmes do método. Do mesmo modo, comparamos a terceira semana com o perfil da primeira, visto que na terceira semana também não havia ataques.

Para que pudéssemos avaliar a taxa de detecção, fizemos testes comparando as transações da segunda semana em relação ao perfil da primeira. O mesmo foi feito para a quarta e quinta semanas.

5.3 Os resultados

Na figura 5.1 é mostrado o gráfico usando a primeira semana como treinamento e ela própria como teste a fim de avaliar a taxa de falsos alarmes para o conjunto de medidas $MS=\{ID,IR,ED,ER\}$ considerado.

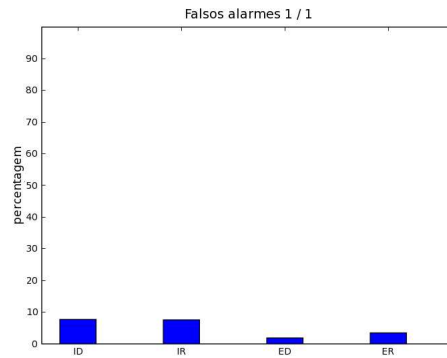


Figura 5.1: Semana 1 com relação à semana 1 - falsos alarmes

A gráfico 5.2 mostra os valores calculados para as transações de teste e o limite para a detecção em tracejado.

No gráfico 5.3 mostramos o teste da terceira semana com relação a primeira, relacionando as duas semanas onde não há anomalias. No gráfico 5.4 mostramos a taxa de falsos alarmes.

É mostrado na figura 5.5 o gráfico da taxa de detecção da segunda semana com relação à primeira e na figura 5.6 o gráfico dos valores e dos limites para a segunda semana. Na figura 5.6 temos também a presença de um gráfico com os ataques ocorridos.

Na figura 5.8 mostramos o gráfico quando usamos a primeira semana como treinamento e a quarta e quinta semanas como testes. Em conjunto com esse gráfico mostramos também um gráfico com os horários de início dos ataques dessas duas semanas.

No gráfico 5.7 mostramos a taxa de detecção do método, usando a primeira semana como treinamento e a quarta e quinta semanas como teste.

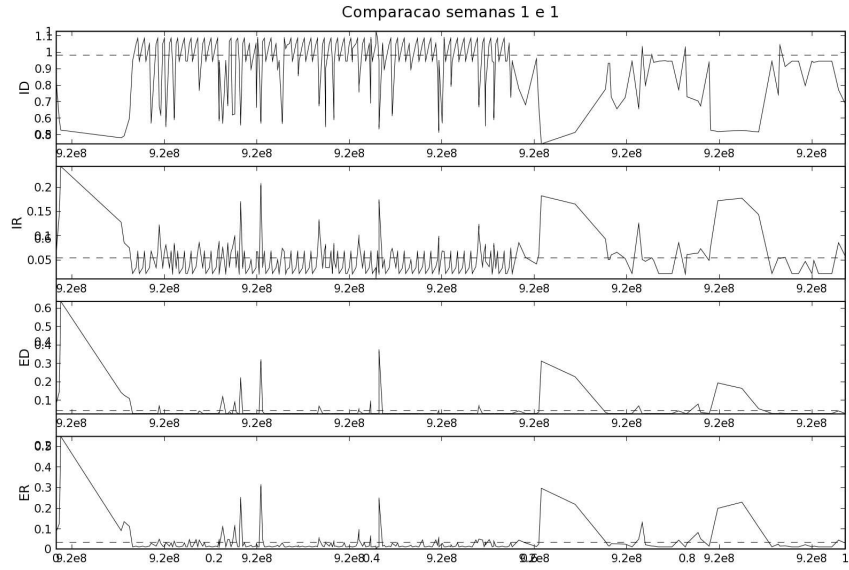


Figura 5.2: Semana 1 com relação à semana 1 - limite e valores

5.4 Conclusões

Podemos perceber, a partir dos gráficos, que o sistema é capaz de detectar anomalias. Seria interessante o estudo com dados mais atuais, abrangendo novos tipos de ataques e a análise a partir de mais usuários, não somente o *root*. Também seria interessante um estudo de como regular os parâmetros de forma correta de acordo com cada usuário. Vemos que o método pode ser eficaz na detecção de intrusões, mas se faz necessário um estudo mais detalhado da técnica e investigação de melhoramentos de modo que possa ser incorporada a um sistema de detecção de intrusões.

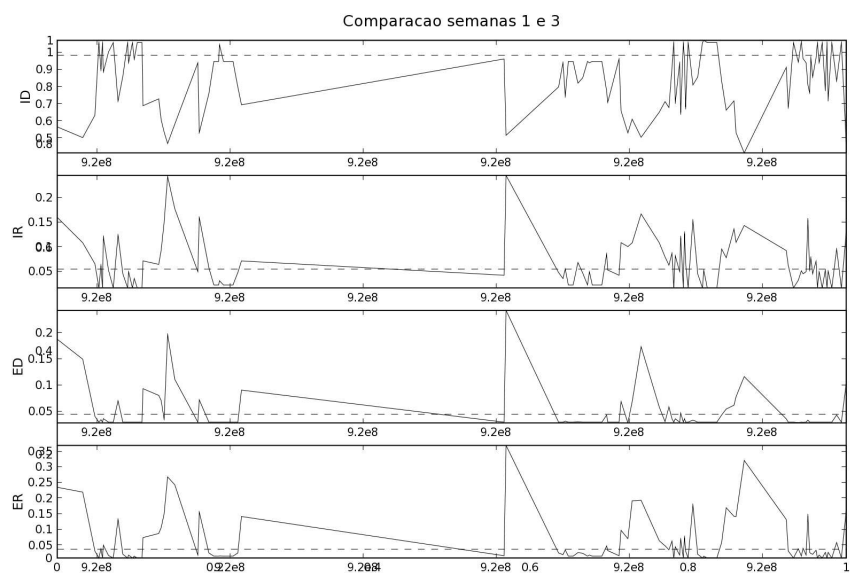


Figura 5.3: Semana 3 com relação à semana 1 - limite e valores

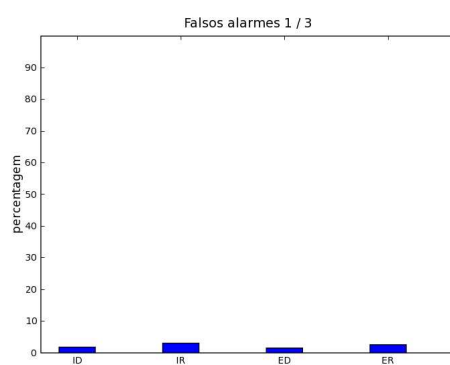


Figura 5.4: Semana 3 com relação à semana 1 - falsos alarmes

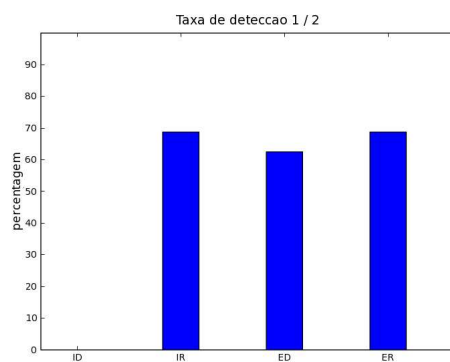


Figura 5.5: Semana 2 com relação à semana 1 - taxa de detecção

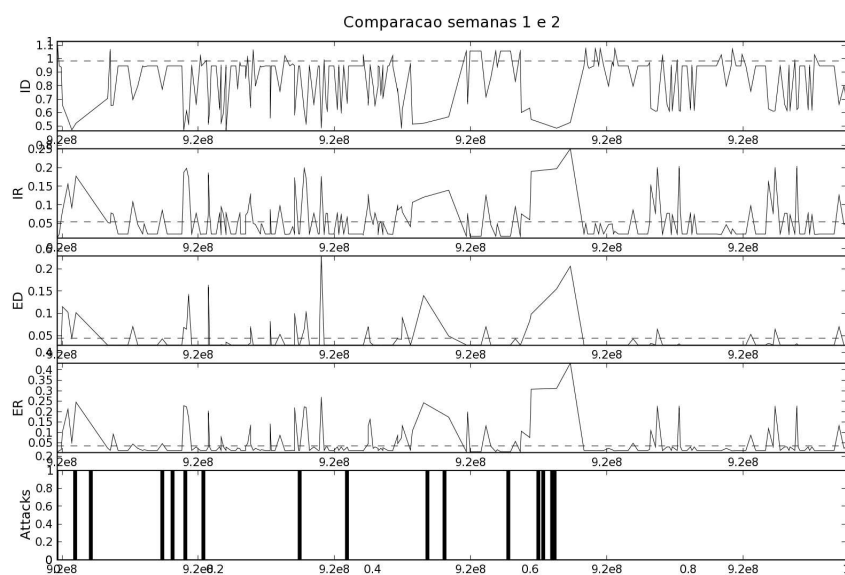


Figura 5.6: Semana 2 com relação à semana 1 - limite e valores

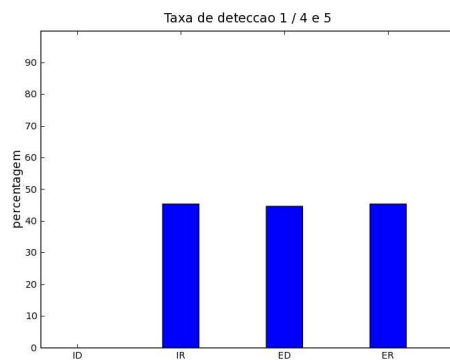


Figura 5.7: Semanas 4 e 5 com relação à semana 1 - taxa de detecção

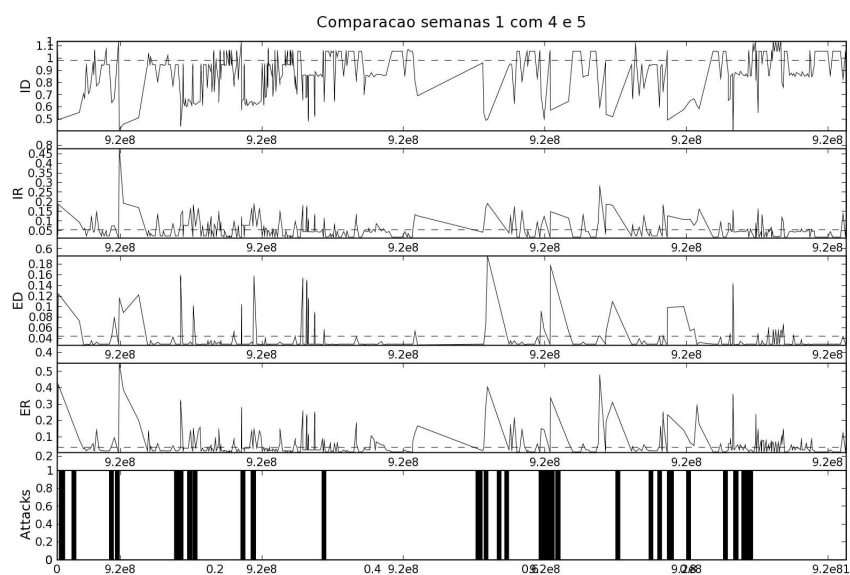


Figura 5.8: Semanas 4 e 5 com relação à semana 1 - limite e valores

Capítulo 6

Trabalhos futuros

Aqui foram feitas algumas investigações e estudos sobre um assunto que, apesar de relativamente novo, está em grande desenvolvimento e tem o interesse da comunidade de segurança na computação. A maioria dos sistemas atuais são baseados em rede e em assinaturas. A proposta para trabalhos futuros seria a criação de um sistema de detecção de intrusão funcional para o sistema operacional GNU/Linux que fosse baseado em anomalia de comportamento de usuários ou programas de modo que pudéssemos validar algumas técnicas recentes da comunidade científica.

Referências Bibliográficas

- [ACF⁺00] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99TR-028, Carnegie Mellon University, 2000. 7, 9
- [And72] James P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, VOL. II, Deputy for Command and Management Systems - HQ Electronic Systems Division, 10 1972. 3
- [cdC01] Lando Mendonça di Carlantonio. Novas metodologias para clusterização de dados. Master's thesis, COPPE/UFRJ, Coordenação de Programas de Pós-Graduação em Engenharia, Aug 2001. 13
- [CER04] CERT. Cert/cc statistics. Technical report, CERT, 1988-2004. 10
- [DDW99] H. Debar, M. Dacier, and A. Wepsi. A revised taxonomy for intrusion-detection systems. Technical Report RZ 3176(93222), IBM Research, 10 1999. 7, 8, 9, 10
- [Den76] Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, 1976. 4
- [EKSX96] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press. 14
- [GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, citeseer.ist.psu.edu/guha98cure.html, 1998. ACM. 13
- [Jó4] Hilário Seibel Júnior. Posicionamento das equipes de campo da escola. Technical report, Universidade Federal do Espírito Santo - UFES, 2004. 14
- [KS74] Paul A. Karger and Roger R. Schell. Multics security evaluation: Vulnerability analysis. Technical Report ESD-TR-74-193, VOL. II, Information Systems Technology Applications Office - Deputy for Command and Management Systems - Electronic Systems Division, 6 1974. 3

- [LS98] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998. 9
- [LS00] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, 2000. 9, 10
- [LX01] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the IEEE Symposium on Security and Privacy*, page 130. IEEE Computer Society, 2001. 8
- [MM04] Tom Mills and Lockheed Martin. Trusted operating systems. Technical report, Carnegie Mellon University, 2004. 4
- [oD85] Department of Defense. Trusted computer system evaluation criteria. Technical Report DoD 5200.28-STD, Department of Defense, 12 1985. 4, 11, 16
- [OL03] Sang Hyun Oh and Won Suk Lee. An anomaly intrusion detection method by clustering normal user behavior. *Computer & Security - Elsevier*, 22(7):596–612, 2003. 16, 17, 19
- [Org99] Information Systems Security Organization. Controlled access protection profile. Technical Report 1.d, National Security Agency, 10 1999. 4