Detecção autônoma de intrusões utilizando aprendizado de máquina

Eduardo Alves Ferreira

Detecção autônoma de intrusões utilizando aprendizado de máquina

Eduardo Alves Ferreira

Orientador: ***Prof. Dr. Rodrigo Fernandes de Mello***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. VERSÃO REVISADA.

USP – São Carlos
Julho/2011

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

Alves Ferreira, Eduardo
A474d Detecção autônoma de intrusões utilizando aprendizado de máquina / Eduardo Alves Ferreira; orientador Rodrigo Fernandes de Mello -- São Carlos, 2011.
88 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2011.

1. Detecção de intrusão. I. Mello, Rodrigo Fernandes de, orient. II. Título.

Agradecimentos

À minha esposa Gislaine pelo apoio e paciência durante a condução desse trabalho.

Aos colegas Cássio M. M. Pereira, Marcelo Keese Albertini, Paulo Henrique Ribeiro Gabriel, Renato Ishii, Ricardo Rios e Vinicius Reis, pelas discussões que auxiliaram na realização do trabalho.

Aos professores e funcionários do ICMC-USP, que auxiliaram direta e indiretamente o trabalho.

Ao professor e amigo Rodrigo Fernandes de Mello, pela excepcional orientação durante este trabalho, e pelo enorme apoio em diversos outros.

Resumo

A evolução da tecnologia da informação popularizou o uso de sistemas computacionais para a automação de tarefas operacionais. As tarefas de implantação e manutenção desses sistemas computacionais, por outro lado, não acompanharam essa tendência de forma ágil, tendo sido, por anos, efetuadas de forma manual, implicando alto custo, baixa produtividade e pouca qualidade de serviço. A fim de preencher essa lacuna foi proposta uma iniciativa denominada Computação Autônoma, a qual visa prover capacidade de autogerenciamento a sistemas computacionais. Dentre os aspectos necessários para a construção de um sistema autônomo está a detecção de intrusão, responsável por monitorar o funcionamento e fluxos de dados de sistemas em busca de indícios de operações maliciosas. Dado esse contexto, este trabalho apresenta um sistema autônomo de detecção de intrusões em aplicações *Web*, baseado em técnicas de aprendizado de máquina com complexidade computacional próxima de linear. Esse sistema utiliza técnicas de agrupamento de dados e de detecção de novidades para caracterizar o comportamento normal de uma aplicação, buscando posteriormente por anomalias no funcionamento das aplicações. Observou-se que a técnica é capaz de detectar ataques com maior autonomia e menor dependência sobre contextos específicos em relação a trabalhos anteriores.

Abstract

The use of computers to automatically perform operational tasks is commonplace, thanks to the information technology evolution. The maintenance of computer systems, on the other hand, is commonly performed manually, resulting in high costs, low productivity and low quality of service. The Autonomous Computing initiative aims to approach this limitation, through self-management of computer systems. In order to assemble a fully autonomous system, an intrusion detection application is needed to monitor the behavior and data flows on applications. Considering this context, an autonomous Web intrusion detection system is proposed, based on machine-learning techniques with near-linear computational complexity. This system is based on clustering and novelty detection techniques, characterizing an application behavior, to later pinpoint anomalies in live applications. By conducting experiments, we observed that this new approach is capable of detecting anomalies with less dependency on specific contexts than previous solutions.

Sumário

Sumário xii
Lista de Figuras xiii
Lista de Tabelas xv
Lista de Algoritmos xvii

1 Introdução **1**
1.1 Motivação e objetivo 3
1.2 Organização da dissertação 7

2 Detecção de intrusão **9**
2.1 Taxonomia de sistemas de detecção de intrusão 9
2.2 Comparações das características dos sistemas 11
2.3 Considerações finais 13

3 Técnicas de aprendizado de máquina **15**
3.1 Agrupamento de dados 17
3.1.1 Algoritmo *K-Means* 19
3.1.2 *K-Means on-line* (*Leader-Follower*) 20
3.1.3 *Leader-Follower* adaptativo 22
3.1.4 Rede *Grow When Required* 22
3.2 Detecção de novidades 22
3.2.1 Janela Deslizante de Hofmeyr e Forrest 22
3.2.2 Entropia da Cadeia de Markov 24
3.2.3 *Dynamic Time Warping* 24
3.3 Considerações finais 25

4 Abordagem proposta **27**
4.1 Caracterização da abordagem proposta 27
4.2 Metodologia 29

5 Avaliação da abordagem proposta **31**
5.1 Descrição dos experimentos 32
5.1.1 Modelo de dados e função de distância 32
5.1.2 Etapa de agrupamento 35
5.1.3 Validação do Agrupamento 36
5.1.4 Detecção de novidades 38
5.1.5 Conjuntos de dados de detecção de intrusão 39
5.2 Resultados 44
5.2.1 Agrupamento de dados 44
5.2.2 Detecção de novidades 46
5.2.3 Desempenho de execução 50
5.2.4 Correlação entre agrupamento e detecção 51
5.3 Considerações finais 54

6 Avaliações adicionais sobre a abordagem proposta **57**
6.1 Ambiente de simulação 58
6.1.1 Simulação de comportamento normal 61
6.1.2 Simulação de incidentes de ataque 62
6.2 Definição dos modelos de dados 64
6.3 Descrição dos experimentos 65
6.4 Resultados 67
6.4.1 Detecção de ataques em chamadas de sistema 68
6.4.2 Avaliação da escalabilidade 70
6.4.3 Detecção de ataques à aplicação *Web* 71
6.5 Considerações finais 74

7 Conclusões **77**

Referências **85**

Lista de Figuras

1.1 Laço de controle de um sistema autônomo (Murch, 2004). 2
1.2 Percentual de incidentes por tipo de ataque (BREACH, 2008). . . 5
1.3 Abordagem de caracterização de comportamento por meio de agrupamento de dados e detecção de novidades. 6

3.1 Abordagem da janela deslizante Forrest et al. (1996). 23

4.1 Proposta para caracterização de comportamento e detecção de novidades. 28

5.1 Distâncias utilizadas para cálculo da silhueta e do erro quadrático. 37
5.2 Aplicação da abordagem para fins de validação. 40
5.3 Pontos de coleta de dados. 44
5.4 Índices de agrupamento para o processo `rpc.statd`. 45
5.5 Índices de agrupamento para o processo `WU-FTPD`. 46
5.6 Correlação entre índices de agrupamento e detecção de novidades. 51
5.7 Curvas ROC para o conjunto de dados `rpc.statd` quando a Entropia é utilizada como índice de novidade. 53
5.8 Curvas ROC `WU-FTPD` quando a Entropia é utilizada como índice de novidade. 54

6.1 Diagrama de instalação e vetor de ataques. 61
6.2 Informações sobre usuário administrador extraídas do sistema. . 63
6.3 Curvas ROC para conjunto de dados de chamadas de sistema. . . 68
6.4 Área sob curvas ROC em relação ao parâmetro k. 69
6.5 Curvas ROC para conjunto de dados DARPA IDEval. 70
6.6 Curvas ROC para conjunto de dados *HTTP*. 71
6.7 Curvas ROC para conjunto de dados XML-RPC. 72
6.8 Curvas ROC para conjunto de dados *SQL*. 73

Lista de Tabelas

2.1 Taxonomia de Sistemas de Detecção de Intrusão 10

3.1 Exemplo de sequência de chamadas de sistema em uma aplicação 17
3.2 Agrupamento das sequências 18

5.1 Trecho do conjunto de dados utilizado. 33
5.2 Exemplo de distância entre atributos. 34
5.3 Parâmetros e valor de retorno da chamada de sistema como atributos do algoritmo de agrupamento (com atributos do tipo cadeia de caracteres, nominal e numérico), após normalização. 34
5.4 `rpc.statd`: DTW / *K-Means*($k = 1$). 47
5.5 `rpc.statd`: DTW / *K-Means*($k = 16$). 47
5.6 `WU-FTPD`: DTW / *K-Means*($k = 1$). 48
5.7 `WU-FTPD`: DTW / *K-Means*($k = 16$). 48
5.8 `rpc.statd`: Incremento na diferença de médias. 49
5.9 `WU-FTPD-A`: Incremento na diferença de médias. 50
5.10 `WU-FTPD-B`: Incremento na diferença de médias. 50
5.11 `WU-FTPD-C`: Incremento na diferença de médias. 51
5.12 `rpc.statd`: Correlação do E.Q. 52
5.13 `rpc.statd`: Correlação da Silhueta. 52
5.14 `WU-FTPD`: Correlação do E.Q. 52
5.15 `WU-FTPD`: Correlação da Silhueta. 53
5.16 Índice de correlação entre área sob a curva ROC e índices de agrupamento. 54

Lista de Algoritmos

1 *K-Means* 19
2 Algoritmo *K-Means on-line* (*Leader-Follower*) 21
3 DTW para múltiplas séries 39

CAPÍTULO

1

Introdução

A redução de custo e popularização de computadores permitiram a automatização de diversas tarefas rotineiras. Em contrapartida, sistemas computacionais continuam sendo operados manualmente, o que agrega alta probabilidade de erro e demanda grande tempo para efetuar intervenções. Como consequência, observa-se alto custo de operação e manutenção, além de baixa produtividade e qualidade de serviço. Visando abordar tais limitações, Murch (2004) apresentou sua iniciativa de Computação Autônoma, cujo objetivo é automatizar a operação de sistemas computacionais. Nessa abordagem, tarefas operacionais são tratadas de forma automatizada, o que permite que analistas e técnicos se encarreguem apenas de tarefas gerenciais, como coordenação de processos e definição de regras que atuam sobre a operação do sistema.

O elemento básico de um sistema autônomo é o laço de controle (Figura 1.1), abordagem fundamentada na Teoria de Controle da engenharia (Astrom e Murray, 2008). Segundo essa abordagem, os componentes de um sistema são classificados em recursos gerenciados e componentes de gerenciamento. Os recursos gerenciados executam tarefas operacionais (como exemplo, um servidor de banco de dados), e os componentes de gerenciamento analisam operações desse sistema e tomam decisões para sua reconfiguração. Por exemplo, considere que um servidor de banco de dados enviou um sinal para um componente de gerenciamento, informando alta carga. Em resposta, o componente envia um sinal de controle para que um gerenciador de recursos de infra-estrutura reserve mais elementos de processamento para atender tal demanda.

Para participar de um sistema autônomo, os recursos gerenciados devem expor sensores e atuadores (Figura 1.1). Sensores são interfaces que exportam

informações sobre qualquer aspecto do recurso que possa ser relevante ao monitoramento de seu funcionamento. Atuadores, por sua vez, permitem que o estado do recurso monitorado seja alterado. Dessa forma, os componentes de gerenciamento são responsáveis por monitorar recursos por meio de sensores e efetuar análises de informações coletadas em busca de eventuais anomalias. No caso de incidentes (i.e. eventos que não fazem parte da operação padrão dos serviços, que causam redução na qualidade de serviço ou comprometem a confiabilidade em um sistema), componentes de gerenciamento devem planejar e executar ações corretivas, utilizando os atuadores para a reconfiguração dos recursos gerenciados.

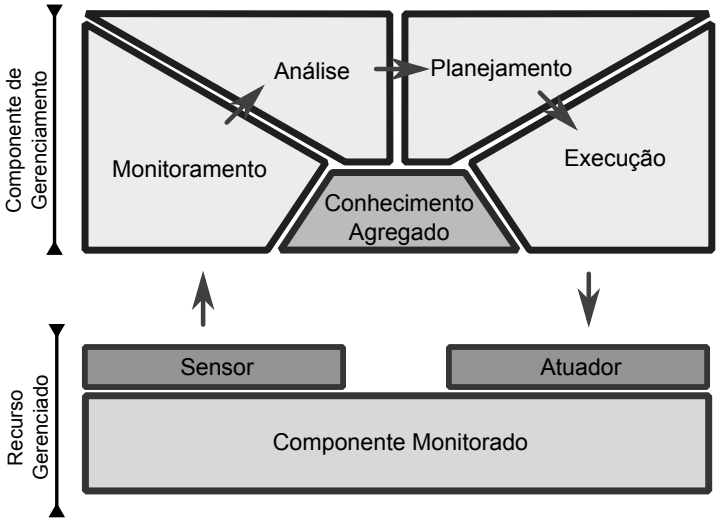


Figura 1.1: Laço de controle de um sistema autônomo (Murch, 2004).

Quatro aspectos principais são abordados pela Computação Autônoma (Murch, 2004): *auto-configuração*, que administra a instalação de novos componentes e integração entre sistemas; *auto-otimização*, que busca adaptar as operações a novas situações, com o objetivo de maximizar a eficiência na execução de tarefas; *auto-cura*, que busca diagnosticar e reparar faltas de sistema; e, por fim, a *auto-proteção*, que visa detectar e intervir sobre operações indevidas, indesejáveis ou maliciosas.

Os aspectos de auto-configuração, auto-otimização e auto-cura têm sido abordados pelo grupo de pesquisa no qual o autor está inserido (Dodonov e de Mello, 2010; Ishii e de Mello, 2009; Pereira e de Mello, 2009), ficando em aberto a auto-proteção, o que motivou o estudo de sistemas de detecção de intrusão (SDI's). Os SDI's monitoram atividades de sistemas computacionais em busca de comportamentos indevidos ou indesejáveis, emitindo alertas em caso de risco. Esse tipo de aplicação cobre as etapas de monitoramento e análise do aspecto de auto-proteção em um sistema autônomo. As etapas de planejamento e execução (que tratam de ações corretivas a serem tomadas após a

detecção dos ataques) estão relacionadas a aspectos operacionais, de forma que essas tarefas costumam ser delegadas a aplicações de infra-estrutura. De qualquer forma, a etapa de detecção é um pré-requisito para esse tipo de resposta, i.e. a etapa de detecção é o primeiro elo em uma cadeia que compõe um sistema de proteção completo.

O monitoramento efetuado por um sistema de detecção de intrusão pode ser baseado em padrões de ataque ou em detecção de novidades (Axelsson, 2000). A primeira abordagem é capaz de reconhecer apenas ataques previamente cadastrados pelo operador do sistema, enquanto a segunda permite a descoberta de novos tipos de ataques. Além disso, a abordagem baseada em padrões tende a ter menor custo computacional, uma vez que a busca por padrões é mais simples que a caracterização do comportamento de um sistema. A busca por padrões de ataques conhecidos é comum em aplicações comerciais, tendo como vantagens maior desempenho e a maior previsibilidade no comportamento do sistema. A busca por anomalias é mais comum em trabalhos acadêmicos, onde tem-se por objetivo automatizar a detecção de ataques desconhecidos.

A detecção de novidades tem sido abordada por meio de técnicas de aprendizado de máquina, mais precisamente por meio de abordagens de detecção de novidades. Detecção de novidades busca identificar padrões ou sinais que não tenham sido observados previamente pela técnica de aprendizado de máquina (Markou e Singh, 2003a,b; Albertini e de Mello, 2009). Da mesma forma, sistemas de detecção de intrusão autônomos caracterizam o comportamento normal de um sistema e identificam novidades em seu comportamento, que são associadas a incidentes de ataque.

1.1 Motivação e objetivo

Grande parte dos trabalhos em detecção de intrusão tem abordado aspectos de segurança de serviços de rede (Twycross e Aickelin, 2010; Kruegel et al., 2003; Forrest et al., 1996; Zanero e Savaresi, 2004; Maggi et al., 2009), uma vez que a integridade dessa camada é de vital importância para a continuidade das operações. Por outro lado, os ataques em nível de aplicação também apresentam grande relevância, principalmente com a popularização dos sistemas *Web*, e apenas trabalhos mais recentes (Criscione et al., 2009) abordam esse tipo de intrusão. A vulnerabilidade mais comum à qual esse tipo de aplicação está sujeita é a injeção de código (OWASP, 2010). Para efetuar um ataque desse tipo, um usuário mal-intencionado forja requisições HTTP, inserindo código executável (como, por exemplo, comandos SQL, LDAP ou *Shell Script*) em parâmetros da aplicação (Cannings et al., 2008). O valor desses parâme-

tros é então utilizado pelo sistema sem ser validado, o que permite que esse usuário execute operações arbitrárias em sistemas que deveriam estar protegidos desse tipo de acesso.

Quadro 1 Exemplo de injeção de código *SQL*

```
http://www.hostname.com/service?userId=123

http://www.hostname.com/service?userId=123'; DROP TABLE USERS; --
```

O Quadro 1 apresenta um exemplo de injeção de código SQL por meio de uma requisição HTTP. Um serviço *Web* aqui denominado *service* recebe como parâmetro um identificador de usuário. A primeira requisição apresenta um uso esperado do serviço, onde o parâmetro contém um identificador de usuário. O valor desse parâmetro é diretamente concatenado a um comando SQL, conforme o Quadro 2, sem qualquer tipo de validação. Quando o valor desse parâmetro é manipulado, recebendo o comando `123'; DROP TABLE USERS; -`, além do comando previsto pela aplicação, é executado um comando que exclui toda a tabela de usuários do sistema. A ausência de uma etapa de validação do parâmetro de entrada permite então que sejam executados comandos arbitrários na base de dados.

Quadro 2 Exemplo de código vulnerável a injeção

```
'SELECT * FROM USERS WHERE USER_ID = ' + $userId
```

A relevância desse tipo de ataque pode ser verificada na quantidade de aplicações vulneráveis e no número de incidentes observados nos últimos anos. Em dois anos, o Centro de Atendimento a Incidentes de Segurança da RNP emitiu alertas de vulnerabilidades de injeção de código em pelo menos 50 aplicações de diversos fornecedores sob diversas plataformas (CAIS, 2005, 2006a,b,c). O relatório de vulnerabilidades críticas do *Open Web Application Security Project* (OWASP, 2010) reporta que o número de vulnerabilidades de injeção de código passou do segundo lugar na classificação de 2007 para o primeiro em 2010. Em relação ao número de incidentes, os relatórios do *Web Application Security Consortium* (WASC, 2007) mostram que em 2007 mais de um terço dos incidentes reportados utilizaram alguma técnica de injeção de código, sendo que a injeção de código SQL representou 20% do total. Em 2008, com o surgimento de algumas vulnerabilidades que permitiram ataques automatizados, a proporção de incidentes envolvendo esse tipo de ataque subiu para 30% (conforme Figura 1.2). Mais de 500.000 *websites* foram vítimas desse tipo de intrusão (BREACH, 2008).

A prevenção a esse tipo de ataque se baseia em práticas de programação segura e em ferramentas de análise de padrões (e.g. GreenSQL (2009)). A

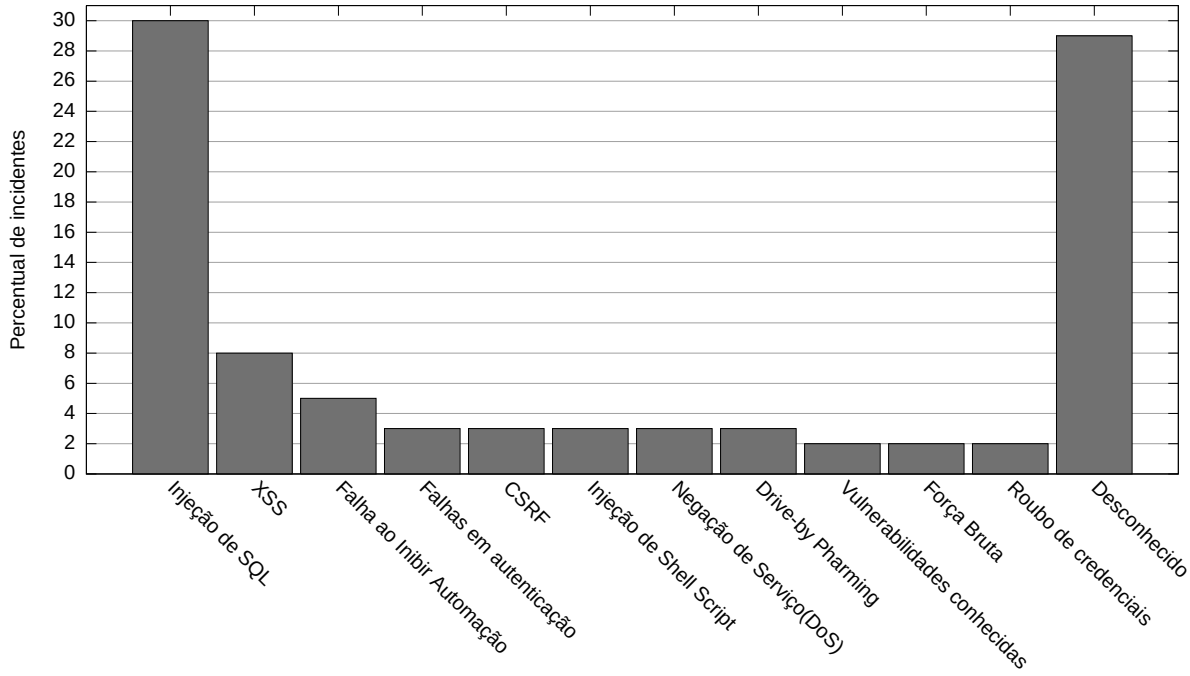


Figura 1.2: Percentual de incidentes por tipo de ataque (BREACH, 2008).

principal limitação da primeira abordagem é a necessidade de intervenção humana para a validação de todo o código de uma aplicação. Entre as limitações da segunda abordagem estão a impossibilidade da descoberta de novos tipos de ataques e a necessidade de definição de padrões para reconhecer possíveis ataques, o que nem sempre é viável. Esse cenário motiva o projeto de um sistema de detecção de intrusão autônomo, baseado em anomalias, capaz de monitorar o funcionamento normal de uma aplicação *Web* e avaliar novidades a fim de coletar evidências para emissão de alertas de intrusão.

Parte dos trabalhos anteriores (Twycross, 2007; Forrest et al., 1996) em detecção de intrusão baseada em anomalias detecta ataques direcionados a serviços de rede (e.g. servidores FTP ou NFS). Trabalhos mais recentes (Criscione et al., 2009) abordam ataques de nível de aplicação, e apresentam bons índices de detecção de ataques ao utilizar técnicas baseadas em modelos, cuja aplicação é específica a um determinado domínio e depende de conhecimento de especialistas para sua aplicação, o que não é adequado para propor um sistema autônomo. Como este trabalho visa detectar ataques a aplicações instaladas em servidores *Web* (i.e. ataques de nível de aplicação) seguindo uma abordagem autônoma, o sistema aqui descrito se baseia em técnicas de aprendizado de máquina (para que o requisito de autonomia seja atendido) com complexidade computacional próxima de linear (para que seja aplicável em sistemas reais).

Para a definição do sistema de detecção de intrusão são utilizadas técnicas de detecção de novidades para a caracterização de comportamentos normais

e para a avaliação de incidentes de ataque. Antes de empregar qualquer uma dessas abordagens, deve-se representar o comportamento de aplicações *Web* (em termos de suas variáveis e comandos executados em sistemas legados, tais como servidores de banco de dados) por meio de séries temporais. No entanto, as informações capturadas nesse tipo de aplicação tendem a ser pouco estruturadas, o que requer um agrupamento prévio para geração de séries que possam ser, posteriormente, analisadas. Esse agrupamento se faz necessário dado que sem essa etapa, pequenas variações de comportamento, esperadas no funcionamento normal do sistema, seriam detectadas como incidentes. A representação de instâncias por meio de identificadores de grupos reduz o volume de memória ocupado pela representação do fluxo de dados, e assim evita a associação de ruídos a intrusões.

A detecção de intrusões é, em seguida, conduzida por meio da identificação de anomalias encontradas nessas séries e que podem apresentar relações com eventos de ataque. Neste trabalho, considera-se uma série temporal como uma sequência de valores nominais ou numéricos, que representa o comportamento de uma de aplicação, i.e. os objetos nessa sequência não necessariamente são coletados em intervalos fixos de tempo, e esse intervalo não é levado em consideração. A Figura 1.3 apresenta a abordagem de agrupamento para geração de uma série temporal, na qual as técnicas de detecção de novidades buscam por anomalias. Nesse contexto, os comandos SQL são submetidos ao algoritmo de agrupamento, que os agrupa a fim de eliminar as pequenas variações que são esperadas no comportamento normal. Os identificadores de grupo são então utilizados para compor uma série temporal, que representa o comportamento da aplicação, e a observação de novidades nessa série é associada a eventos de intrusão. Nesse caso, a novidade é a observação de um novo identificador (3) nunca antes observado na série, gerado a partir de um comando SQL adulterado.

Essa abordagem é avaliada em dois grupos principais de experimentos. O primeiro busca intrusões em um conjunto de dados composto por sequências de chamadas de sistema. Diversas técnicas de aprendizado de máquina são avaliadas, e evidencia-se a consistência da técnica quando se observa alta correlação entre os índices de qualidade das etapas de agrupamento de dados e de classificação. O segundo grupo de experimentos avalia a técnica quando aplicada segundo uma abordagem autônoma, independente do contexto da aplicação. São executados experimentos sobre chamadas de sistemas, sobre *logs* de auditoria de sistema operacional e sobre as saídas de uma aplicação *Web*. Esses experimentos finais evidenciam a validade e a escalabilidade da abordagem, e corroboram a hipótese principal deste trabalho, de que o uso de técnicas eficientes de agrupamento de dados e de detecção de novidades é

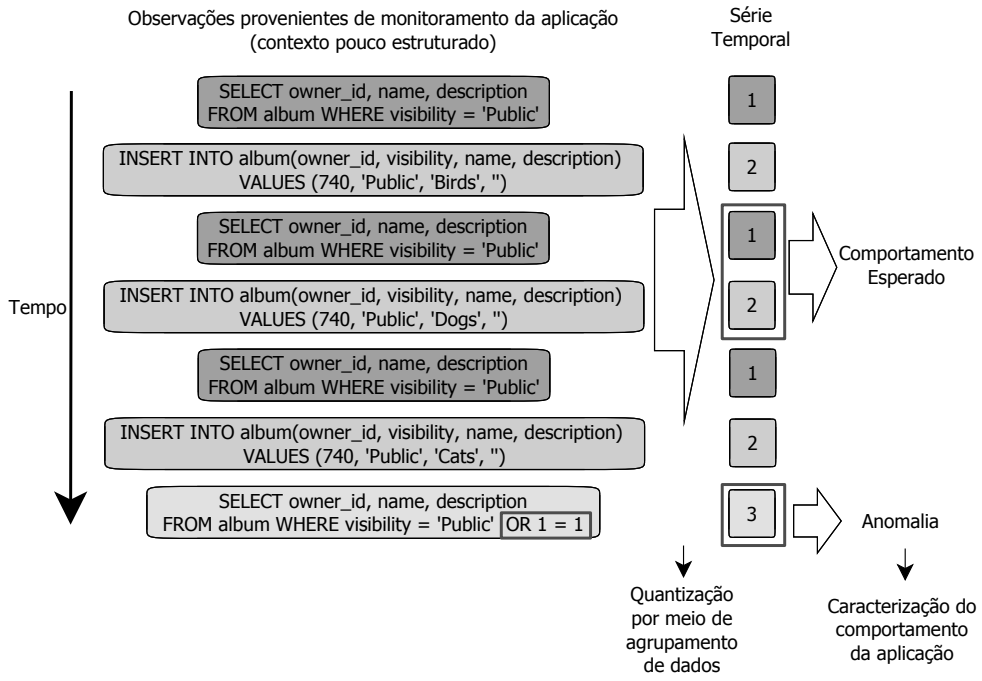


Figura 1.3: Abordagem de caracterização de comportamento por meio de agrupamento de dados e detecção de novidades.

válido para a proposta de um sistema de detecção de intrusão em um contexto pouco estruturado, como o de uma aplicação *Web*.

1.2 Organização da dissertação

O Capítulo 2 apresenta conceitos de detecção de intrusão e descreve uma taxonomia de sistemas de detecção de intrusão. No Capítulo 3 é apresentada uma discussão sobre técnicas de agrupamento de dados e de detecção de novidades que serão utilizadas para o projeto do sistema de detecção de intrusão proposto. No Capítulo 4 são descritas as bases de dados utilizadas na avaliação deste trabalho, além de detalhar a abordagem a ser empregada. O Capítulo 5 apresenta os experimentos iniciais, que avaliam a consistência da técnica quando aplicada a um problema específico, enquanto o Capítulo 6 apresenta os resultados dos experimentos nos quais a técnica é aplicada de acordo com a abordagem proposta neste trabalho. O Capítulo 7 apresenta as considerações finais sobre este trabalho. Por fim, o Apêndice A apresenta uma breve descrição de diversos ataques direcionados a aplicações *Web*.

CAPÍTULO

2

Detecção de intrusão

Sistemas de detecção de intrusão (SDI's) visam identificar e alertar sobre atividades nocivas em sistemas computacionais. SDI é um dos elementos que compõe o aspecto de auto-proteção dentro da área de Computação Autônoma (Murch, 2004). Esse elemento coopera com demais componentes, tais como sistemas de prevenção de intrusão, *firewalls*, criptografia e controle de acesso (Anderson, 2001), a fim de tratar questões relativas à segurança de um sistema computacional.

Diferentemente de ferramentas que restringem o acesso a determinadas informações ou serviços, como criptografia ou *firewalls*, SDI's buscam sinais de violações em serviços expostos a um determinado público. Por exemplo, um servidor *Web* oferece seus serviços por meio do protocolo HTTP, porém esse meio pode ser um vetor de ataques. Nesse caso, um sistema de detecção de intrusão pode monitorar o tráfego HTTP, procurando por traços de ataques e emitindo alertas em situações de intrusão. Dada a importância dos SDI's e sua relação com este trabalho, este capítulo descreve os principais conceitos relacionados a detecção de intrusão.

2.1 Taxonomia de sistemas de detecção de intrusão

Taxonomias dividem os sistemas de detecção de intrusão, tipicamente, em dois grupos principais (Axelsson, 2000): o primeiro, no qual os sistemas buscam por comportamento indevido, previamente conhecido; e o segundo, no qual esses sistemas observam comportamentos em busca de eventos anômalos. Essas técnicas são chamadas, respectivamente, de detecção por assinaturas e detecção de novidades.

A abordagem baseada em assinaturas é comumente utilizada por sistemas comerciais de detecção de intrusão. Esses sistemas contam com bases de assinaturas construídas por especialistas, onde são cadastrados todos os comportamentos indevidos conhecidos de um sistema. Quando um novo ataque é descoberto, especialistas devem adicionar uma regra de detecção à base de assinaturas. A maior vantagem desse tipo de aplicação está relacionada ao baixo número de falsos positivos, e a principal limitação está no fato de que não há qualquer autonomia para a descoberta de novos ataques. Snort e Suricata (Roesch e Green, 2009; Suricata, 2010) são exemplos de *softwares* que utilizam esse tipo de abordagem. Esses *softwares* monitoram o tráfego de rede em busca de padrões de ataques conhecidos a diversas aplicações.

Em contrapartida, a abordagem baseada em anomalias não busca por padrões conhecidos, mas observa o funcionamento do sistema, caracterizando comportamentos "normais" ou "comuns", a fim de disparar sinais de alerta ao observar comportamentos divergentes. Geralmente, essa abordagem utiliza técnicas estatísticas ou de aprendizado de máquina. Sua grande vantagem é a capacidade de detectar intrusões antes desconhecidas, o que viabiliza a criação de sistemas autônomos para detecção de intrusões. Ao mesmo tempo que essa característica apresenta vantagens, ela pode identificar anomalias ao observar comportamentos eventuais ou raros, o que implica na emissão de alertas não necessariamente vinculados a eventos nocivos, o que demanda uma etapa de testes mais cuidadosa para limitar o número de falsos positivos.

Axelsson (2000) define ainda uma terceira categoria denominada "inspirada em assinaturas". Sistemas sob tal classificação são baseados tanto no reconhecimento de assinaturas quanto na observação de anomalias. Para isso, eles geralmente utilizam técnicas semi-supervisionadas de aprendizado de máquina. A Tabela 2.1 apresenta os principais elementos dessa taxonomia.

Tabela 2.1: Taxonomia de Sistemas de Detecção de Intrusão

Baseado em Assinaturas	Estático	Baseado em Estados
		Casamento de *strings*
Baseado em Anomalias	Adaptativo	Com Séries Temporais
		Batch
	Estático	-
Inspirado em Assinaturas	Adaptativo	-

Alguns trabalhos (Scarfone e Mell, 2007) ainda classificam os SDI's de acordo com o contexto monitorado. São descritas duas categorias principais: os *Network-based Intrusion Detection Systems* (NIDS), que monitoram tráfego ou fluxos de rede, e os *Host-based Intrusion Detection Systems* (HIDS), que monitoram o comportamento de processos e fluxos de informação em um único computador.

Os sistemas baseados em assinaturas buscam rastros de ataque, com base em vulnerabilidades conhecidas. Não há o conceito de comportamento normal ou anômalo nessa categoria. Dessa maneira, qualquer variação desconhecida no comportamento do sistema é ignorada. O modelo de assinaturas é estático por definição, evoluindo somente por meio de intervenções do operador.

Uma forma de se construir um sistema desse tipo é por meio de máquinas de estado, onde cada etapa de uma intrusão é mapeada em transições. Outra abordagem emprega o casamento de cadeias de caracteres, onde algum aspecto do sistema (por exemplo, mensagens de rede) é analisado em busca de cadeias de ataque conhecidas.

Conforme descrito anteriormente, sistemas baseados em anomalias não buscam traços de intrusão conhecidos. Eles, na verdade, procuram mapear comportamentos normais e anômalos da aplicação. Esses sistemas são classificados em adaptativos e estáticos.

Sistemas estáticos não utilizam regras e padrões de ataque, mas demandam que o operador do sistema defina eventos irregulares. Por exemplo, a aplicação pode observar o número de tentativas de *login* em um sistema, o número de conexões de rede, ou número de comandos executados, e utilizá-los, em conjunto com ferramentas estatísticas, para definir quais valores observados são considerados anômalos. A aplicação OSSEC (Hay et al., 2008) utiliza essa abordagem para monitorar *logs* de sistema em busca de comportamentos geralmente observados em situações de ataque.

Sistemas adaptativos utilizam técnicas de aprendizado de máquina para definir o comportamento normal de um sistema e identificar suas anomalias. Existem basicamente duas abordagens para esse tipo de técnica: a primeira é baseada na extração de regras ou estatística descritiva para a definição dos comportamentos normais e anômalos do sistema, considerando processamento completo de informações (*batch*). A outra abordagem utiliza modelos mais complexos, considerando informações temporais, por meio de ferramentas como Cadeias de Markov, redes neurais e sistemas imunológicos artificiais. Um exemplo de sistema *batch* é a aplicação AppArmor (AppArmor, 2009), que analisa o comportamento de aplicações a fim de gerar perfis de acessos a recursos do sistema.

Em contrapartida, sistemas inspirados em assinaturas são modelos híbridos baseados em assinaturas e anomalias. Ambas são utilizadas em conjunto a fim de extrair regras de comportamento, que possam ser posteriormente utilizadas, para melhor caracterização de intrusões.

2.2 Comparações das características dos sistemas

As características dos sistemas de detecção de intrusão mapeados pela taxonomia de Axelsson (2000) são definidas em termos do tipo de intrusão que reconhecem, do momento da detecção e do tipo de resposta executada.

O primeiro aspecto a ser abordado é o tipo de ataque que cada SDI é capaz de reconhecer. Podem-se dividir os ataques em três tipos: ataques conhecidos, generalizáveis e desconhecidos. Os ataques conhecidos seguem um padrão rígido de comportamento e são amplamente divulgados, geralmente associados a uma vulnerabilidade específica e executados de forma automatizada. Os ataques generalizáveis são semelhantes aos conhecidos, contando com pequenas variações no seu modo de operação. Por fim, os ataques desconhecidos são aqueles ainda não difundidos, ou que tenham características muito genéricas, o que dificulta sua descrição por meio de uma regra.

Para lidar com ataques conhecidos, os SDI's mais apropriados são os baseados em assinaturas, uma vez que há regras bem definidas para identificar a operação de um ataque e sua eficiência computacional é alta. Por outro lado, a precisão desse tipo de solução deixa a desejar quando se verificam ataques generalizáveis, os quais são melhor identificados por abordagens inspiradas em assinaturas. Por fim, ambas as soluções são inapropriadas para lidar com ataques desconhecidos, que podem ser descobertos por abordagens de detecção de novidades.

Outro aspecto importante é o momento em que a detecção ocorre. Alguns sistemas detectam a intrusão em tempo real ou em tempo próximo de real (i.e. com um pequeno atraso), enquanto outros dependem de processamento após o término completo das operações. Essas abordagens são chamadas, respectivamente, de *on-line* e *batch*. A grande maioria dos sistemas busca soluções *on-line*, uma vez que a quantidade de dados auditada em cenários reais é grande e a detecção no momento exato ou próximo da intrusão é fator determinante para medidas de proteção. Sistemas baseados em assinaturas são geralmente *on-line*, enquanto os baseados em anomalias ou inspirados em assinaturas apresentam tanto soluções *on-line* quanto em *batch*.

O tipo de resposta de um SDI pode ser ativo ou passivo. A maioria dos sistemas de detecção de intrusão é do tipo passivo, que simplesmente detecta ataques e reporta eventos às entidades responsáveis. Alguns sistemas são do tipo ativo, que tomam ações em decorrência da detecção de incidentes (por exemplo, fechando uma conexão de rede quando uma atividade suspeita é detectada). Muitos autores denominam esse tipo de solução como "sistemas de prevenção de intrusão", não os classificando como SDI's, mas sim como sistemas auxiliares.

2.3 Considerações finais

Este capítulo apresentou os principais conceitos e uma taxonomia de sistemas de detecção de intrusão. Esses conceitos e taxonomia são necessários dado o contexto desta dissertação, que visa projetar um sistema autônomo de detecção de intrusões de nível de aplicação, que possa ser utilizado em sistemas *Web*. Para isso, o sistema deve ser baseado em comportamento de processos e fluxos de dados em aplicações, sendo capaz de detectar ataques cujo vetor é a injeção de código em requisições HTTP trocadas entre cliente e servidor. Esse tipo de ataque é bastante genérico, podendo ser executado tanto de forma manual (i.e. sem seguir um padrão rígido) quanto automatizada. Muitas vezes o ataque demanda uma etapa exploratória antes que ocorram danos ao sistema ou às suas informações. Esses ataques são, portanto, do tipo desconhecido, havendo certa tolerância a atrasos em sua detecção. Sendo assim, técnicas ideais para a detecção desse tipo de intrusão são baseadas em anomalias (por não ser possível formular assinaturas dos ataques, dada sua generalidade) e de resposta em tempo real ou tempo próximo de real.

O próximo capítulo discute técnicas avaliadas para a construção do sistema proposto. Inicialmente são abordadas as técnicas de agrupamento de dados, utilizadas na etapa de discretização do comportamento da aplicação. Considerando as necessidades do trabalho, é dado maior foco às técnicas adaptativas e de varredura única do conjunto de dados. Em seguida, são abordadas técnicas de detecção de novidades que avaliam se as séries geradas são referentes a comportamento normal ou anômalo.

CAPÍTULO

3

Técnicas de aprendizado de máquina

A caracterização de comportamento de processos é uma técnica frequentemente considerada para a construção de sistemas de detecção de intrusão. Para a aplicação dessa técnica, é necessária inicialmente a extração de informações sobre o funcionamento de um processo, as quais visam caracterizar seu comportamento normal (e.g. quando a aplicação é executada em um ambiente simulado ou em uma rede isolada). Posteriormente, monitora-se o sistema em ambiente de produção, associando-se novidades observadas a incidentes de ataque.

A fim de caracterizar o comportamento de uma aplicação, duas abordagens principais são descritas na literatura. A primeira utiliza técnicas baseadas em modelos a fim de avaliar as saídas de uma aplicação, ignorando qualquer dependência temporal entre eventos. Por exemplo, Eskin et al. (2002) utilizam técnicas de agrupamento de dados para caracterizar observações de um sistema, e em seguida técnicas de detecção de *outliers* são utilizadas para a detecção de novidades. Já o trabalho proposto por Kruegel et al. (2003) utiliza um modelo baseado em detectores para mapear características dos parâmetros das chamadas de sistema (como comprimento de cadeias de caracteres ou distribuição de caracteres em uma sequência). Observações que não pertençam a esse modelo são consideradas intrusões.

A segunda abordagem considera apenas a dependência temporal entre um número reduzido de observações. Os procedimentos de uma aplicação geram uma sequência de objetos (denominados eventos) quando executados, e, segundo essa abordagem, essa sequência de objetos é o que define o comporta-

mento de uma aplicação. Como exemplo, Forrest et al. (1996) utilizam uma janela deslizante para observar as sequências de chamadas de sistema efetuadas por uma aplicação, enquanto Pereira e de Mello (2009) estimam a Entropia de Cadeias de Markov, geradas a partir dessa sequência, como indicador de novidades.

As maiores limitações dessas abordagens são, primeiro, o fato delas ignorarem algum aspecto da aplicação, e segundo, o fato de serem fortemente dependentes de um determinado modelo de dados. Esse modelo de dados é geralmente específico para um determinado tipo de aplicação, o que limita a proposta de uma solução totalmente autônoma. Em contrapartida, uma terceira abordagem, proposta por Zanero e Savaresi (2004), é baseada tanto em técnicas de agrupamento de dados quanto em análise de séries temporais. Com isso, a técnica considera os dois aspectos do comportamento de uma aplicação (i.e. a multidimensionalidade das observações do sistema e a dependência temporal entre eventos), além de ser aplicável de forma independente do modelo de dados. Essa abordagem também é utilizada por Maggi et al. (2009) para avaliar os parâmetros das chamadas de sistemas de diversas aplicações UNIX, onde uma técnica de agrupamento hierárquico é utilizada em um estágio inicial, e Cadeias de Markov são utilizadas para avaliar o grau de novidade nas sequências de teste.

A maior parte dos trabalhos que segue essa terceira abordagem apresenta, porém, duas grandes limitações: em primeiro lugar, esses trabalhos não utilizam algoritmos de varredura única, apresentando soluções de alto custo computacional, o que é inviável para um sistema real. A segunda limitação é que esses trabalhos utilizam funções específicas para um determinado contexto de aplicação, o que limita a proposta de um sistema totalmente autônomo, ou a aplicação da técnica em contextos mais elaborados.

Dado esse contexto, este capítulo apresenta as principais técnicas de aprendizado de máquina capazes de tratar essas limitações. São abordados inicialmente tópicos em agrupamento de dados aplicáveis ao contexto proposto, e em seguida são apresentadas técnicas de detecção de novidades que podem ser utilizadas para a caracterização de sistemas e detecção de novidades no seu comportamento. Este trabalho tem uma forte relação com a área de fluxo de dados (Babcock et al., 2002), dado que a caracterização de comportamento de sistemas gera uma grande quantidade de informações que não pode ser mantida em armazenamento secundário. Dessa forma, é necessário que os algoritmos tenham não apenas complexidade próxima de linear, mas que sejam capazes de operar em uma única varredura sobre o conjunto de dados, i.e. processando cada observação no momento em que é produzida (ou em período próximo ao da sua geração), mantendo em memória apenas uma pe-

quena fração das informações observadas.

3.1 Agrupamento de dados

Agrupamento de dados é uma ferramenta para a segmentação de conjuntos de dados que não depende de conhecimento estrutural ou categorização prévia (Jain e Dubes, 1988). Diferentemente das técnicas de aprendizado supervisionado, o objetivo do agrupamento de dados é obter uma representação conveniente de um conjunto de dados, sem estabelecer regras para a categorização dos mesmos. Tendo um conjunto de objetos como entrada, essas técnicas geram, como saída, subconjuntos onde elementos similares tendem a ser dispostos em um mesmo grupo. A definição de grupo é subjetiva por natureza, fato evidente nos resultados distintos provenientes de diferentes técnicas de agrupamento aplicadas a um mesmo conjunto de dados (Xu e Wunsch, 2008).

Os primeiros passos para aplicação de um algoritmo de agrupamento são a definição da representação dos dados e a definição de uma função de similaridade entre elementos. Como exemplo, em um sistema de caracterização de comportamentos de processos tem-se uma base de dados de sequências de chamadas de sistemas Linux (onde dados são representados como na Tabela 3.1). Podem-se definir conjuntos de dados contendo os registros referentes a uma mesma chamada, onde os parâmetros das chamadas são vistos como dados em uma matriz de padrões, que são submetidos ao algoritmo de agrupamento.

Tabela 3.1: Exemplo de sequência de chamadas de sistema em uma aplicação

...
brk(*0x00*)
brk(*0x804db04*)
open("*/var/lib/nfs/state*", *O_RDWR*|*O_CREAT*, *0600*)
socket(*PF_INET*, *SOCK_DGRAM*, *IPPROTO_UDP*)
brk(*0x804e000*)
...

Conforme se observa ta Tabela 3.2, a partir desse conjunto de chamadas, tem-se o objetivo de agrupar as observações similares (como as que se observa na segunda e na quinta linha da tabela), e diferenciar as observações distintas, gerando uma série temporal que represente o comportamento do sistema.

Diversas abordagens existem para a definição da função de similaridade entre elementos do conjunto, tais como distância Euclidiana ou de *Manhattan* (Jain e Dubes, 1988). Para a distância entre atributos nominais, pode-se utilizar a regra do agrupamento simples (Jain e Dubes, 1988) ou uma medida

Tabela 3.2: Agrupamento das sequências

...	-
brk(0x00)	1
brk(0x804db04)	2
open("/var/lib/nfs/state", O_RDWR|O_CREAT, 0600)	3
socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)	4
brk(0x804e000)	2
...	-

baseada na frequência de um determinado atributo (Goodall, 1966). Para objetos com tipos mistos (i.e. com atributos numéricos e nominais) diversos trabalhos (Gupta et al., 1999; Huang, 1998) utilizam uma função, como a apresentada na Equação 3.1, para agregar valores, onde d_c são distâncias entre atributos contínuos, d_n são distâncias entre atributos nominais e γ é um parâmetro de peso, variando de zero a infinito, que reflete a importância dos atributos nominais sobre a função de distância. Pode-se, também, utilizar alguma técnica de normalização, para que as distâncias dos atributos com grande variância não dominem a função de similaridade entre elementos.

$$d = \sum d_c + \gamma \sum d_n \tag{3.1}$$

Quando o conjunto de dados em que se trabalha é pouco estruturado, ou tem-se pouca informação sobre sua estrutura, o uso da distância *Normalized Compression Distance* (NCD) (Cilibrasi e Vitányi, 2005) é uma alternativa à definição de uma função de distância específica. A distância NCD pode ser aplicada sobre dados de qualquer tipo, representados como cadeias de *bytes*, uma vez que essa distância utiliza algoritmos de compressão para estimar a complexidade de Kolmogorov (Kolmogorov, 1965) condicional dessas sequências (valor cujo cálculo é incomputável). Essa estimativa é obtida pela Equação 3.2, onde $C(x)$ e $C(y)$ são os tamanhos, em *bytes*, das cadeias geradas pela compressão das cadeias x e y, e $C(xy)$ é o tamanho em bytes da cadeia gerada pela compressão da cadeia oriunda da concatenação das cadeias x e y.

$$NCD(x,y) = \frac{C(xy) - min(C(x), C(y))}{max(C(x), C(y))} \tag{3.2}$$

Apesar da grande facilidade de uso, a distância NCD pode não ser útil em qualquer contexto, principalmente naqueles bem estruturados onde funções de distâncias específicas possam ser definidas com maior facilidade. Além disso, o cálculo dessa função tem alto custo computacional (dada a necessidade de serializar e comprimir objetos para seu cálculo). Cebrián et al. (2005) ainda observam que o cálculo dessa distância apresenta problemas quando o tamanho dos objetos é maior que as janelas dos algoritmos de compressão

(que são da ordem de $32KBytes$ para o algoritmo *gzip* e entre 50 e $450KBytes$ para o algoritmo *bzip2*).

O próximo passo do procedimento de agrupamento é a definição do algoritmo mais adequando ao problema em questão. Esse algoritmo deve ser aplicado sobre o conjunto de dados, a fim de extrair grupos, e realizar, posteriormente, o processo de validação. Por fim, os resultados de agrupamento devem ser interpretados por um analista, ou, no caso deste trabalho, submetidos a uma segunda etapa de processamento que visa buscar anomalias no comportamento do sistema.

Algoritmos de agrupamento podem ser do tipo hierárquico ou particional (Jain e Dubes, 1988). Algoritmos hierárquicos processam dados e agrupam elementos par-a-par, ou particionam o conjunto de dados de forma iterativa. Uma das vantagens desses algoritmos é que uma de suas saídas é um dendrograma, que permite boa visualização de dissimilaridades entre elementos, uma vez que essa abordagem organiza os objetos de maneira hierárquica. A limitação dessa abordagem está na complexidade computacional: algoritmos clássicos têm ordem $O(n^2)$ e possíveis otimizações não apresentam tempo linear de processamento, não sendo em geral aplicáveis em problemas de varredura única. Algoritmos particionais, por outro lado, agrupam elementos por meio de uma heurística que visa minimizar uma determinada função objetivo. Apesar de não gerar uma estrutura hierárquica, os algoritmos particionais apresentam a vantagem de executar em tempo mais próximo de linear. Para o problema deste trabalho, onde a técnica de agrupamento é utilizada apenas para a rotulação de dados, e os conjuntos de dados são grandes, não se justifica o uso de algoritmos hierárquicos. Dessa forma, analisam-se, a seguir, apenas algoritmos particionais.

3.1.1 Algoritmo K-Means

K-Means (MacQueen, 1967) é um dos algoritmos de agrupamento de dados mais conhecidos e populares (Xu e Wunsch, 2008), dada sua simplicidade, fácil implementação e convergência a ótimos locais após poucas iterações. Inicialmente, esse algoritmo divide o espaço do problema em grupos arbitrários, por meio de k centróides posicionados de maneira aleatória (onde cada ponto pertence ao centróide mais próximo). Inicia-se, então, um procedimento de otimização iterativo que visa diminuir a dispersão entre objetos pertencentes a um mesmo grupo. O procedimento é descrito no Algoritmo 1.

Algoritmo 1: *K-Means*

1 Posicione k centróides no espaço do problema;
2 Associe cada objeto ao centróide mais próximo;
3 Recalcule todos os centróides por meio de uma média de cada grupo;
4 Retorne ao passo 2 até que os grupos se estabilizem;

Quando existem atributos nominais no conjunto de dados, o valor da moda desses atributos pode ser utilizada (Gupta et al., 1999) no lugar da média (linha 3 – Algoritmo 1). A complexidade desse algoritmo é de ordem $O(nkdt)$, onde n é o número de elementos no conjunto, k é o número de centróides, d é a dimensão do conjunto de dados e t é o número de iterações. Como d e k são constantes, geralmente muito menores que n, e quando se sabe que o valor de t tende a ser pequeno, pode-se dizer que o algoritmo é linear em relação ao número de elementos. Porém, esse algoritmo apresenta algumas limitações, entre elas a convergência para soluções ótimas locais, a necessidade da definição do parâmetro k e a necessidade de execução em modo *batch*. O problema da convergência a ótimos locais pode ser tratado de diversas formas, como múltiplas inicializações ou com o uso de algoritmos genéticos. Já a necessidade da definição do parâmetro k pode ser tratada com métodos de validação de grupos. Não é possível, por outro lado, eliminar a necessidade da execução em modo *batch*, pois o algoritmo deve percorrer a base de dados t vezes, ou seja, os dados precisam estar armazenados em algum dispositivo de memória secundária. Isso pode não ser possível na caracterização do comportamento de aplicações, uma vez que a quantidade de dados gerada tende a ser grande. Dessa forma, esse algoritmo será utilizado, neste trabalho, apenas para comparação de precisão e eficiência em relação a outros algoritmos.

3.1.2 K-Means on-line (Leader-Follower)

Uma implementação alternativa do algoritmo *K-Means* é descrita por Xu e Wunsch (2008) e trabalha de forma incremental, possibilitando o agrupamento de dados por meio de uma única varredura, eliminando a necessidade de armazenamento de informações. O parâmetro de entrada desse algoritmo não é o valor de k (número de grupos), mas sim um θ que representa a distância mínima necessária para a criação de centróides adicionais. O procedimento é descrito no Algoritmo 2, onde η é o fator de aprendizado, que controla a movimentação do centróide em direção ao elemento observado.

Algoritmo 2: Algoritmo *K-Means on-line* (*Leader-Follower*)

```
1  para cada objeto O faça
2      Calcule a distância de O a todos os centróides;
3      Escolha o centróide (C) com menor distância (D_min) de O;
4      se D_min < θ então
5          C ← C + η(O − C);
6      fim
7      senão
8          Crie um novo centróide na posição O;
9      fim
10 fim
```

Nesse procedimento, uma única varredura do conjunto de dados é efetuada. Procura-se, para cada objeto observado, o centróide que lhe seja mais próximo. Se a distância entre esses o objeto e esse centróide for menor que o parâmetro θ agrega-se o objeto ao grupo do centróide. Se a distância for maior, é gerado um novo grupo utilizando o objeto observado como centróide.

O valor de η varia de zero a um: se $\eta = 0$, o centróide não se adapta às novas observações, e se $\eta = 1$, o centróide é movido diretamente para o ponto observado. Para valores intermediários, o novo centróide é um ponto no segmento de reta que une o novo elemento ao centróide. É interessante observar que, nesse algoritmo, a importância de observações passadas decai exponencialmente, conforme novas observações são adicionadas ao grupo, i.e. existe um fator de "esquecimento". Para eliminar esse fator, pode-se utilizar o valor da média das observações (ou o valor da moda, para atributos nominais) como centróide.

As maiores limitações desse algoritmo são a sensibilidade à ordem dos elementos e a necessidade da definição do parâmetro θ. Além disso, a criação *on-line* de centróides requer a adição de um mecanismo de *buffer* ao algoritmo, caso múltiplas inicializações sejam necessárias. Esse mecanismo permite a apresentação dos elementos com certa aleatoriedade. A definição do valor de θ também é fundamental para um bom resultado do algoritmo: quando θ é muito grande (i.e. quando seu valor é maior que a distância entre quaisquer elementos do conjunto de dados) apenas um grupo é constituído, contendo todos elementos do conjunto. Quando seu valor é muito pequeno (i.e. menor que a distância entre os dois elementos mais próximos do conjunto) o algoritmo cria um grupo para cada objeto (i.e. $k = n$).

3.1.3 *Leader-Follower adaptativo*

A dificuldade na definição do parâmetro θ do algoritmo *Leader-Follower* motiva a proposta de uma abordagem adaptativa para a definição desse parâmetro. É desejável que, ao invés de se definir o valor de θ, defina-se o valor de k (i.e. número de grupos) e que o valor de θ adapte-se, automaticamente, aos valores de distâncias observados.

Uma adaptação do algoritmo *Leader-Follower* (Charikar et al., 1997), inspirada na rede neural $ART2A$ (He et al., 2004), pode ser aplicada: além de k, o algoritmo recebe um parâmetro α, que funciona como mecanismo de tolerância. Quando o número de grupos observados ultrapassar $(1 + \alpha)k$, executa-se uma operação de redução do número de grupos, em que se buscam os dois centróides com maior similaridade, os quais são então fundidos em um único ponto. Esse procedimento se repete até que o número de grupos volte a ser k. Esse procedimento resolve o problema para θ subestimado. Para o problema de θ superestimado, um decaimento no seu valor inicial pode ser aplicado.

3.1.4 *Rede Grow When Required*

A rede neural *Grow When Required* (GWR), proposta por Marsland et al. (2002), é capaz de agrupar dados e de se adaptar a variações nas informações por meio de uma técnica semelhante ao *threshold* do algoritmo *Leader-Follower*. Além da adaptação com criação de novos nós apenas quando os existentes não são válidos, essa rede conta com uma funcionalidade de "esquecimento", onde nós isolados ou de pouca ativação são eventualmente descartados. Essa rede é capaz de detectar novidades em observações, dado que cada nó possui um contador de ativações. Para isso, considera-se que a ativação de nós raramente visitados e a criação de novos nós representam novidades no sistema.

3.2 Detecção de novidades

Técnicas de detecção de novidades buscam identificar variações ou padrões não observados em séries temporais (Albertini e de Mello, 2009). Algumas abordagens para detecção de novidades são apresentadas a seguir.

3.2.1 Janela Deslizante de Hofmeyr e Forrest

A Janela Deslizante de Hofmeyr e Forrest (Forrest et al., 1996; Hofmeyr et al., 1998) é um método simples e computacionalmente eficiente para caracterizar uma série temporal. Essa abordagem é baseada em aprendizado supervisionado: na etapa de treinamento uma série é observada por meio de uma

janela, e sequências observadas são mantidas em uma estrutura de árvore. Na etapa de testes, as sequências observadas são procuradas na estrutura de árvore. Caso uma janela não seja encontrada, é calculada a menor distância Euclidiana dessa janela em relação às observações presentes na árvore. Dividindo-se o valor dessa distância pelo tamanho da janela, tem-se o grau de novidade naquele ponto.

Hofmeyr e Forrest (Forrest et al., 1996; Hofmeyr et al., 1998) apresentam e avaliam a abordagem da janela deslizante para detecção de novidades no comportamento de processos computacionais. Nesses trabalhos, as sequências das chamadas de sistema são observadas durante o funcionamento normal da aplicação, com o uso de uma janela deslizante de tamanho fixo. Essas janelas são utilizadas para definir uma árvore *trie* (i.e. árvore de prefixos) com todas as possíveis sequências de chamadas observadas no comportamento normal do sistema. A Figura 3.1 mostra sequências de chamadas de sistemas observadas dentro de uma janela e mantidas em uma estrutura de árvore. Todas as sequências de nós geradas a partir da raiz até as folhas representam ordenações de chamadas observadas durante o funcionamento normal do sistema. A etapa de treino carrega todas as sequências válidas e a etapa de teste busca por ordens de chamadas que não pertençam ao conjunto de treinamento.

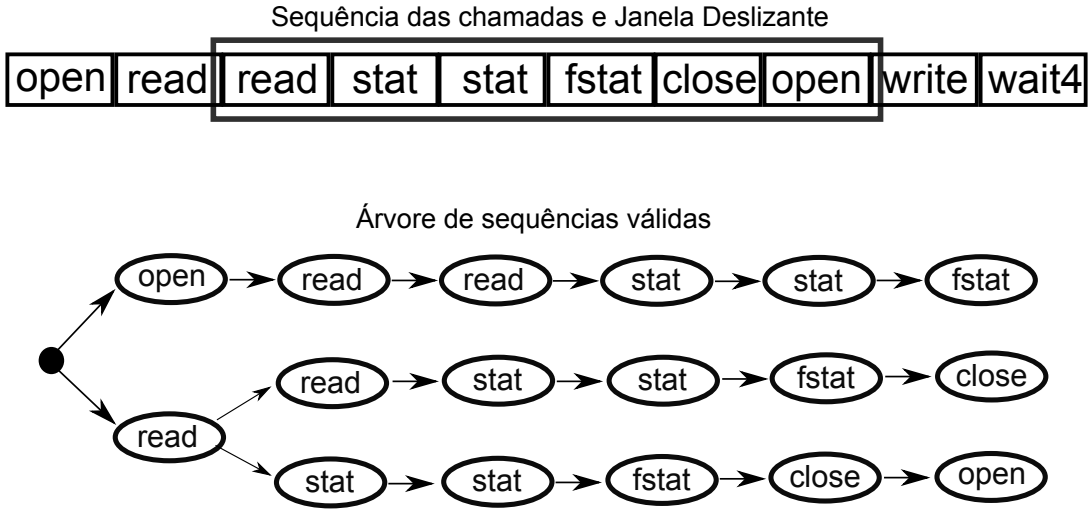


Figura 3.1: Abordagem da janela deslizante Forrest et al. (1996).

Experimentos conduzidos por Forrest et al. (1996) utilizam janelas de tamanho variado. Nesse contexto, os autores observaram o comportamento das aplicações `sendmail` e `lpr`. Os parâmetros das chamadas e os valores de retorno não são utilizados. Para a caracterização do comportamento normal do sistema foram executados testes com dados artificiais (e.g. para

teste do `sendmail` inicializa-se o servidor e diversas mensagens são enviadas). Para a caracterização do comportamento anômalo, foram utilizados *scripts* de ataques conhecidos (efetuando intrusões com e sem sucesso) e erros de configuração que levavam a estados indesejáveis (como repetição infinita de repasse de mensagem entre servidores).

Hofmeyr et al. (1998) ainda estenderam esse trabalho com o uso de bases de dados provenientes de sistemas reais (i.e. com uma quantidade muito maior de chamadas de sistema, avaliando a escalabilidade da abordagem), e também avaliaram essa abordagem com o comportamento da aplicação `WU.FTPD` (i.e. servidor FTP). Nesse cenário, eles utilizaram distância normalizada de Hamming para detectar anomalias, o que resultou em uma caracterização mais precisa do comportamento do sistema. Os testes em ambos trabalhos mostraram que curtas sequências de chamadas de sistema são suficiente para discriminar o comportamento de processos distintos, além de diferenciar funcionamento normal de indesejável. Além disso, o sistema apresentou eficiência computacional e capacidade de escalabilidade adequadas à tarefa de monitoramento *on-line*: o consumo de memória é de ordem $O(Nk)$ (sendo N o tamanho da janela e k o número de chamadas de sistemas existentes), o que permite utilizar o sistema mesmo sobre longas bases de dados oriundas de sistemas em produção.

3.2.2 Entropia da Cadeia de Markov

A dissimilaridade entre séries temporais pode ser expressa pela diferença do valor da Entropia da Cadeia de Markov gerada a partir dessas séries (Albertini e de Mello, 2009; Pereira e de Mello, 2009). Considerando-se cada valor na série como um estado em uma Cadeia de Markov, pode-se estimar as probabilidades de transição entre estados a partir de sequências de valores observados. Calcula-se, então, a Entropia da Cadeia de Markov (Equação 3.3, onde C é o conjunto de todos os estados da cadeia, e $p(i,j)$ é a probabilidade de transição do estado i para o estado j). Espera-se que a diferença no valor da Entropia de séries similares seja pequena, e a derivada desse valor em relação ao tempo representa o grau de novidade em determinado momento.

$$H = -\sum_{i \epsilon C} \sum_{j \epsilon C} p(i,j) log_2(p(i,j)) \tag{3.3}$$

3.2.3 Dynamic Time Warping

Dynamic Time Warping (DTW) (Keogh e Ratanamahatana, 2005) é uma técnica utilizada com sucesso para o alinhamento de série fora de fase. Essa técnica utiliza um algoritmo de programação dinâmica, semelhante ao da dis-

tância Levenshtein (ou *Edit distance*), entre duas séries de tamanho n e m, $n \leq m$. Essa técnica foi utilizada com sucesso por Pereira e de Mello (2009) para a caracterização do comportamentos de processos no sistema `UNIX`. Para a caracterização de sistemas, geram-se diversas séries referentes a um determinado processamento: quanto maior o valor da DTW entre duas séries, maior é a dissimilaridade entre elas, o que pode ser utilizado para caracterizar uma anomalia.

A grande limitação dessa técnica é o fato do algoritmo ser de ordem $O(n)$ em relação a consumo de memória, e $O(nm)$ (i.e. quadrático) em relação a tempo de execução. De qualquer forma, esse algoritmo tem sucesso desde que o tamanho das séries seja limitado, e pode ser utilizado para fins de comparação com algoritmos *on-line*.

3.3 Considerações finais

Neste capítulo foram apresentados diversos conceitos úteis para a caracterização do comportamento de aplicações. Diferentes ferramentas podem ter aplicabilidade e eficiência distintas dependendo do problema analisado, de forma que uma análise crítica dessas ferramentas é essencial para a tomada de qualquer decisão de projeto.

Diversos outros algoritmos incrementais voltados a agrupamento em fluxos de dados são descritos na literatura (como por exemplo nos trabalhos de Aggarwal et al. (2003) e Aggarwal et al. (2004)), e esses poderiam ser avaliados para a abordagem desse trabalho. Optou-se por não utilizar esses trabalhos nesta dissertação dado que, em primeiro lugar, seria inviável a análise de todos esses algoritmos, e em segundo lugar, o fato de que o objetivo deste trabalho é mostrar a viabilidade da abordagem mesmo sem a definição de algoritmos mais sofisticados.

CAPÍTULO

4

Abordagem proposta

O objetivo deste trabalho é projetar e avaliar um sistema autônomo de detecção de intrusões em aplicações *Web*, baseado em técnicas de aprendizado de máquina e computação bio-inspirada. Deseja-se que esse sistema atue de forma autônoma, i.e., que seja capaz de caracterizar intrusões sem o uso de assinaturas de ataques, ou seja, por meio de anomalias observadas durante o seu funcionamento. Avalia-se inicialmente a hipótese de que técnicas de agrupamento são capazes de gerar séries temporais que caracterizam o comportamento normal de um sistema e de distingui-lo de comportamentos anômalos observados em situações de ataque. Neste capítulo são caracterizadas tanto a abordagem proposta, quanto a metodologia utilizada para sua avaliação.

4.1 Caracterização da abordagem proposta

O projeto de sistema de detecção de intrusão baseado em anomalias demanda a caracterização de aspectos relevantes do sistema em estudo. Por exemplo, em trabalhos anteriores (Twycross, 2007; Forrest et al., 1996), onde se buscavam intrusões em serviços de rede, a sequência de chamadas de sistema é definida como o principal aspecto a ser observado, dado que ataques alteram seu comportamento. Por outro lado, essa caracterização não é válida para aplicações *Web*, uma vez que o comportamento de um servidor HTTP em situações de injeção de código não tende a diferir de situações normais. O que pode ser observado, nesse contexto, são as mensagens trocadas entre cliente e servidor, os valores de variáveis da aplicação, além das sequências de comandos executados pelos sistemas legados (tais como Sistemas de Gerenciamento de Banco de Dados – SGBD ou serviços de diretório como *Lightweight Directory*

Access Protocol – LDAP).

Seja qual for o aspecto observado, a abordagem proposta neste trabalho aplica um processo de discretização (que pode ser visto como uma função na forma $f : Objeto \rightarrow \mathbb{N}$) a fim de reduzir a dimensionalidade dos dados, ou para caracterizar um aspecto pouco estruturado da aplicação (como registros em *log* de aplicações, ou mensagens trocadas pela rede). Para essa tarefa são utilizadas técnicas de agrupamento de dados, que recebem como entrada elementos observados no sistema (e.g. comandos executados nos SGBD's ou valores de variáveis) e geram como saída um identificador de grupo ao qual esse elemento pertence. Dessa forma, esse procedimento gera uma série temporal composta por identificadores de grupo. Essa série é então analisada em busca de anomalias, que possam representar um comportamento de intrusão. Essa abordagem é apresentada na Figura 4.1 (as setas tracejadas entre os componentes representam dependências).

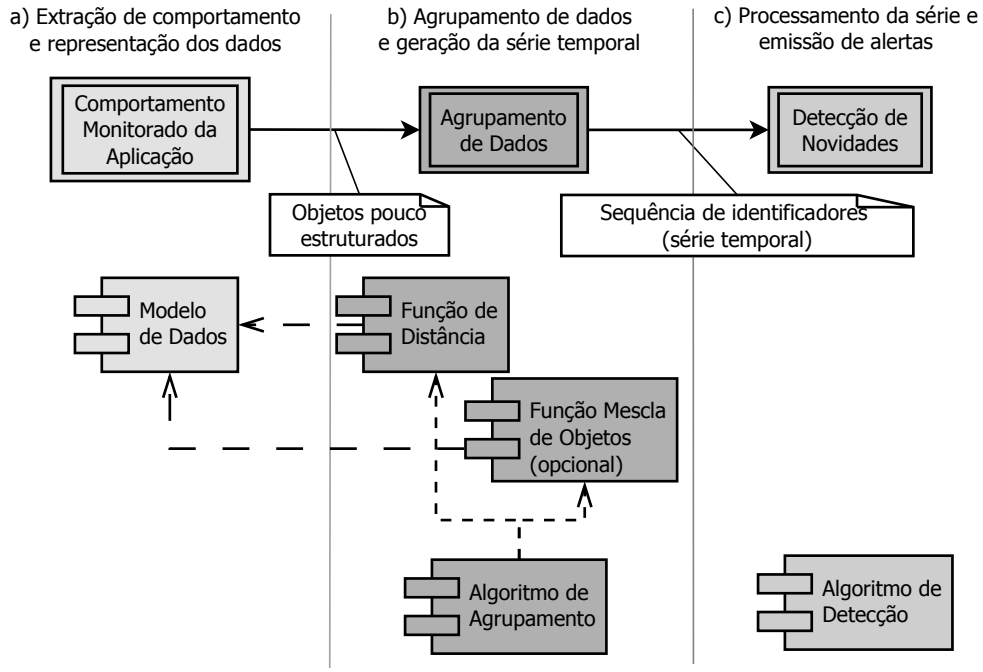


Figura 4.1: Proposta para caracterização de comportamento e detecção de novidades.

A abordagem proposta neste trabalho divide o problema em três etapas, conforme a Figura 4.1. Para aplicar essa abordagem, é necessário: a) extrair e representar um aspecto do funcionamento de um sistema (e.g. chamadas de sistema executadas por um processo, consumo de memória e processamento, mensagens de *log* ou tráfego de rede) por meio de um modelo de dados; b) definir uma função de distância entre objetos e, em alguns casos, uma função de mescla de diversos objetos como uma única observação, além de um algoritmo de agrupamento que recebe as saídas do sistema e gera uma série temporal que representa o comportamento da aplicação; c) aplicar um algo-

ritmo de detecção de novidades sobre a série gerada em busca de alterações no comportamento do sistema.

4.2 Metodologia

Para a avaliação da abordagem, são executados experimentos com base em sessões que representam o comportamento normal e os incidentes de ataque em diversas aplicações. Utilizam-se, neste trabalho, conjuntos de dados referentes a detecção de intrusão, a fim de obter uma avaliação mais precisa da proposta. Em seguida, são executados os experimentos, a fim de se observar a diferenciação entre situações normais e ataques, e por fim é feita a avaliação dos resultados.

A avaliação da eficácia desta dissertação é feita em três etapas: primeiramente, deve-se avaliar se o processo de agrupamento gera grupos válidos e pouco comuns (i.e. se técnicas de agrupamento são aplicadas de forma que grupos representativos sejam formados). Em segundo lugar, deve-se verificar se a série gerada pelo agrupamento caracteriza situações normais e permite distinguir comportamentos anômalos (ou de ataque). Por fim, verifica-se a existência correlação entre os índices de agrupamento e detecção, o que dá crédito à hipótese de que a informação adicionada à série auxilia tarefa de caracterização do sistema.

Para verificar o agrupamento serão utilizadas técnicas de validação interna e relativa. Como índice de validação interna é utilizado o Erro Quadrático Médio (Jain e Dubes, 1988) e como índice relativo a Silhueta de Rousseeuw (1987). É comparada também a eficácia das abordagens de agrupamento *online* tendo como base os resultados do algoritmo *K-Means*.

Após a etapa de agrupamento, as séries geradas são submetidas às técnicas de detecção de novidades, em busca de evidências da hipótese principal deste trabalho: de que a discretização seguida de detecção de novidades é capaz de caracterizar o comportamento de um sistema e detectar anomalias referentes a situações de ataque.

Conforme discutido anteriormente, é importante que a avaliação da abordagem seja aplicada não apenas a um conjunto de dados de um ambiente simulado, dado que essa simulação é limitada e pode não refletir o funcionamento real de um sistema, mas também a um conjunto de dados público que tenha sido utilizado em outros trabalhos. Os próximos capítulos apresentam os experimentos e os resultados dessas avaliações. Primeiro, a abordagem é avaliada com a aplicação ao conjunto de dados de caracterização de sistemas proposto por Twycross (2007), e em seguida é descrito o ambiente *Web* de simulação e os experimentos conduzidos nos dados extraídos desse ambiente.

CAPÍTULO

5

Avaliação da abordagem proposta

Este capítulo apresenta os resultados de uma avaliação inicial da técnica quando aplicada a um contexto semelhante ao do objetivo final deste trabalho. A técnica proposta foi avaliada por meio de experimentos executados sobre um conjunto de dados de caracterização de processos fornecido por Twycross e Aickelin (2010)[1]. Esse conjunto de dados apresenta diversas sessões de comportamento normal e de ataque para dois processos `UNIX` (`rpc.statd` e `WU-FTPD`). Entre os aspectos monitorados do sistema estão as chamadas de sistema executadas pelos processos e informações sobre desempenho do ambiente de execução.

Os experimentos executados sobre esse conjunto de dados utilizam as informações das chamadas de sistema, de forma que as instâncias e as sequências das chamadas representam o contexto no qual as técnicas de agrupamentos de dados e detecção de novidades são aplicadas para caracterizar o comportamento da aplicação. São utilizados o nome das chamadas e os valores dos parâmetros como entrada ao processo de agrupamento, que por sua vez gera uma série temporal na qual técnicas de detecção de novidades são aplicadas a fim de diferenciar o comportamento normal do anômalo.

Os objetivos desses experimentos foram, em primeiro lugar, avaliar a eficácia da técnica proposta na tarefa de diferenciar situações normais de situações anômalas, e em segundo lugar, avaliar a eficiência das técnicas *on-line* em comparação com uma técnica *batch*. Verificou-se que as técnicas *on-line* tem desempenho comparável a técnicas *batch* no conjunto de dados utilizados, e que a abordagem proposta é capaz de diferenciar as situações normais

[1] Conjunto de dados de comportamento de processos apresentado por Twycross (2007), descrito no Capítulo 4 e disponível em *http://www.cs.nott.ac.uk/ jpt/datasets.html*

das situações de ataque. Além disso, observou-se que a capacidade de diferenciação entre situações normais e de ataque é diretamente proporcional à qualidade de agrupamento, o que ocorre dado que as séries geradas por algoritmos de agrupamento com maior qualidade contém uma maior quantidade de informação, o que permite uma caracterização mais precisa do comportamento do sistema. Além disso, é feita uma breve análise do desempenho das técnicas em relação a tempo de execução dos procedimentos.

5.1 Descrição dos experimentos

Para a avaliação da abordagem, diversos algoritmos de agrupamento de dados e detecção de novidades são aplicados ao conjunto de dados fornecido por Twycross e Aickelin (2010). Esse conjunto de dados contém diversas informações sobre o ambiente de execução da aplicação. Todavia, os experimentos executados se limitam a utilizar o nome das chamadas, seus parâmetros e valores de retorno. Optou-se por não utilizar informações adicionais dado que o objetivo final deste trabalho é caracterizar o comportamento de uma aplicação a partir dos seus fluxos de dados, dando-se menos importância a informações sobre o ambiente.

Conforme descrito no Capítulo 3, o processo de agrupamento de dados é dividido em quatro etapas: definição da representação dos dados, definição da função de similaridade entre elementos, definição do algoritmo de agrupamento e aplicação do processo de validação. A seguir são detalhadas cada uma dessas etapas nos experimentos conduzidos.

5.1.1 Modelo de dados e função de distância

O conjunto de dados de chamadas de sistemas contém todas as chamadas executadas por duas aplicações `UNIX` (`rpc.statd`, um monitor do protocolo RPC, e `WU-FTPD`, um servidor FTP) extraídas utilizando-se o comando `strace`. São apresentadas sessões de comportamento normal, de ataques com sucesso e de ataques falhos. O conjunto de dados da aplicação `rpc.statd` é de menor tamanho (com cerca de 4.000 registros) e tem comportamento mais previsível que o da aplicação `WU-FTPD` (que conta com cerca de 80.000 registros). Os ataques com sucesso à aplicação `WU-FTPD` são divididos em três classes: A, onde o invasor explora a vulnerabilidade mas não toma nenhuma ação após o acesso ser concedido, B, onde ações comuns após ataques são executadas, e C, onde o invasor dispara um ataque a uma terceira máquina.

Os dados do conjunto são apresentados em arquivos com formato apresentado na Tabela 5.1. Esses arquivos contêm a saída do programa `strace` com as chamadas de sistema executadas pelo programa monitorado. É fornecido o

identificador de processo, o momento de execução, o valor do ponteiro contador de programa, o nome da chamada, seus parâmetros e o valor de retorno. São separados os arquivos com seções referentes a uso normal e a ataques.

Tabela 5.1: Trecho do conjunto de dados utilizado.

PID	*timestamp*	Ponteiro PC	Conteúdo da chamada
359	1137705530.312	[400bea8d]	close(0) = 0
359	1137705530.313	[400beb54]	lseek(0, 0, SEEK_SET) = 0
359	1137705530.314	[400c52dd]	fsync(0) = 0
359	1137705530.319	[400bea8d]	close(0) = 0
359	1137705530.322	[400aa257]	getpid() = 359

Esse arquivo é lido e interpretado, de forma que cada linha gera um objeto com todas as informações referentes a cada invocação de chamada de sistema. Cada objeto contém o tipo (i.e. nome) da chamada, uma lista com os valores dos parâmetros e, quando disponível, um valor de retorno. A fim de simplificar a representação dos dados, valores inteiros, endereços de rede IP e datas são convertidos em representações de ponto flutuante. Dessa forma, os parâmetros das chamadas podem ser do tipo numérico, nominal ou cadeia de caracteres. Uma etapa de pré-processamento é aplicada aos valores numéricos para normalizá-los, uniformemente, entre zero e um (i.e. $x_n = (x_i - x_{min})/(x_{max} - x_{min})$). Os parâmetros nominais e de cadeias de caracteres não são pré-processados.

Foi definida uma abordagem hierárquica para a aplicação do algoritmo de agrupamento e para a definição da função de similaridade. As chamadas são primeiramente agrupadas pelo nome, para em seguida serem submetidas ao processo de agrupamento, utilizando os valores dos parâmetros e do retorno da chamada. Assim sendo, chamadas com nomes distintos nunca são colocadas em um mesmo grupo, de forma que não é necessário definir uma função de distância que considere esse tipo de diferença. Sendo c o número de chamadas de sistema distintas que se observam no conjunto de dados, são aplicadas c instâncias distintas do algoritmo de agrupamento em questão. Dessa forma, garante-se que a pior desempenho de agrupamento (que ocorre quando todos os objetos são colocados em um mesmo grupo) gera uma série que representa apenas o nome da chamada de sistema, assim como ocorria nos trabalhos anteriores (Twycross, 2007; Forrest et al., 1996) que desconsideravam os valores dos parâmetros. Quando se observam melhores índices de agrupamento, as séries geradas contém uma maior quantidade de informação, dado que seus pontos não representam apenas o nome das chamadas, mas também os valores dos seus parâmetros.

Para obter a distância entre dois objetos do conjunto (i.e. duas chamadas de sistema do mesmo tipo), primeiro é necessário o cálculo da distância entre cada par de seus atributos (exemplos de atributos são fornecidos na Tabela

5.3). Para atributos do tipo numérico utiliza-se a diferença aritmética, cujo valor varia entre zero e um, dado que os valores foram previamente normalizados. Para valores nominais utiliza-se 1 se os atributos forem diferentes e 0 se forem iguais. Para cadeias de caracteres é utilizada a distância de Hamming (1950) dividida pelo comprimento da maior cadeia (caso as cadeias tenham comprimentos distintos, considera-se que as últimas posições da maior cadeia sempre divergem em seus valores). Independentemente do tipo, quando apenas um dos atributos tem valor nulo, considera-se que existe distância máxima entre eles (i.e. 1). Se ambos forem nulos, considera-se a distância mínima (i.e. 0). Tem-se, dessa forma, para cada par de chamadas de sistemas, um vetor de distâncias (variando sempre entre zero e um) com dimensão $p + 1$, onde p é o número de parâmetros que uma chamada de sistema aceita, conforme exemplo na Tabela 5.2.

Tabela 5.2: Exemplo de distância entre atributos.

Chamada	Atributo 1	Atributo 2	Atributo 3
`open`	`/etc/resolv.conf`	`O_RDONLY`	$0,0$
`open`	`/var/lib/nfs/state`	`O_RDWR`	$0,8$
Distância	$0,88$	$1,0$	$0,8$

Os valores desse vetor de distâncias devem, por fim, ser consolidados em um valor escalar. Para isso, primeiramente, consolidam-se os valores sob o mesmo tipo. Valores nominais utilizam a regra *simple matching* (Jain e Dubes, 1988), enquanto valores do tipo numérico e cadeia de caracteres utilizam distância Euclidiana normalizada (i.e. divide-se o valor da distância pelo número de atributos de um determinado tipo), conforme a Equação 5.1, onde a_n e b_n são os valores do atributo n das chamadas a e b, d_t é a dimensão de um determinado tipo (i.e. o número de atributos que possui determinado tipo) e $eq(a_n, b_n) = 1$ quando $a_n = b_n$ e $eq(a_n, b_n) = 0$ quando $a_n \neq b_n$. Assim, uma chamada pode ter no máximo três valores de distâncias consolidadas: uma para atributos numéricos, uma para atributos nominais e uma para atributos do tipo cadeia de caracteres.

$$D_{ct} = \begin{cases} \frac{\sum(eq(a_n,b_n))}{d_t} & \text{se tipo é nominal} \\ \\ \frac{\sqrt{\sum(a_n-b_n)^2}}{d_t} & \text{se tipo é numérico ou cadeia de caracteres} \end{cases} \tag{5.1}$$

Para a consolidação final dos valores é utilizada uma abordagem inspirada no trabalho de Gupta et al. (1999), onde o valor consolidado dos atributos nominais é multiplicado por um peso e somado ao valor consolidado dos atributos numéricos. Como neste trabalho tem-se atributos de três tipos diferentes, propõe-se uma adaptação a essa abordagem onde são definidos três pesos,

Tabela 5.3: Parâmetros e valor de retorno da chamada de sistema como atributos do algoritmo de agrupamento (com atributos do tipo cadeia de caracteres, nominal e numérico), após normalização.

Conteúdo da chamada	Atributos (após normalização)		
`open("/etc/resolv.conf", O_RDONLY)=0`	$/etc/resolv.conf$	O_RDONLY	$0,0$
`open("/etc/ld.so.cache", O_RDONLY)=0`	$/etc/ld.so.cache$	O_RDONLY	$0,0$
`open("/etc/localtime", O_RDWR)=2`	$/etc/localtime$	O_RDWR	$1,0$

um para cada tipo de atributo, sendo a distância consolidada final descrita na Equação 5.2. Considerou-se $w_{nom} = w_{num} = w_{cc} = 1.0$, isto é, o mesmo peso para todos os tipos.

$$D_c = \frac{w_{nom} \cdot D_{cnom} + w_{num} \cdot D_{cnum} + w_{cc} \cdot D_{ccc}}{w_{nom} + w_{num} + w_{cc}} \tag{5.2}$$

Além da função de distância, alguns algoritmos demandam ainda a definição de uma função de mescla de observações, a fim de consolidar diversos objetos em um único ponto médio. Nesses casos utilizou-se a função de mescla proposta por Gupta et al. (1999), que sugere o uso da média aritmética para agregar valores numéricos, e a moda estatística para agregar atributos nominais ou de cadeia de caracteres.

5.1.2 Etapa de agrupamento

Terminado o pré-processamento do conjunto de dados e definida a função de distâncias, deve-se definir o algoritmo de agrupamento a ser aplicado. Optou-se por avaliar os algoritmos *K-Means* (MacQueen, 1967), *Leader-Follower* Simples (Xu e Wunsch, 2008), *Leader-Follower Online-K-Means* (Xu e Wunsch, 2008), *Leader-Follower* Adaptativo (Charikar et al., 1997) e a rede neural GWR (Marsland et al., 2002). Com exceção do *K-Means*, todos esses algoritmos são *on-line* ou foram adaptados para serem aplicados de forma *on-line*. O algoritmo *K-Means* é utilizado como base para comparação de resultados por ser largamente utilizado, dado que ele comprovadamente converge a soluções ótimas locais após poucas iterações. Para aumentar as chances de se encontrar os máximos globais, são utilizadas 16 inicializações independentes. As iterações são executadas até que o valor do erro quadrático se estabilize em um mínimo local.

O algoritmo *Leader-Follower* foi aplicado em três variações: Simples, *Online-K-Means* e Adaptativo. O algoritmo simples utiliza um parâmetro θ constante, e os centróides não se movem no espaço conforme novos pontos são observados. A variação *Online-K-Means* é similar à simples dado que não adapta o parâmetro θ, mas utiliza a média de todos os pontos adicionados a um grupo como centróide. O algoritmo adaptativo é baseado no algoritmo guloso de

Charikar et al. (1997), inicializado com parâmetro θ de tamanho reduzido e com um parâmetro k que limita o número máximo de centróides. É executada uma operação para reduzir o número de centróides sempre que o seu número excede k. Nessa operação, os dois centróides mais próximos são consolidados em um único ponto. Nos experimentos conduzidos, o novo centróide não é um ponto médio entre os dois centróides, mas sim o centróide de maior diâmetro (considera-se como diâmetro de um centróide a maior distância entre o ponto central e os pontos aceitos por esse centróide). O valor do parâmetro θ desse novo ponto é a distância entre os dois centróides somada ao diâmetro do menor centróide.

Uma variação da rede neural GWR (Marsland et al., 2002) foi utilizada para permitir o uso de atributos não-numéricos e para ser aplicada de maneira *online*. A primeira diferença entre a rede original e a adaptação proposta é que na adaptação não existe uma etapa distinta de inicialização da rede. No trabalho original, essa inicialização seleciona aleatoriamente dois pontos distintos do conjunto como centróides iniciais, enquanto que na adaptação os dois primeiros pontos distintos observados são selecionados como os centróides iniciais. Outra distinção é que a rede original itera sobre a base de dados por t "períodos", enquanto na variação proposta é feita uma única varredura da base de dados. As funções de distância e de adaptação de centróides também são diferenciadas. A rede original é baseada na distância Euclidiana, enquanto que este trabalho utiliza a distância entre chamadas de sistema definida anteriormente. Para adaptar os centróides, o trabalho original movia os centróides em direção às observações utilizando uma média ponderada, de acordo com $\Delta w = \epsilon(\xi - w)$, onde w é o vetor que representa o centróide, ξ é a observação e ϵ é um parâmetro que define a velocidade pela qual os centróides se adaptam às novas observações. Devido à presença de atributos nominais, é utilizada a média ponderada (ou a moda estatística para atributos nominais) de todos os pontos adicionados a um grupo para definir um novo centróide. O trabalho original também não especificava se as operações de remoção de centróides deveriam ocorrer no final de cada período ou após o processamento de cada operação. Dado que não se aplica neste trabalho o conceito de período, a remoção de centróides pode ocorrer após o processamento de qualquer ponto do conjunto de dados.

5.1.3 Validação do Agrupamento

Dois mecanismos de validação da qualidade de agrupamento são utilizados: Erro Quadrático Médio (Jain e Dubes, 1988) e Silhueta (Rousseeuw, 1987). O erro quadrático médio considera apenas a dispersão intragrupo, enquanto a Silhueta leva em consideração a dispersão intragrupo e a separação entre

grupos. A Figura 5.1 apresenta os valores utilizados no cálculo dos índices para cada ponto do conjunto de dados: no caso, a_i representa a dispersão intragrupo, enquanto b_i representa a separação entre grupos.

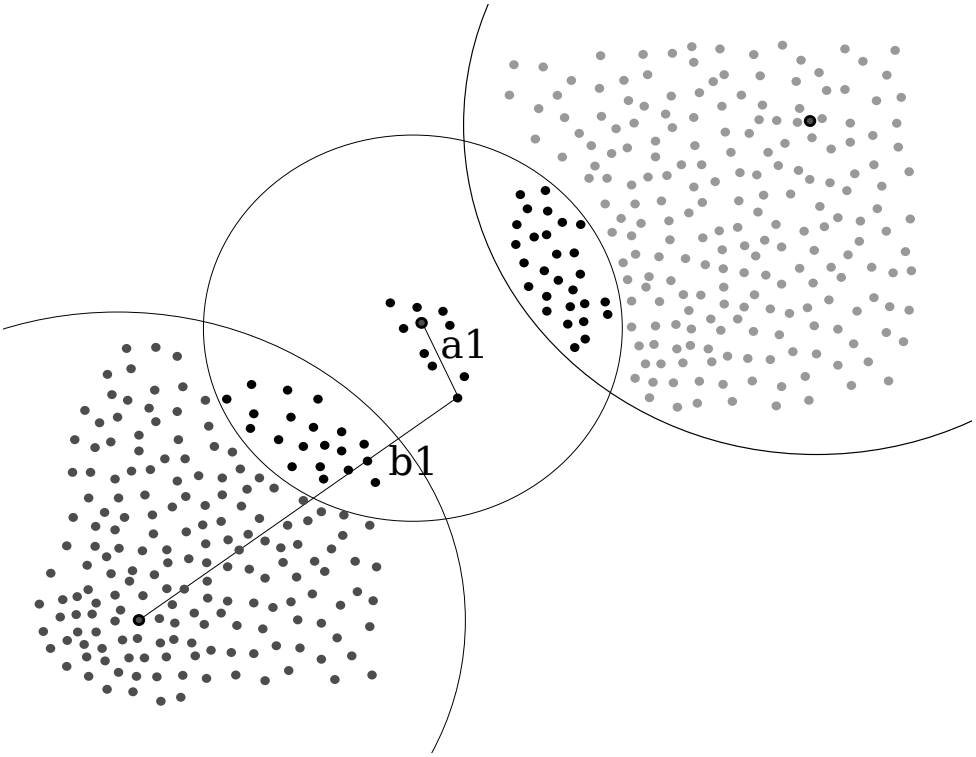


Figura 5.1: Distâncias utilizadas para cálculo da silhueta e do erro quadrático.

O valor do Erro Quadrático Médio é definido pela Equação 5.3. Esse índice calcula o valor médio da distância entre entre cada ponto e o ponto médio do grupo ao qual ele pertence. Valores baixos de Erro Quadrático Médio estão geralmente relacionados a maior qualidade de agrupamento, sendo exceção a essa regra os casos onde o número de grupos é superestimado (i.e. casos de $overfitting$).

$$EQM = \frac{\sum_{i=1}^{n}(a_i)^2}{n} \tag{5.3}$$

O valor da Silhueta é obtido a partir da Equação 5.4, onde $b(i)$ é a distância média entre um objeto e os objetos do segundo grupo mais próximo, e $a(i)$ é a distância média entre um objeto e os elementos do seu próprio grupo, e n é o total de pontos no conjunto de dados. O valor da Silhueta varia sempre entre -1 e 1, onde valores próximos a 1 tendem a representar uma melhor qualidade do agrupamento. Em ambas as equações n representa o número total de objetos no conjunto submetido a agrupamento.

$$SM = \frac{\sum_{i=1}^{n} \frac{b(i) - a(i)}{max(b(i), a(i))}}{n} \tag{5.4}$$

Implementou-se neste trabalho a Silhueta simplificada (Vendramin et al., 2009) por apresentar maior desempenho e qualidade comparável à Silhueta tradicional, uma vez que os conjuntos de dados são de grande porte.

5.1.4 Detecção de novidades

Para a caracterização do comportamento de processos foram avaliadas três técnicas: Janela Deslizante (Hofmeyr et al., 1998), Entropia de Shannon da Cadeia de Markov (Pereira e de Mello, 2009) e *Dynamic Time Warping* (DTW) (Pereira e de Mello, 2009). Essas técnicas recebem como entrada uma série temporal em uma etapa inicial de caracterização, para posteriormente avaliarem a quantidade de novidade observada em séries de teste. Neste trabalho as técnicas são submetidas a uma etapa de caracterização (i.e. de treino) utilizando as instâncias de funcionamento normal, sendo em seguida utilizadas para verificação de anormalidade tanto nas instâncias de funcionamento normal quanto nas instâncias de ataque. É esperado que as técnicas observem quantidades reduzidas de novidade nas séries de comportamento normal e valores altos nas séries de ataque.

A janela deslizante foi testada com uma janela de 10 posições, e não se observou impacto significativo na variação do valor desse parâmetro. Para distinguir as séries utiliza-se o valor da quantidade de novidade observada na etapa de validação. Quanto maior esse valor, maior é a distinção entre as séries, i.e. maior a distinção entre comportamentos. A janela deslizante considera como novidade a observação, na etapa de validação, de uma sequência que não tenha sido observada na etapa de caracterização. O valor na novidade em um determinado ponto da série é dado pela menor distância de Hamming entre a janela observada na validação e as janelas observadas na etapa de caracterização, dividido pelo tamanho da janela. O valor consolidado da novidade observada em uma série é dado pelo somatório da novidade em todos os pontos, dividido pelo número de observações na série.

Para estimar a Entropia de Shannon (Shannon, 1948) de uma sequência deve-se representar essa série como uma Cadeia de Markov, onde cada estado representa uma classe de observação da série, e as probabilidades de transições são estimadas a partir da sequência. A Entropia da cadeia dada pela Equação 5.5, onde C é o conjunto de todos os estados e $p(i, j)$ é a probabilidade de transição do estado i ao j.

$$H = -\sum_{i \epsilon C} \sum_{j \epsilon C} p(i,j) log_2(p(i,j)) \tag{5.5}$$

Quando diversas séries devem ser comparadas (e.g. quando existem diversas séries de comportamento normal e diversas séries de ataque, o que ocorre quando os processos observados ocorrem em paralelo) é utilizada a Entropia média de todos os processos como índice de novidade. O valor normalizado da diferença da Entropia, dado por $(H_{max} - H_{min})/H_{max}$, é utilizado como índice de diferenciação entre séries.

Algoritmo 3: DTW para múltiplas séries

```
1  minDistancias ← ∅ ;
2  para cada aSeq em sequenciasDeAtaque faça
3      distancias ← ∅ ;
4      para cada nSeq in sequenciasNormais faça
5          d ← dtw(aSeq, nSeq) ;
6          distancias.adiciona(d) ;
7      fim
8      distancia = min(distancias) ;
9      minDistancias.adiciona(distancia) ;
10 fim
11 msDTW ← media(minDistancias) ;
```

Por ter complexidade computacional $O(n^2)$, o algoritmo DTW é avaliado apenas para fins de comparação. DTW é implementado por meio de um algoritmo de programação dinâmica similar ao da distância de Levenshtein (Levenshtein, 1966), recebendo como entrada duas séries e uma função de distância entre observações da série. Como função de distância entre observações da série utiliza-se a distância entre centróides, resultante do processo de agrupamento (utilizando 1.0 como distância entre chamadas de tipo diferente). A saída desse procedimento representa a distância entre as duas séries. Neste trabalho é aplicado um processo de normalização a esse índice, que divide o valor da distância pelo comprimento da maior série. Quando existem diversas séries de comportamento normal e de ataque, o valor da distância DTW é calculado pelo Algoritmo 3, onde cada série de ataque é comparada a todas as séries normais, em busca da menor distância. A distância final desse grupo de séries é a média de todas as distâncias de ataque.

A Figura 5.2 apresenta a instanciação da abordagem para a condução dos experimentos. Conforme esse diagrama, a representação como objetos dos conjuntos de dados de chamadas de sistema é utilizado como modelo de dados. A distância entre chamadas de mesmo tipo e a mescla entre objetos inspirada no trabalho de Gupta et al. (1999) são utilizadas como função de

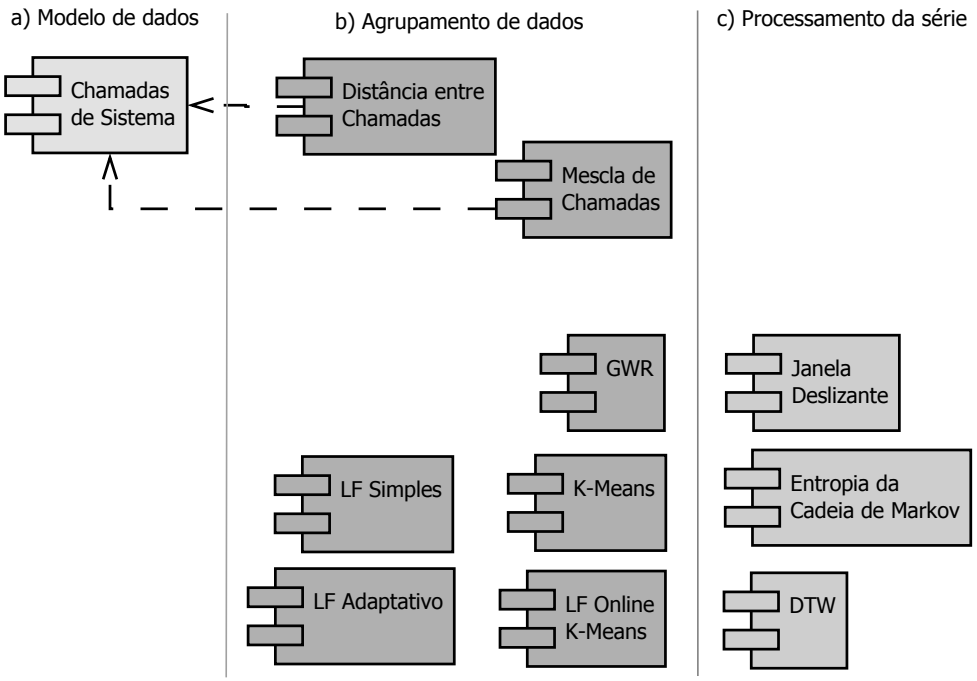


Figura 5.2: Aplicação da abordagem para fins de validação.

distância e mescla entre objetos, e são avaliados os 5 algoritmos de agrupamento e os 3 de detecção de novidades (sendo os algoritmos *K-Means* e DTW utilizados apenas para fins de comparação).

5.1.5 Conjuntos de dados de detecção de intrusão

A construção de conjuntos de dados para detecção de intrusão não é uma tarefa trivial: existem basicamente duas abordagens, ambas com grandes limitações. A primeira realiza a coleta de dados de sistemas reais, e a segunda, a simulação do comportamento de um sistema em ambiente controlado.

A coleta de dados de sistemas reais permite uma caracterização bastante precisa de um sistema, uma vez que as informações coletadas refletem justamente o comportamento e os fluxos de dados observados em ambiente de produção. Por outro lado, essa abordagem apresenta duas grandes limitações. A primeira é que os dados do sistema real podem conter informações pessoais ou confidenciais. Uma forma de se tratar essa questão é uma anonimização cuidadosa dos dados (por exemplo com a troca de nomes e números de documentos por dados fictícios). Essa geralmente é uma tarefa manual, bastante custosa e sujeita a erros. Além disso, existe a possibilidade dos dados anonimizados terem relevância para a detecção de intrusão, ou seja, existe o risco de perda de informação, e grande parte das ferramentas atua apenas em nível de rede (Pang et al., 2006), descartando grande parte das informações. A segunda limitação é que os dados de um sistema real não são categorizados, de forma que não se sabe, sem uma análise manual e cuidadosa, o que

de fato representa um ataque e o que representa comportamento normal da aplicação.

A simulação do comportamento de um sistema, por outro lado, é uma tarefa mais simples e precisa do ponto de vista de projeto de experimento. Nessa abordagem coletam-se apenas informações estatísticas de um sistema real, a fim de que se reproduzam cenários observados em um ambiente simulado. Paralelamente à simulação, disparam-se ataques em alguns momentos específicos. Dessa forma, não é necessário ocultar informações confidenciais e os eventos são categorizados. A grande limitação dessa abordagem é que existe risco de que o comportamento simulado não reflita o comportamento real de um sistema.

A seguir serão apresentados alguns conjuntos de dados utilizados em outros trabalhos, além de uma abordagem para a construção de um conjunto de dados de ataques a aplicações *Web*.

Conjunto de dados do "DARPA Intrusion Detection Evaluation"

Em 1998 e 1999 o *Information Systems Technology Group (IST)* do *MIT Lincoln Laboratory* gerou um conjunto de dados de intrusões em rede a pedido do *Defense Advanced Research Projects Agency (DARPA ITO)* e do *Air Force Research Laboratory (AFRL/SNHS)*, sendo esse conjunto comumente denominado DARPA IDEval (DARPA, 1999). Para a construção desse conjunto, o tráfego de rede em uma base da força aérea norte-americana foi mensurado e caracterizado, para em seguida ser simulado em uma rede isolada. Diversos ataques foram reproduzidos durante essa simulação (Cunningham et al., 1999a), paralelamente ao tráfego normal.

No trabalho de 1998 foram coletados dados durante nove semanas. Fazem parte da rede simulada apenas sistemas `UNIX` (Linux e Solaris), e entre os protocolos simulados estão SNMP, SMTP, telnet, FTP, HTTP e POP. Para reproduzir o comportamento foram utilizados *scripts* que simulam operações de usuários, como trocas de *e-mails*, acessos *Web* e sessões `telnet` (Lippmann et al., 2000). Em paralelo ao tráfego simulado, foram executadas mais de 300 instâncias de 38 tipos diferentes de ataques. Foram extraídas informações sobre todas as mensagens de rede (coletadas com a ferramenta TCPDUMP (2009)), saídas do comando `ps` do `UNIX` executado a cada minuto, além de dados de auditoria do *Solaris Basic Security Module* (BSM). Esse conjunto também fornece uma listagem com todos os ataques executados no sistema, com horário de execução e conexões de rede associadas.

Esse trabalho foi expandido em 1999 com a inclusão de máquinas com sistema operacional *Windows* (Cunningham et al., 1999b). Foram executadas nessa extensão 200 instâncias de 58 tipos diferentes de ataques. Além das

informações coletadas no ano anterior, esse trabalho capturou *logs* de auditoria da plataforma *Windows* e listagens diárias dos sistemas de arquivos em algumas máquinas.

Por capturar todo o tráfego de rede, esse conjunto de dados é de grande valia para a construção de ferramentas de detecção de intrusão baseadas nesse contexto (Cunningham et al., 1999a,b), tendo sido utilizado em diversos trabalhos. Por exemplo, Dasgupta e Gonzalez (2002) utilizam esse conjunto para construir um sistema imunológico artificial que busca intrusões a partir do tráfego da rede, onde são analisadas informações estatísticas (como número de *bytes* trafegados por segundo, número de pacotes por segundo e número de pacotes ICMP por segundo). É implementada uma solução de aprendizado supervisionado, onde se treina o sistema utilizando um intervalo sem ataques e verifica-se a capacidade de diferenciação das situações de ataques. Observa-se que os perfis são bastante distintos, e que a diferenciação de comportamentos é possível.

Esse conjunto de dados é um dos mais sofisticados e mais utilizados atualmente, sendo o único trabalho de grande escala que teve como objetivo gerar um conjunto que caracterizasse todo o comportamento de uma rede real (McHugh, 2000). Apesar disso, esse trabalho apresenta diversas falhas e limitações (Brugger, 2007). Uma análise superficial verifica diversas irregularidades nos padrões das mensagens. Mahoney e Chan (2003) observam, por exemplo, que existem apenas 29 endereços TCP remotos em conexões de duas semanas, sendo metade desses endereços observada no primeiro $0,1\%$ do tráfego. Apenas 9 valores distintos do atributo *time-to-live* são observados, alguns desses apenas em situações de ataque. Existem também indícios de ataques não documentados nas listagens.

Outra limitação observada é a granulosidade das informações coletadas. Com exceção das informações de rede, onde todo o tráfego observado é armazenado, as informações são coletadas de forma esporádica ou arbitrária. Informações sobre arquivos nos discos das máquinas são coletadas uma única vez por dia, e as informações sobre os processos em execução (i.e. as saídas do comando `ps`) são coletadas uma vez a cada minuto, sendo insuficientes para uma caracterização precisa do comportamento do sistema. O módulo BSM gera uma grande quantidade de informações para auditoria, mas muitos registros estão incompletos: experimentos conduzidos neste trabalho mostram, por exemplo, que nos registros de quarta-feira da terceira semana, verifica-se apenas $6,6\%$ dos registros das chamadas de sistema com ID de processo associado, e que desses ID's apenas $13,3\%$ têm um nome de processo associado (o que ocorre dada a granulosidade das execuções do comando `ps`). Não é possível então caracterizar o comportamento de processos específicos

de forma precisa, o que impede o uso de conjunto de dados para a avaliação da abordagem principal deste trabalho. Por outro lado, o grande tamanho do conjunto de dados é um fator interessante para a avaliação da escalabilidade da solução, desde que a precisão da mesma seja avaliada por meio de outros experimentos.

Conjuntos de dados de caracterização de processos

Diversos trabalhos em detecção de intrusão constroem conjuntos de dados para caracterização de comportamento de processos. A limitação de grande parte dos trabalhos é que os conjuntos gerados contêm apenas informações utilizadas pelo sistema proposto. Hofmeyr et al. (1998), por exemplo, geram conjuntos com sequências de chamadas de sistema em situações normais e de ataque, e os publicam apenas com o nome de chamadas, sem seus parâmetros. Essa limitação não se observa na tese apresentada por Twycross (2007), onde são gerados conjuntos a partir das aplicações `rpc.statd` (monitor do protocolo RPC) e `WU-FTPD` (servidor FTP) em situações normais e de ataque. Apesar de não serem utilizados pela tese, os parâmetros das chamadas estão presentes no conjunto de dados.

Para o *software* `rpc.statd` são gerados seis conjuntos de dados, contendo horário de execução, nome e parâmetros das chamadas, além do valor de retorno. Dois desses conjuntos caracterizam o funcionamento normal do sistema (com cerca de 450 registros cada), dois caracterizam o funcionamento em situação de ataque sem sucesso (com cerca de 500 registros cada) e dois caracterizam o funcionamento sob ataque com sucesso (com cerca de 1700 registros cada). Para a aplicação `WU-FTPD` é executado um número muito maior de casos de testes. O conjunto de dados de uso normal contém cerca de 40.000 chamadas de sistemas, geradas a partir de 55 sessões FTP. Para os ataques são executados 4 casos de sucesso e um de falha, onde cada um dos conjuntos conta com cerca de 7.000 registros.

Um sistema que caracterize o comportamento de aplicações deve ser capaz de identificar similaridades entre conjuntos sem ataques, e distinguir conjuntos com ataques. Partindo da hipótese de que uma maior qualidade de agrupamento de dados adiciona maior informação à caracterização do sistema, planeja-se utilizar esse conjunto para avaliar, em primeiro lugar, a eficiência das técnicas de agrupamento, e em segundo lugar, utilizar a série resultante do agrupamento para caracterizar o sistema em estudo. É mostrado nos capítulos a seguir que o agrupamento de dados, ao considerar uma quantidade maior de informações, fornece uma caracterização mais precisa, permitindo uma maior diferenciação entre situações normais e de ataque, que a abordagem que considera apenas o nome das chamadas.

Conjuntos de dados de sistemas sintéticos

Conjuntos de dados de detecção de intrusão são escassos na literatura, e ainda mais escassos são aqueles referentes a aplicações *Web*. Grande parte dos conjuntos de dados de sistemas *Web* contém apenas *logs* de acesso (como os disponibilizados pelo *Internet Traffic Archive* (LBNL, 1998), por exemplo), sendo a maioria coletada a partir de sistemas em produção, ou seja, sem categorização. No contexto deste trabalho, as maiores limitações desses conjuntos são, em primeiro lugar, o fato de que eles não apresentam o comportamento de nenhum componente de software, mas apenas os registros de acessos. Em segundo lugar, a ausência de categorização dificulta a tarefa de validação dos resultados. Essas limitações motivam a proposta de um ambiente simulado para a avaliação dos resultados finais deste trabalho.

Grande parte das aplicações *Web* atualmente disponíveis é baseada no modelo de três camadas, com uma camada de apresentação, representada pelo navegador do usuário, uma camada lógica, onde a aplicação de fato é codificada, e uma camada de dados, onde informações de negócio são armazenadas (implementado por exemplo em um SGBD). Em relação a infraestrutura de rede, é comum o uso de *firewalls* e *proxies* reversos para que os clientes não acessem diretamente uma determinada aplicação. Para a caracterização do comportamento desse tipo de aplicação pode-se monitorar os dados enviados ao servidor HTTP pelos clientes, extraídos no proxy reverso, além dos comandos executados no SGBD (Figura 5.3) que podem ser obtidos a partir de relatórios da própria ferramenta.

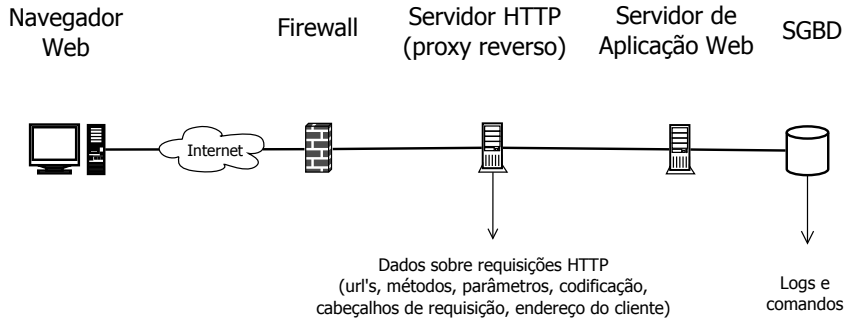


Figura 5.3: Pontos de coleta de dados.

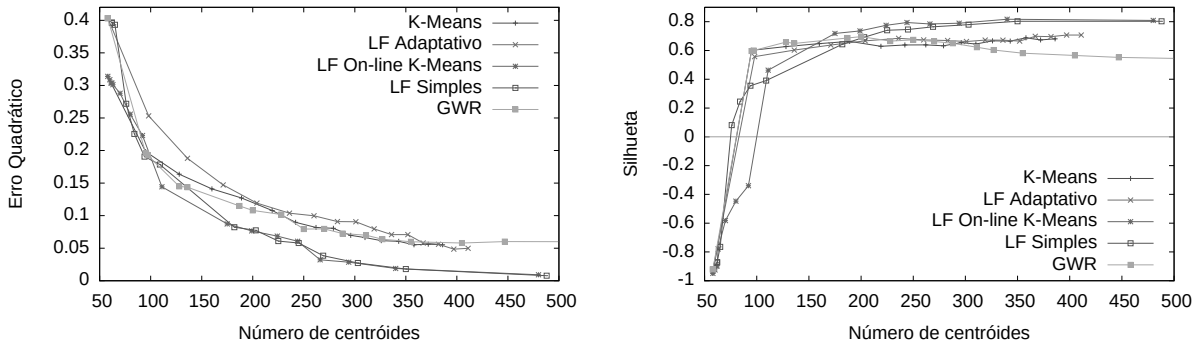
Para a simulação desse tipo de ambiente são necessários, portanto, um simulador de acessos HTTP e uma aplicação *Web*. Para a execução dos testes, pode-se utilizar uma aplicação com algumas vulnerabilidades, e programar o simulador de acessos para a execução de seções de acesso normal e de ataques.

5.2 Resultados

Nesta seção são apresentados os resultados da aplicação da técnica proposta ao conjunto de dados de chamadas de sistema. Inicialmente são apresentados os resultados da etapa de agrupamento, e em seguida são discutidos os resultados da etapa de detecção de novidades. É então analisada a correlação entre os índices de qualidade de agrupamento e a capacidade de diferenciação entre situações normais e anomalias, considerando a hipótese de que as séries com maior quantidade de informação, geradas a partir de processos de agrupamento mais precisos, permitem uma melhor diferenciação entre situações normais e ataques. Por fim, são apresentadas e analisadas curvas ROC estimadas a partir dos índices de novidade.

5.2.1 Agrupamento de dados

A Figura 5.4 apresenta os índices de Erro Quadrático Médio e Silhueta em relação ao número de centróides gerados para as chamadas de sistema do processo `rpc.statd`. Conforme descrito anteriormente, esse conjunto de dados contém um menor número de objetos e apresenta comportamento mais previsível que o do `WU-FTPD`, sendo útil para as análises iniciais das técnicas a serem empregadas (Twycross, 2007). Esse conjunto é composto por 5.379 chamadas, das quais 884 compõe duas sessões de funcionamento normal, 3.482 compõe duas sessões de ataques com sucesso e 1.013 chamadas compõe duas sessões de ataques falhos.



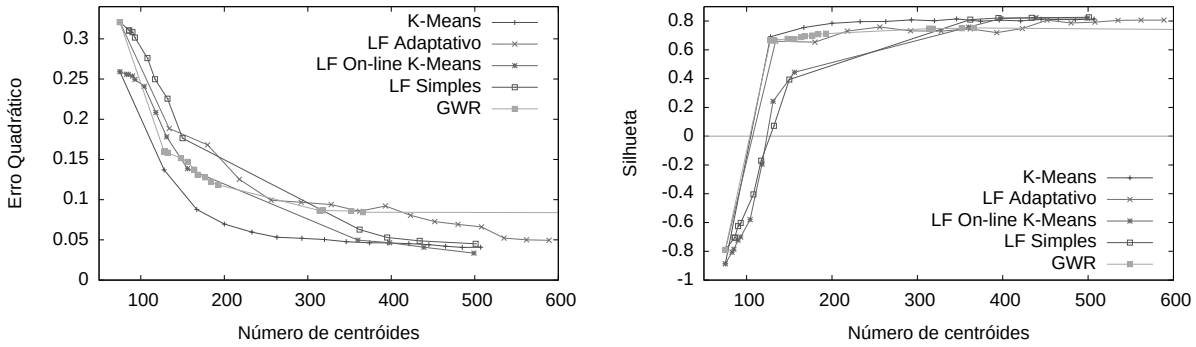
(a) Erro quadrático médio (b) Silhueta

Figura 5.4: Índices de agrupamento para o processo `rpc.statd`.

Ao analisar os índices de silhueta e erro quadrático médio observa-se que os algoritmos *K-Means*, *Leader-Follower* Adaptativo e a rede neural GWR apresentam melhores resultados apenas quando o número de centróides é baixo. Nota-se que todos os algoritmos *on-line* apresentam precisão semelhante ao *K-Means*, principalmente quando o número de centróides é alto. Observa-se

ainda uma piora no desempenho da rede GWR quando o número de centróides é muito alto.

A Figura 5.5 apresenta os índices de agrupamento para o processo `WU-FTPD`. Esse conjunto de dados é mais complexo que o `rpc.statd`, e representa com maior precisão um cenário de sistema real dado que foi gerado a partir de um conjunto de dados público de sessões FTP. Esse conjunto é composto por 83.271 objetos, dos quais 44.103 compõe 55 sessões de funcionamento normal, 32.737 objetos representam quatro sessões de ataques com sucesso e 6.431 chamadas representam uma sessão de ataque falho.



(a) Erro quadrático médio (b) Silhueta

Figura 5.5: Índices de agrupamento para o processo `WU-FTPD`.

Assim como no conjunto `rpc.statd`, observa-se precisão semelhante entre todos os algoritmos de agrupamento. Observam-se melhores índices de agrupamento pelo algoritmo *K-Means* quando o número de centróides é reduzido, desempenho similar entre todos os algoritmos conforme o número de centróides é incrementado, e menores índices de qualidade pela rede GWR quando o número de centróides é muito alto.

Considerando os índices de qualidade de agrupamento e a facilidade de uso dos algoritmos, observa-se que o algoritmo *Leader-Follower* Adaptativo apresenta boa precisão, tendo parâmetros cuja estimativa é mais simples que o θ do *Leader-Follower* Simples e do *Leader-Follower Online-K-Means*. Enquanto esses algoritmos demandam a definição de um θ com razoável precisão, o *Leader-Follower* Adaptativo demanda um parâmetro θ subestimado e um k superestimado, o que evita os problemas de escalabilidade dos outros algoritmos *on-line*. A rede GWR, apesar de ter caráter adaptativo, demanda a definição de um parâmetro "*insertion threshold*" similar ao θ dos outros algoritmos, cuja estimativa é inviável sem uma análise exploratória do conjunto de dados.

5.2.2 Detecção de novidades

Gerada a série temporal a partir da etapa de agrupamento de dados, é aplicada a etapa de detecção de novidades a fim de diferenciar o comportamento normal das situações de ataque. As 55 sessões de comportamento normais são agrupadas em 5 blocos de 11 sessões cada, a fim de aumentar a precisão dos detectores e facilitar a interpretação dos dados. Essa consolidação não é necessária para o processo `rpc.statd` dado que existem apenas duas sessões normais.

As Tabelas 5.4, 5.5, 5.6 e 5.7 são exemplos das saídas do processo de detecção de novidades. São efetuados treinos (i.e. caracterizações) com as situações normais, e testes com as situações normais e ataques. As tabelas apresentam os resultados da aplicação da distância DTW sobre as séries geradas a partir de agrupamento pelo algoritmo *K-Means* com $k = 1$ (i.e. com índice de qualidade mínimo, onde a série contém apenas informações referentes ao tipo da chamada de sistema) e com $k = 16$ (i.e. quando a série contém informações não apenas sobre o tipo da chamada, mas também sobre os valores dos parâmetros).

Tabela 5.4: `rpc.statd`: DTW / *K-Means*($k = 1$).

	normal 1	normal 2
normal 1	$0,00\%$	$1,52\%$
normal 2	$1,52\%$	$0,00\%$
-	min:$0,0$, med: $0,7$, max: $1,5$	
ataque 1	$36,14\%$	$36,08\%$
ataque 2	$36,17\%$	$36,14\%$
-	min:$36,1$, med: $36,1$, max: $36,2$	

Tabela 5.5: `rpc.statd`: DTW / *K-Means*($k = 16$).

	normal 1	normal 2
normal 1	$0,00\%$	$1,71\%$
normal 2	$1,71\%$	$0,00\%$
-	min:$0,0$, med: $0,9$, max: $1,7$	
ataque 1	$53,06\%$	$52,67\%$
ataque 2	$53,14\%$	$52,72\%$
-	min:$52,7$, med: $52,9$, max: $53,1$	

A partir dessas tabelas pode-se observar que o processo `rpc.statd` é, de fato, muito mais previsível que o `WU-FTPD`, o que pode ser observado pelo maior valor na diferenciação entre situações normais e de ataque, e pela menor variação na quantidade de novidade nas situações normais. Pode-se observar também no comportamento do processo `WU-FTPD` que as três classes de ataque apresentam quantidades distintas de novidade: na sessão 1 (classe A) observa-se a menor quantidade de novidade em relação às sessões de ataque, dado que

a vulnerabilidade foi explorada (i.e. o invasor conseguiu acesso à máquina) mas nenhuma ação nociva foi executada. Nas sessões 2 e 3 (classe B) observa-se uma maior quantidade de novidade, dado que os invasores efetuaram ações comuns em ataques. A maior quantidade de novidade é observada na sessão 4 (classe C), onde um novo ataque é disparado, o que deixa a maior quantidade de rastros de ataque.

Comparando-se os resultados da Tabela 5.4 com a Tabela 5.5 observa-se um incremento na capacidade de diferenciação entre situações normais e ataques quando a técnica é aplicada. Quando $k = 1$ os pontos da série temporal representam apenas o tipo da chamada de sistema, enquanto que com $k = 16$ informações sobre os parâmetros das chamadas são agregadas às séries (uma vez que uma mesma chamada pode ter até 16 diferentes representações na série). Observa-se um pequeno aumento na quantidade de novidade das situações normais, mas um aumento muito maior na quantidade de novidade nas situações de ataque, o que mostra que a informação agregada é útil à diferenciação entre situações normais e ataques.

Tabela 5.6: `WU-FTPD`: DTW / *K-Means*($k = 1$).

	bloco 1	bloco 2	bloco 3	bloco 4	bloco 5
bloco 1	$0,00\%$	$18,21\%$	$12,39\%$	$12,76\%$	$9,58\%$
bloco 2	$8,68\%$	$0,00\%$	$9,89\%$	$4,30\%$	$5,90\%$
bloco 3	$20,27\%$	$30,65\%$	$0,00\%$	$18,04\%$	$19,45\%$
bloco 4	$9,35\%$	$19,36\%$	$11,34\%$	$0,00\%$	$8,16\%$
bloco 5	$8,68\%$	$16,92\%$	$13,36\%$	$9,50\%$	$0,00\%$
-	min:$0,0$, med: $10,7$, max: $30,6$				
ataque 1	$34,25\%$	$34,14\%$	$34,59\%$	$34,99\%$	$34,99\%$
ataque 2	$42,26\%$	$42,18\%$	$42,39\%$	$43,01\%$	$43,05\%$
ataque 3	$43,52\%$	$44,28\%$	$43,74\%$	$44,30\%$	$43,85\%$
ataque 4	$77,65\%$	$78,07\%$	$74,43\%$	$77,40\%$	$77,55\%$
-	min:$34,1$, med: $49,5$, max: $78,0$				

Tabela 5.7: `WU-FTPD`: DTW / *K-Means*($k = 16$).

	bloco 1	bloco 2	bloco 3	bloco 4	bloco 5
bloco 1	$0,00\%$	$19,37\%$	$14,01\%$	$13,83\%$	$10,41\%$
bloco 2	$9,51\%$	$0,00\%$	$10,86\%$	$4,87\%$	$6,60\%$
bloco 3	$21,43\%$	$32,30\%$	$0,00\%$	$19,16\%$	$20,24\%$
bloco 4	$10,33\%$	$20,92\%$	$13,00\%$	$0,00\%$	$8,92\%$
bloco 5	$9,74\%$	$18,47\%$	$14,64\%$	$10,66\%$	$0,00\%$
-	min:$0,0$, med: $11,6$, max: $32,3$				
ataque 1	$38,32\%$	$37,97\%$	$38,71\%$	$39,10\%$	$39,24\%$
ataque 2	$46,26\%$	$46,00\%$	$46,20\%$	$47,00\%$	$47,11\%$
ataque 3	$47,51\%$	$48,41\%$	$47,75\%$	$48,31\%$	$48,01\%$
ataque 4	$80,55\%$	$80,53\%$	$78,98\%$	$80,49\%$	$80,65\%$
-	min:$38,0$, med: $53,4$, max: $80,6$				

As Tabelas 5.6 e 5.7 apresentam os resultados da caracterização para o

processo `WU-FTPD`. O incremento na diferenciação entre situações normais e ataques é menos expressivo que o observado para o processo `rpc.statd` (dado que o conjunto de dados é maior e o comportamento é menos previsível), mas ainda assim a diferença é estatisticamente significativa, o que mostra que a informação adicional é útil à caracterização do sistema.

Para consolidar esse resultado em um único índice, que representa a separabilidade entre as situações normais e as de ataque, são utilizados dois índices: a área sob a curva ROC, apresentada posteriormente, e a diferença entre as médias dos dois conjuntos de resultados. Quanto maior a separabilidade, melhor é a caracterização do sistema, uma vez que se consegue uma maior distinção entre os diferentes tipos de situações. No restante deste trabalho, esse valor é chamado de "diferença de médias". Como exemplo, a diferença de médias entre os resultados da aplicação do *K-Means* com $k = 1$ e DTW é $(36,1 - 0,7) = 35,4$ para o processo `rpc.statd` e $(49.5 - 10.7) = 38.8$ para o processo `WU-FTPD`.

É analisado, para cada algoritmo de agrupamento, o incremento na diferença de médias em duas situações: quando se observa o pior índice de erro quadrático (o que ocorre quando todas as observações são agrupadas em um mesmo conjunto, i.e. quando $k = 1$ ou $\theta = 1.0$) e quando se observa o melhor índice de erro quadrático (que se observa quando o número de centróides é incrementado). Por exemplo, o incremento na diferença de médias do algoritmo *K-Means* com DTW quando k varia de 1 a 16 é $(52.9 - 0.9) - (36.1 - 0.7) = 16.6$.

A Tabela 5.8 apresenta o incremento na diferença de médias para o processo `rpc.statd`, onde se observa que todos os algoritmos de agrupamento e detecção de novidades apresentam aumento significativo na capacidade de detecção quando a técnica de agrupamento é aplicada. Todos esses valores passam por um teste de significância estatística por meio de *resampling* (Shasha e Wilson, 2008) com intervalo de confiança de 95%.

Tabela 5.8: `rpc.statd`: Incremento na diferença de médias.

	LF Adaptativo	GWR	*LF* Simples	*K-Means*	*Online KMLF*
Entropia	18.44	19.67	31.53	18.18	31.51
Janela Deslizante	15.53	20.73	16.69	19.83	15.75
DTW	16.95	16.69	16.94	16.67	16.92

Para o cálculo do intervalo de confiança são executados 10.000 experimentos onde, a partir dos conjuntos com os resultados da caracterização do comportamento normal e de ataque (e.g. das Tabelas 5.4, 5.5, 5.6 e 5.7), geram-se dois conjuntos pelo meio de sorteio de elementos aleatórios de cada um dos conjuntos. Calcula-se a diferença da média de cada um desses conjuntos, e monta-se uma lista com todos esses valores. Essa lista é ordenada, e, dessa forma, os elementos de número 500 e 9500 dessa lista apresentam os valores

do intervalo de confiança de 90%.

Para o cálculo da significância estatística são também executados 10.000 experimentos onde são gerados conjuntos de elementos aleatórios. A diferença é que, nesse caso, os elementos de cada um dos conjuntos podem ser selecionados a partir de qualquer uma das listas (com probabilidade de 50% de se selecionar um elemento de cada origem). É calculada a diferença entre as médias dos conjuntos gerados, e dessa forma o valor da significância é dado pelo percentual de experimentos onde o valor da média for maior ou igual ao valor da média observada nos dados originais.

As Tabelas 5.9, 5.10 e 5.11 apresentam o incremento na diferença de médias para o processo `WU-FTPD` para as três classes de ataque (A, B e C). A Tabela 5.9 apresenta os incrementos da situação normal quando comparada ao ataque de classe A, onde nenhuma ação nociva foi tomada. A Tabela 5.10 apresenta os resultados para os ataques de classe B, onde ações comuns em ataques foram executadas após o acesso ser concedido, enquanto que a Tabela 5.11 apresenta os incrementos para a classe C, onde um novo ataque foi disparado a partir da máquina comprometida. DTW foi a única técnica que não apresentou um incremento significativo nos índices de detecção após as técnicas de agrupamento serem aplicadas. Todas as outras diferenças de médias passam um teste de significância estatística por resampling com intervalo de confiança de 90%.

Tabela 5.9: `WU-FTPD-A`: Incremento na diferença de médias.

	LF Adaptativo	GWR	*LF* Simples	*K-Means*	*Online KMLF*
Entropia	12.30	6.98	14.59	17.34	15.35
Janela Deslizante	7.14	9.37	9.05	9.89	8.70
DTW	2.84	3.31	3.16	3.23	3.12

Tabela 5.10: `WU-FTPD-B`: Incremento na diferença de médias.

	LF Adaptativo	GWR	*LF* Simples	*K-Means*	*Online KMLF*
Entropia	10.55	5.50	11.84	14.29	12.34
Janela Deslizante	7.81	9.77	9.21	10.60	9.04
DTW	2.95	3.28	3.07	3.15	3.25

Tabela 5.11: `WU-FTPD-C`: Incremento na diferença de médias.

	LF Adaptativo	GWR	*LF* Simples	*K-Means*	*Online KMLF*
Entropia	-13.28	-34.20	-9.31	-12.68	-9.65
Janela Deslizante	13.70	13.75	14.04	11.46	14.23
DTW	1.19	2.10	1.79	2.34	1.98

A partir das Tabelas 5.9 e 5.10 pode-se observar um aumento considerável nos índices de detecção das técnicas de Janela Deslizante e Entropia da Cadeia de Markov, o que mostra que a informação dos parâmetros adicionada

à série foi útil para a caracterização do comportamento e para a diferenciação entre situações normais e de ataque. Em contrapartida, a Tabela 5.11 mostra que a informação adicionada auxiliou apenas a técnica da Janela Deslizante, e que interferiu nos resultados da técnica baseada em Entropia.

5.2.3 Desempenho de execução

Em relação ao tempo de execução, todos os algoritmos *on-line* de agrupamento apresentam desempenho similar. Os testes foram executados em uma máquina com 2 *cores* de 2.8GHz, levando em média 6 segundos para processar o conjunto de dados `rpc.statd` e 4 minutos para processar o conjunto de dados `WU-FTPD`. O algoritmo *K-Means* levou 5 minutos para processar o conjunto `rpc.statd` e 4 horas para o conjunto `WU-FTPD`.

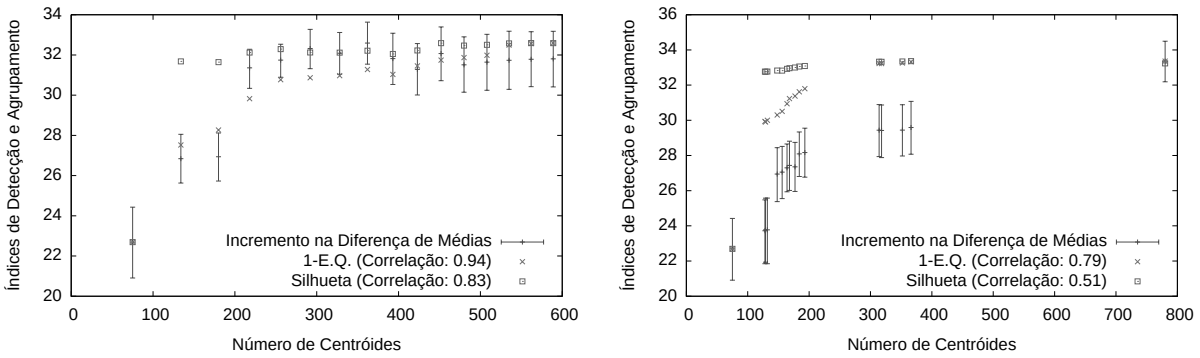
O desempenho dos algoritmos de detecção de novidades, por outro lado, varia consideravelmente entre as técnicas. A Entropia é a técnica que apresenta melhor desempenho, uma vez que depende apenas da transição entre estados na sequência, apresentando complexidade próxima de linear. Já o desempenho da Janela Deslizante depende não apenas do tamanho da janela, mas principalmente da quantidade de novidade observada em uma série. A DTW tem tempo de processamento quadrático, e por isso apresenta pior desempenho.

O tempo de processamento (i.e. o tempo necessário para a construção de uma tabela de detecção, como as Tabelas 5.4 ou 5.6) da Entropia foi de menos de um segundo para o conjunto `rpc.statd` e cerca de dois segundos para o `WU-FTPD`. A Janela Deslizante leva cerda de 3 segundos para processar o conjunto `rpc.statd` e 9 minutos para processar o `WU-FTPD`. Esse tempo, porém, é diretamente proporcional à quantidade de informação contida na série: o processamento de uma série gerada pelo *K-Means* a partir do conjunto `WU-FTPD` com $k = 1$ (i.e. quando a série contém informações apenas sobre o nome das chamadas) leva cerca de 3 minutos, enquanto que o processamento da série gerada com $k = 16$ leva cerca de 12 minutos. A DTW leva em média 6 segundos para processar o conjunto `rpc.statd` e 2 horas para o conjunto `WU-FTPD`.

5.2.4 Correlação entre agrupamento e detecção

A etapa final dessa série de experimentos avaliou a correlação entre a qualidade de agrupamento (pelos índices de Silhueta e Erro Quadrático Médio) e a qualidade de detecção (pelo Incremento na Diferença de Médias). A Figura 5.6 apresenta os valores de Incremento na Diferença de Médias (com intervalo de confiança de 90%), Silhueta e Erro Quadrático Médio em relação ao número de

centróides gerados pelos experimentos. Os índices de agrupamento são colocados em escala para comparação com o índice de diferenciação (no caso do Erro Quadrático é utilizado o valor inverso para que se observe correlação positiva dado que menores índices de erro quadrático estão associados a maior qualidade de agrupamento). Os gráficos são referentes à aplicação da Janela Deslizante sobre o conjunto de dados `WU-FTPD`, submetido a agrupamento pelos algoritmos *Leader-Follower* Adaptativo e rede GWR (i.e. os dois algoritmos adaptativos).



(a) *LF* Adaptivo (b) GWR

Figura 5.6: Correlação entre índices de agrupamento e detecção de novidades.

Conforme esperado, pode-se observar que uma melhora na qualidade de agrupamento (representada pelo aumento no valor da Silhueta e diminuição no valor do Erro Quadrático Médio) é acompanhada por uma maior separabilidade entre os comportamentos das séries. Também é observada alta correlação entre os índices de agrupamento e a capacidade de diferenciação entre situações normais e de ataque, o que mostra que a informação dos parâmetros é útil na caracterização do sistema e que as técnicas de agrupamento *on-line* são válidas para a discretização dessas informações.

Tabela 5.12: `rpc.statd`: Correlação do E.Q.

	LF Adaptativo	*Online KMLF*	*LF* Simples	*K-Means*	GWR
Entropia	0.70	0.96	0.89	0.77	0.60
Janela Deslizante	0.89	0.97	0.96	0.98	0.83
DTW	0.87	0.97	0.88	0.83	0.90
-	min:0.60, mean: 0.86, max: 0.98				

Tabela 5.13: `rpc.statd`: Correlação da Silhueta.

	LF Adaptativo	*Online KMLF*	*LF* Simples	*K-Means*	GWR
Entropia	0.39	0.91	0.78	0.47	0.35
Janela Deslizante	0.61	0.93	0.90	0.71	0.45
DTW	1.00	0.98	0.95	1.00	0.98
-	min:0.35, mean: 0.76, max: 1.00				

As Tabelas 5.12 and 5.13 apresentam a correlação entre os índices de agrupamento e diferenciação para todos os algoritmos quando aplicados ao

conjunto de dados `rpc.statd`, e as Tabelas 5.14 e 5.15 apresentam as correlações para o conjunto `WU-FTPD`. Exceto por eventuais *outliers*, a maioria dos algoritmos apresenta alta correlação entre os índices de agrupamento e detecção de novidades. Considerando-se 14 graus de liberdade, $\alpha \leq 0.05$ e uma distribuição de probabilidade t de *Student* bicaudal, o valor crítico para a correlação de pearson vale 0.49. Para os experimentos no conjunto de dados `WU-FTPD`, exceto para o algoritmo GWR, todos os valores observados se encontram acima desse limite. Quando se analisam todos os experimentos, 90% deles apresentam índices de correlação acima desse valor.

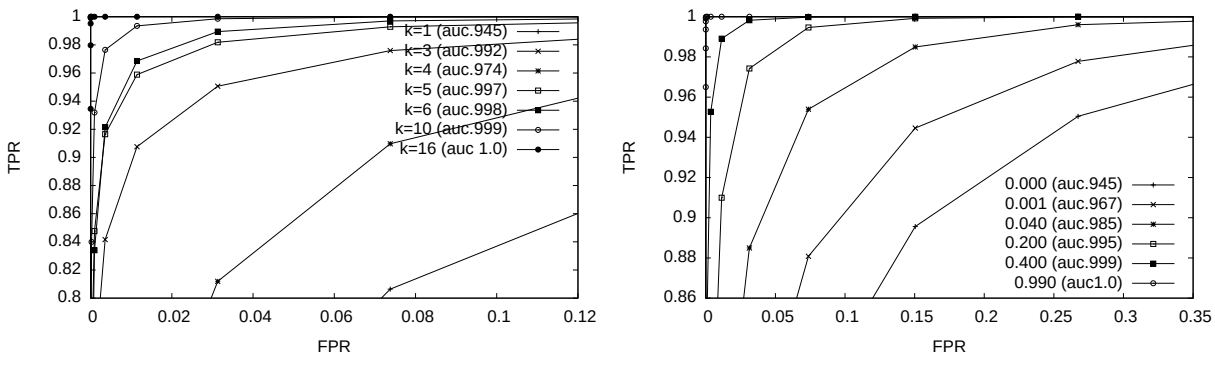
Tabela 5.14: `WU-FTPD`: Correlação do E.Q.

	LF Adaptativo	*Online KMLF*	*LF* Simples	*K-Means*	GWR
Entropia	0.90	0.98	0.99	0.81	0.22
Janela Deslizante	0.94	0.99	0.99	0.83	0.79
DTW	0.99	0.99	0.99	0.94	0.94
-	min:0.22, mean: 0.89, max: 0.99				

Tabela 5.15: `WU-FTPD`: Correlação da Silhueta.

	LF Adaptativo	*Online KMLF*	*LF* Simples	*K-Means*	GWR
Entropia	0.73	0.92	0.99	0.67	0.29
Janela Deslizante	0.83	0.96	0.97	0.67	0.51
DTW	0.83	0.95	0.97	0.84	0.76
-	min:0.29, mean: 0.79, max: 0.99				

Finalmente, são avaliadas as curvas ROC estimadas a partir dos índices de novidade observados nas sessões normais e de ataque. Utiliza-se a área sob a curva como índice de precisão de um classificador (em lugar de índices como acurácia ou *recall*) pelo fato de que esse índice não apresenta viés (Bradley, 1997) quando o número de observações em diferentes categorias não é balanceado (o que é o caso para a maioria dos conjuntos de dados de detecção de novidades).

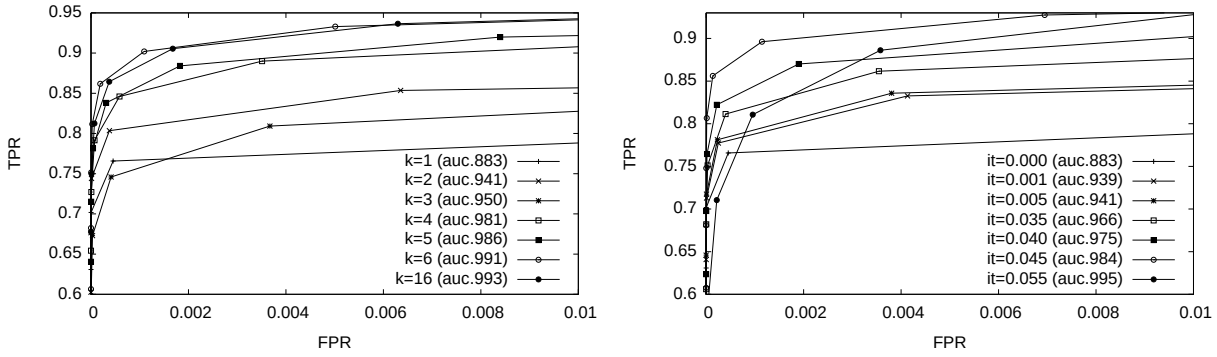


(a) Adaptive LF (b) GWR

Figura 5.7: Curvas ROC para o conjunto de dados `rpc.statd` quando a Entropia é utilizada como índice de novidade.

Para estimar essa curva, assume-se que os índices de novidade nas sessões normais e nas de ataque geram duas distribuições normais, e estima-se o valor da média e do desvio padrão dessas curvas a partir dos valores observados nos experimentos. A Figura 5.7 apresenta os índices de verdadeiro positivos (TPR) e falsos positivos (FPR), i.e. as curvas ROC, obtidas ao se aplicar a abordagem ao conjunto de dados `rpc.statd`, quando a Entropia é utilizada como índice de novidade (apenas as curvas mais relevantes são apresentadas). Em ambos os experimentos nota-se maior capacidade de diferenciação entre situações normais e ataques (representadas pelas curvas mais próximas ao ponto $(0, 1)$, com área sob a curva mais próxima de $1,0$) para experimentos com maior número de centróides (i.e. quando se observam melhores índices de agrupamento de dados, quando a série temporal conta com maior quantidade de informação agregada).

A Figura 5.8 apresenta as curvas ROC para o conjunto de dados `WU-FTPD`. Assim como nos experimentos apresentados anteriormente, observam-se melhores índices de classificação quando se observam melhores índices de agrupamento de dados. Os valores de área sob a curva são ligeiramente inferiores que os observados para o conjunto `rpc.statd` dado que esse conjunto é mais complexo e menos previsível.



(a) Adaptive LF (b) GWR

Figura 5.8: Curvas ROC `WU-FTPD` quando a Entropia é utilizada como índice de novidade.

Em todos os experimentos, quando se utiliza a DTW ou a Janela Deslizante como técnica de detecção de novidades, a área sob a curva varia de $0,979$ até $1,0$, dado que esses algoritmos apresentam maior capacidade de diferenciação quando as séries contam com menor quantidade de informação agregada. Por outro lado, a área sob a curva quando se utiliza a Entropia para diferenciar as séries varia de $0,59$ até $1,0$ (com média de $0,96$), de forma que é válida a análise de correlação entre os índices de agrupamento e de classificação. A Tabela 5.16 apresenta a os índices de correlação para os algoritmos adaptativos de agrupamento. Exceto para a rede GWR, quando aplicado ao conjunto de da-

dos `rpc.statd`, todos os experimentos apresentam altos índices de correlação (i.e. valores acima do valor crítico descrito acima), o que corrobora a hipótese principal deste trabalho.

Tabela 5.16: Índice de correlação entre área sob a curva ROC e índices de agrupamento.

	rpc.statd		WU-FTPD	
	LF Adaptativo	GWR	*LF* Adaptativo	GWR
AuC x E.Q.	0.82	0.24	0.97	0.90
AuC x Silhueta	0.90	0.24	0.90	0.78

5.3 Considerações finais

Os experimentos descritos neste capítulo dão crédito à hipótese de que as técnicas de agrupamento de dados são válidas para a geração de uma série temporal que representa o comportamento de um sistema, e que as técnicas de detecção de novidades são capazes de diferenciar as situações normais das situações de ataque. Isso pode ser observado pela correlação entre os índices de validação de ambas etapas, uma vez que se observa uma melhor caracterização do comportamento dos sistemas quando seus estados (representados, nos experimentos executados, pelas chamadas de sistema) são agrupados com maiores índices de qualidade.

Para serem úteis em ambientes reais, todas as abordagens utilizadas devem ser do tipo *on-line*, mantendo em memória apenas uma parcela da informação observada. Os experimentos mostram que os algoritmos *on-line* de agrupamento e de detecção de novidades apresentam precisão similar aos algoritmos *batch*, sendo válidos para a caracterização do comportamento de sistema.

Dado que todos os algoritmos de agrupamento apresentam índices de precisão compatíveis, a escolha do algoritmo para uma aplicação pode ser feita com base na sua facilidade de uso em um determinado contexto. Os algoritmos *Leader-Follower* Simples e *Online-K-Means* dependem da definição precisa do parâmetro θ, o que demanda uma análise exploratória do conjunto de dados, que pode ser inviável em um ambiente real. Em contrapartida, os algoritmos GWR e *Leader-Follower* Adaptativo adotam abordagens que dependem menos da definição de bons parâmetros iniciais. Comparando esses dois algoritmos, nota-se que o *Leader-Follower* Adaptativo é de uso mais simples por dois fatores: primeiro, ele não depende de uma função de mescla entre pontos, que de fato pode não ser viável em alguns contextos. Em segundo lugar, ele demanda a definição de um número menor de parâmetros, os quais tem estimativa mais simples.

A escolha do algoritmo de detecção de novidades, por outro lado, não é uma tarefa tão simples. A técnica da Entropia é a mais eficiente em relação a consumo de memória e processamento, permitindo uma caracterização razoável dos sistemas com reduzido consumo de recursos. A técnica da Janela Deslizante, por outro lado, consome uma maior quantidade de recursos, mas é capaz de lidar com uma quantidade maior de informações de forma mais estável. A escolha da técnica para uma determinada aplicação deve, dessa forma, considerar esses dois aspectos de estabilidade e desempenho.

CAPÍTULO

6

Avaliações adicionais sobre a abordagem proposta

Após a avaliação da eficácia da abordagem deste trabalho, por meio dos experimentos descritos no Capítulo 5, deve-se, por fim, avaliar o uso da técnica em varredura única (i.e. quando se gera a série temporal na mesma varredura que efetua o agrupamento de dados, sem uma segunda varredura para validação) utilizando funções de distâncias genéricas na etapa de agrupamento de dados (i.e. distâncias independentes do domínio em questão, teoricamente aplicáveis a qualquer tipo de dado), além de avaliar a escalabilidade da abordagem, com a aplicação da técnica em um conjunto de dados de maior escala. Por fim, a técnica deve ser aplicada a dados extraídos de um ambiente *Web*, avaliando a proposta principal deste trabalho.

A fim de analisar todos esses aspectos, três grupos de experimentos são conduzidos. O primeiro grupo avalia dois aspectos: a aplicação das distâncias genéricas em comparação com uma distância específica do domínio, além do uso de uma varredura única. Para isso é utilizado o conjunto de dados de chamadas de sistema, previamente utilizado no Capítulo 5, e os experimentos indicam que a técnica é capaz de diferenciar situações normais de ataques por meio de uma única varredura, além de mostrarem que as distâncias genéricas apresentam resultados compatíveis com uma distância específica do contexto em questão.

O segundo grupo de experimentos visa avaliar a escalabilidade da abordagem aplicando-a sobre um conjunto de dados de auditoria de sistema operacional. Nessa etapa são utilizadas algumas das saídas do módulo BSM (um módulo de auditoria do sistema operacional `Solaris`) provenientes do

conjunto de dados DARPA IDEval (DARPA, 1999), que totalizam cerca de 5 milhões de registros. Sendo esse conjunto de dados duas ordens de grandeza maior que o conjunto empregado anteriormente, a efetividade da técnica nesse contexto mostra que a técnica é funcional mesmo em um conjunto de dados com escala de produção.

Avaliada a consistência e a escalabilidade da técnica quando aplicada em varredura única, e tendo sido verificado que as funções genéricas são aplicáveis em diversos contextos, o terceiro grupo de experimentos avalia a proposta principal deste trabalho, que é a criação de um sistema autônomo de detecção de intrusões em aplicações *Web*. Uma vez que os conjuntos de dados normalmente apresentados na literatura não contém as informações necessárias para a avaliação dessa abordagem, foi montado um ambiente de simulação para coleta de informações a partir dos fluxos de dados observados nas aplicações. Nesse contexto também é possível observar que as situações de ataque e anomalia são diferenciáveis do comportamento normal da aplicação.

Este capítulo descreve, inicialmente, o ambiente de simulação, apresentando a aplicação cujo comportamento é simulado e à qual os ataques são direcionados, além de descrever o simulador de acessos, que tem as tarefas de executar os acessos normais e de disparar ataques. Em seguida é descrita a abordagem de extração de comportamento, além dos modelos de dados utilizados para representar o comportamento dos sistemas e aplicações de todos os conjuntos de dados avaliados. Por fim, é descrita a aplicação da abordagem e são discutidos os resultados dos experimentos conduzidos.

6.1 Ambiente de simulação

Para a coleta de dados sobre o comportamento de uma aplicação *Web*, foi montado um ambiente composto por um servidor *HTTP* com aplicações *Web* instaladas, além de um gerenciador de banco de dados utilizado por essas aplicações. Para a execução dos testes foi utilizado um simulador de acessos, para o qual são definidos *scripts* que executam tanto as sessões de acesso normal quanto os testes de penetração.

A identificação de uma vulnerabilidade e o procedimento para execução do teste de penetração utilizado neste trabalho segue a metodologia definida por Scarfone et al. (2008). Segundo essa metodologia, a execução de um teste de penetração deve, inicialmente, identificar uma vulnerabilidade em um sistema. Em seguida, deve-se explorar essa vulnerabilidade, obtendo-se acesso não autorizado ao sistema. O próximo passo consiste na escalada de privilégios, na qual o invasor obtém acesso a funcionalidades restritas do sistema, conseguindo, por exemplo privilégios de administrador. Finalmente, são ex-

ploradas as funcionalidades obtidas, com objetivo de extrair dados do sistema ou iniciar ataques a outros computadores.

De acordo com essa metodologia, a identificação de vulnerabilidades pode ser feita por meio de técnicas de auditoria, pela identificação de serviços irregulares, por varreduras a vulnerabilidades ou por consultas a bases públicas. As técnicas de auditoria buscam falhas de configuração em sistemas de segurança (como regras incompletas em *firewalls* ou SDI's). Já a identificação de serviços irregulares consiste no uso de ferramentas (chamadas *portscanners*) para identificar serviços que não deveriam estar em execução em determinadas máquinas (por exemplo, um computador que atua apenas como servidor *HTTP* não deveria a princípio oferecer um serviço `telnet`). As varreduras por vulnerabilidades, por sua vez, utilizam ferramentas similares a *portscanners*, mas que não se limitam a identificar os serviços em execução, uma vez que também avaliam os serviços, *softwares* e versões de produtos encontrados, gerando relatórios com as possíveis vulnerabilidades encontradas. Por fim, as bases de dados de vulnerabilidades são bases públicas que contém informações sobre vulnerabilidades descobertas em diversas aplicações.

Dado que o objetivo desta etapa do trabalho foi o de simular ataques a aplicações *Web*, a consulta a uma base de vulnerabilidades é tida como a alternativa mais apropriada para a escolha da vulnerabilidade a ser explorada. Utilizou-se neste trabalho a base de vulnerabilidades NIST-NVD (NVD, 2010), que consiste em um repositório de vulnerabilidades mantido pelo DHS (departamento norte-americano de segurança nacional (DHS, 2010)). Essa base de informações contém *checklists* para auditorias, falhas em *softwares* relacionadas a segurança, erros de configurações e métricas normalizadas de impacto das vulnerabilidades. Utilizam-se, neste trabalho, os registros de falhas de segurança em *softwares*, considerando o tipo de aplicação vulnerável e as métricas de impacto.

Para a execução dos testes foi selecionada a vulnerabilidade `CVE - 2010 - 2908`, referente a uma falha no módulo `Joomdle` (Joomdle, 2010) na versão 0.24 e anteriores. `Joomdle` é um módulo do gerenciador de conteúdo `Joomla` (Joomla, 2010), que o integra à ferramenta de *e-learning* `Moodle` (Moodle, 2010) por meio de *Web Services*. A vulnerabilidade permite a injeção de código *SQL* através da interface *Web* da aplicação, o que permite a execução de consultas arbitrárias na base de dados da ferramenta `Moodle` por meio da manipulação de um parâmetro de requisições enviadas por usuários não autenticados. Por ser um ataque de fácil condução, cujos resultados apresentam grande impacto principalmente à integridade e à confiabilidade das informações do sistema, o impacto dessa vulnerabilidade é classificado como alto, sendo assim apropriado para a simulação conduzida neste trabalho.

Para a exploração dessa vulnerabilidade, deve-se forjar uma requisição ao gerenciador de conteúdo manipulando-se um dos parâmetros de uma das funcionalidades do módulo `Joomdle`. O valor desse parâmetro não recebe nenhum tipo de validação e é enviado à ferramenta de *e-learning* por meio de *Web Service* de integração. Por fim, uma consulta à base de dados é formada pela concatenação de uma consulta codificada na aplicação ao parâmetro informado, e em seguida executada. Como em nenhuma etapa do processo esse valor é verificado, a aplicação permite que comandos enviados pelo usuário sejam executados na base de dados.

Optou-se por utilizar essa vulnerabilidade para a execução dos testes por três fatores principais. Em primeiro lugar, por se tratar de uma vulnerabilidade baseada em injeção de código *SQL* por meio de requisições *HTTP*, sendo assim representativa no contexto deste trabalho. Essa vulnerabilidade permite a extração irregular de informações da base de dados, o que viabiliza a posterior escalada de privilégios. Esses fatores viabilizam a composição de um teste de penetração em uma aplicação *Web* seguindo a metodologia definida por Scarfone et al. (2008).

O segundo aspecto em favor da escolha dessa vulnerabilidade é o fato das aplicações afetadas serem aplicações LAMP (i.e. aplicações para servidores de baixo custo, executando `Linux`, `Apache`, `MySQL` e `PHP`) bastante populares e livremente distribuídas na Internet. A popularidade das aplicações reafirma a importância do problema, e a livre distribuição das aplicações que compõe toda a pilha de software facilitam a montagem de um ambiente para simulação.

Por fim, o terceiro aspecto para a escolha foi o fato de que o módulo de integração das aplicações *Web* efetua as comunicações por meio de *Web Services*, permitindo o monitoramento de um fluxo de dados adicional no sistema. A Figura 6.1 apresenta o diagrama de instalação das aplicações, com destaque para os vetores dos ataque, i.e. os contextos que podem ser avaliados pela aplicação de detecção de intrusão (no caso, as requisições *HTTP* recebidas pelas aplicações, as mensagens trocadas pelos *Web Services* de integração, e os comandos *SQL* executados na base de dados).

Para a simulação de acessos foi utilizada a ferramenta `Selenium` (Selenium, 2010). Essa ferramenta permite que sejam gravadas sessões de acesso *Web* a partir de um navegador, e a partir dessas sessões são gerados *scripts* para a automação da execução dos testes. Essa execução automatizada inicializa um navegador *Web* qualquer, e controla seu comportamento, acessando a aplicação sob testes, executando os comandos programados, e validando as saídas da aplicação. Para a execução da simulação de acessos, foi montado um ambiente com um sistema gerenciador de bancos de dados e um servidor

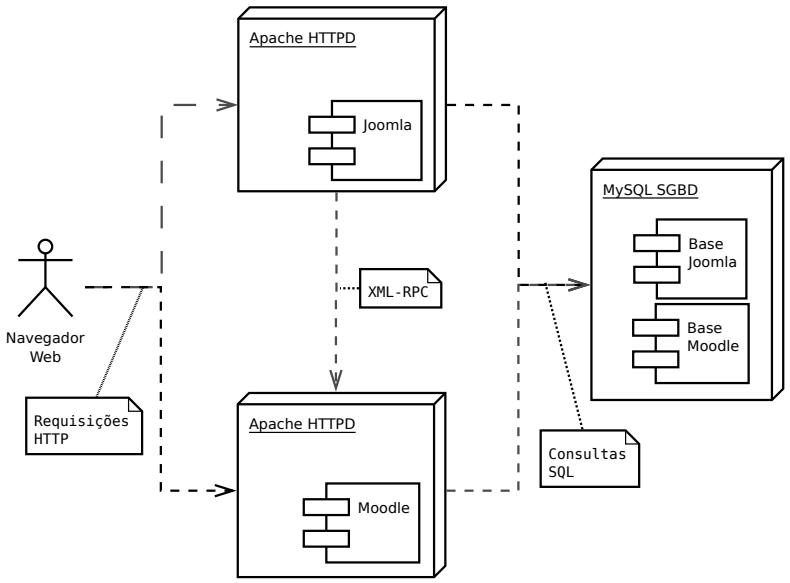


Figura 6.1: Diagrama de instalação e vetor de ataques.

HTTP, no qual as aplicações *Web* foram instaladas e configuradas. Em seguida foram definidos e executados os *scripts* que simulam os acessos normais e que executam os testes de penetração.

6.1.1 Simulação de comportamento normal

Para a simulação de comportamento normal da aplicação, foram definidos três casos de teste, denominados "criar usuário", "criar curso" e "navegação". O caso de teste "criar usuário" acessa a interface administrativa da aplicação `Moodle` e cria um lote de usuários (com tamanho variável), utilizando informações de uma base de nomes pessoais e números de identificação oriunda de uma lista de divulgação do vestibular da Fuvest (2010).

O caso de teste "criar curso" acessa a aplicação `Moodle` utilizando um perfil de professor e cria alguns cursos. Em seguida, o *script* associa um número variável de alunos a esse curso. As informações sobre os cursos (nome, código de identificação e descrição) são extraídas do sistema JúpiterWeb (2010) da Universidade de São Paulo.

Por fim, o caso de teste "navegação" acessa a aplicação `Joomdle` sem efetuar autenticação e visualiza as informações sobre cursos cadastrados no sistema `Moodle`, disponibilizadas por meio do módulo `Joomdle`. São executadas 30 instâncias de cada caso de teste, totalizando 90 instâncias que representam o funcionamento normal da aplicação.

6.1.2 Simulação de incidentes de ataque

Foram definidos dois casos de teste para simular as situações de ataque e de comportamento anômalo. O caso de teste "acesso à senha administrativa" explora a vulnerabilidade de injeção de código *SQL*, enquanto o caso de teste "extração de informações de usuários" acessa informações restritas após a obtenção do acesso administrativo.

O caso de teste "acesso à senha administrativa" explora a vulnerabilidade `CVE - 2010 - 2908`, que permite a injeção de código *SQL* pela manipulação do valor de um dos parâmetros da aplicação. Em situações normais, esse parâmetro recebe apenas valores numéricos e é concatenado a uma consulta executada no banco de dados sem qualquer tipo de validação ou tratamento. A manipulação desse valor permite que sejam executadas consultas arbitrárias no banco de dados, o que viabiliza a extração de quaisquer informações mantidas pelas aplicações. Essa vulnerabilidade é, dessa forma, utilizada para extrair do banco de dados informações sobre o usuário administrador, permitindo o acesso do invasor à interface administrativa.

Quadro 3 Requisição com injeção *SQL*

```
http://localhost/index.php
    ?option=com_joomdle
    &view=detail
    &cat_id=1:miscellaneous
    &course_id=b' and 1=2
                 union SELECT 1 as remoteid,
                              2 as cat_id,
                              username as cat_name,
                              NULL as cat_description,
                              5,
                              password,
                              'sn', '',
                              lastip,
                              lastlogin,
                              '', 'CUR', 10, 0, 0, 0, '',
                              0 as defaultroleid,
                              NULL as defaultrolename
                       from mdl_user
                       where id = 2
                             and 'a' = 'a
    &Itemid=1
```

A requisição que conduz o ataque é apresentada no Quadro 3 (apresenta-se a requisição com identação para melhor visualização). Os parâmetros *option* e *view* indicam a funcionalidade da aplicação a ser utilizada (no caso, a funcionalidade de detalhamento de curso, que apresenta a vulnerabilidade). Os

parâmetros cat_id, $course\_id$ e $Itemid$ são parâmetros de entrada para essa funcionalidade, e, conforme se observa, o valor do parâmetros $course\_id$ pode ser manipulado para alterar o comando *SQL* executado pela aplicação. O comando normalmente executado pela aplicação, que busca detalhes sobre um único curso, tem seu resultado truncado pela contradição $1 = 2$. Em seu lugar, o comando busca informações da tabela de usuários por meio da operação $union$ do banco de dados. A tautologia $'a' = 'a$ é adicionada ao final do comando para que seja consumido o último apóstrofe da consulta contida na aplicação e seja composta uma consulta *SQL* válida.

O resultado da injeção de código *SQL* é apresentado na Figura 6.2. Ao invés de exibir os detalhes sobre um curso, a tela apresenta informações sobre o usuário administrador do sistema. No campo referente ao título, onde normalmente é apresentado o nome do curso, observa-se o valor da *hash MD5* da senha do usuário, e no campo "categoria" verifica-se o nome de *login* do usuário administrador, valores que são então utilizados para obter acesso à interface administrativa.

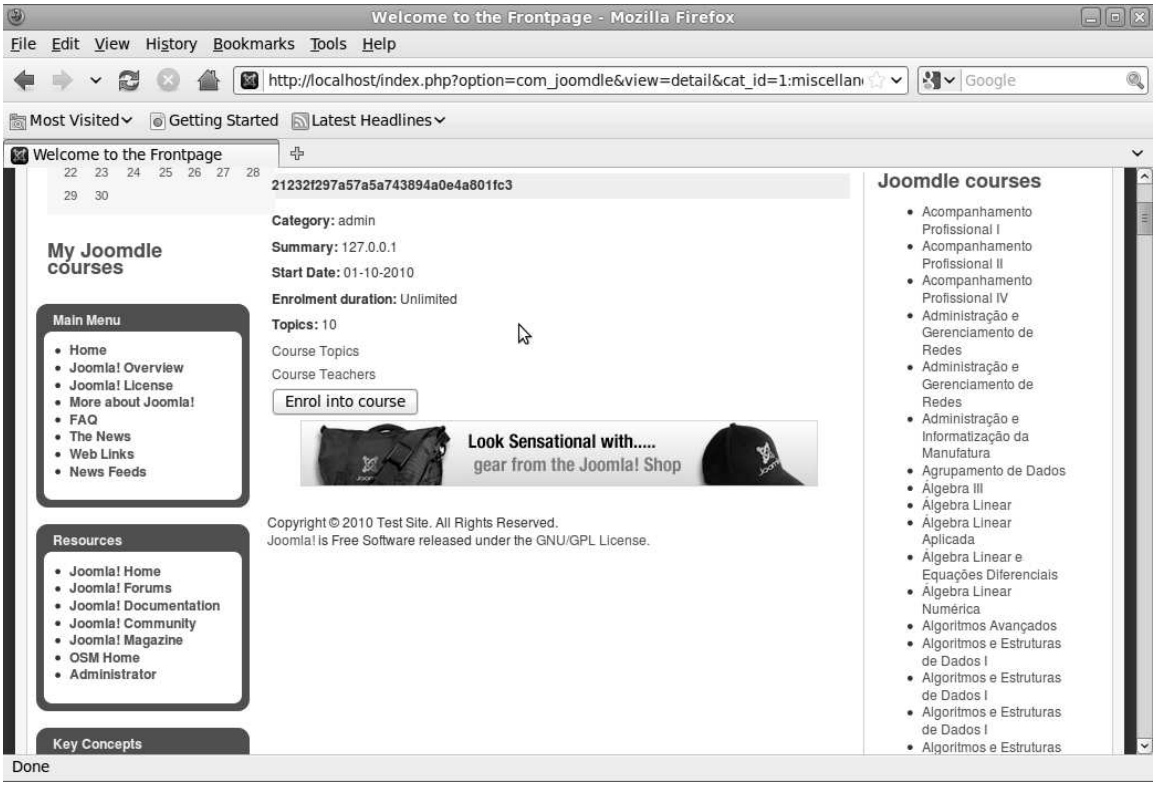


Figura 6.2: Informações sobre usuário administrador extraídas do sistema.

De posse do nome de usuário e senha administrativa, pode ser executado o caso de teste "extração de informações de usuários", que não explora uma vulnerabilidade *SQL*, mas representa uma anomalia no comportamento do sistema apresentada em virtude do incidente anterior. Nesse caso de teste o invasor acessa a interface administrativa do sistema e requisita uma listagem com as informações referentes a todos os usuários do sistema.

São executadas 5 instâncias de cada ataque, totalizando 10 sessões de ataque.

6.2 Definição dos modelos de dados

Os três conjuntos de dados utilizados nos experimentos finais (i.e. o conjunto de dados de caracterização de processos fornecido por Twycross e Aickelin (2010), o conjunto de dados de auditoria de sistema do DARPA IDEval (DARPA, 1999), e o conjunto de dados extraídos da simulação do comportamento da aplicação *Web*) são modelados como sequências de objetos, divididas em sessões que representam comportamento normal ou ataques. Os objetos que compõem cada tipo de série são apresentados nesta seção.

O conjunto de dados de caracterização de processos fornecido por Twycross e Aickelin (2010), utilizado também no Capítulo 5, é composto por sequências de chamadas de sistema invocadas durante a execução do software `WU-FTPD`, contendo o tipo da chamada, os valores dos parâmetros e o valor do retorno. Para melhor representar uma instância real da técnica proposta neste trabalho, nenhum pré-processamento é aplicado aos dados, uma vez que etapas de pré-processamento (como por exemplo normalização) são aplicáveis apenas a conjuntos de dados finitos. Os objetos são lidos do conjunto de dados e submetidos ao algoritmo de agrupamento em uma única varredura. Cada objeto contém o tipo de chamada de sistema, valor de retorno e lista de parâmetros. O tipo de dado dos parâmetros e do valor de retorno (numérico, nominal ou cadeia de caracteres) são previamente conhecidos a partir da especificação do sistema operacional. O conjunto de dados é composto por 55 sessões de comportamento normal e 4 sessões de ataque. Cada etapa de treinamento é executada utilizando-se grupos compostos por 5 sessões normais, e as etapas de validação utilizam todas as sessões disponíveis. Esse conjunto é composto por 85,290 objetos (i.e. chamadas de sistema).

O conjunto de dados de auditoria de sistema do DARPA IDEval (DARPA, 1999) é composto por listagens diárias das saídas do módulo BSM (*Basic Security Module*) do sistema operacional Solaris. Cada listagem apresenta todos os eventos de auditoria registrados por uma das máquinas sujeita a ataques. Cada registro é composto por uma sequência de *tokens*, que possuem tipo (e.g. referência a arquivo, endereço de rede, parâmetros de execução de um programa) e uma lista de valores. Os *tokens* são lidos a partir das listagens e agrupados em registros, que por sua vez são submetidos ao algoritmo de agrupamento. Para a etapa de caracterização de comportamento normal são considerados os registros provenientes da simulação de 6 dias livres de qualquer tipo de ataque, e para a etapa de detecção é utilizada uma sessão na

qual são observados 12 ataques com rastros nas saídas do módulo de auditoria. Esse subconjunto de dados conta com cerca de 38 milhões de *tokens*, que compõe 5,875,309 registros. Cada sessão contém cerca de 850,000 registros.

Por fim, é extraído um conjunto de dados de caracterização de comportamento de uma aplicação *Web* por meio de um ambiente simulado. Uma vez que o objetivo principal deste trabalho é a detecção de ataques de injeção de código *SQL* em aplicações *Web*, i.e. ataques de nível de aplicação, optou-se por monitorar três aspectos do sistema: o conteúdo das requisições enviadas pelo usuário, as requisições *Web Services* enviadas pelas aplicações e comandos executados no SGBD, uma vez que esses são os contextos onde mais se observam rastros desse tipo de ataque. A extração dos comandos executados no banco é uma tarefa trivial, dado que o SGBD utilizado pelas aplicações permite que essas informações sejam registradas em arquivo de *log*. Para a extração de dados referentes às requisições *HTTP* foi construído um módulo para o servidor Apache HTTPD (Kew, 2007) que atua como filtro de interceptação de requisições, extraindo as informações sobre as requisições geradas pelos usuários e sobre as mensagens *Web Services* trocadas pelas aplicações. São geradas 90 sessões de comportamento normal (com 30 sessões para cada *script* de teste) e 10 sessões de ataque (com 5 sessões de injeção de código e 5 sessões de comportamento anômalo). Para a aplicação da abordagem, cada etapa de treinamento é executada sobre grupos compostos por 18 sessões normais, das quais 6 são provenientes de cada tipo de teste. São extraídas 16,006 requisições *HTTP*, 2,789 requisições *XML-RPC* e 498,638 linhas de *log SQL* com a execução dos *scripts* de simulação

6.3 Descrição dos experimentos

Considerando-se a avaliação de diversos algoritmos de agrupamento de dados e detecção de novidades conduzida no Capítulo 5, optou-se por utilizar nos experimentos finais o algoritmo *Leader-Follower* Adaptativo para a tarefa de agrupamento de dados, dada sua facilidade de uso e precisão dos resultados, em conjunto à técnica da Entropia da Cadeia de Markov para a etapa de detecção de novidades, uma vez que a complexidade computacional linear dessa técnica é um ponto importante para a aplicação em conjuntos de dados de maior escala.

São conduzidos três grupos de experimentos na avaliação final da abordagem. No primeiro grupo deseja-se avaliar dois aspectos: a precisão da técnica quando aplicada por meio de varredura única, além da eficácia quando são utilizadas distâncias que não dependem do contexto em questão. Nos experimentos anteriores, as técnicas de agrupamento de dados são aplicadas por

meio de duas varreduras, a fim de aumentar a precisão dos índices de qualidade de agrupamento: a primeira varredura apenas definia os centróides que representavam o conjunto de dados, enquanto na segunda eram calculados os índices de qualidade de agrupamento e era definida a série temporal que caracterizava o comportamento do processo. Nessa segunda etapa uma única varredura é efetuada, i.e. a série temporal é gerada conforme os centróides são definidos.

Outro aspecto avaliado nesse experimento inicial é a aplicação de distâncias genéricas. Nos experimentos anteriores utilizou-se uma abordagem de agrupamento pelo tipo do objeto, onde objetos de tipos distintos não eram agrupados em um mesmo conjunto. Nessa nova etapa são aplicadas duas distâncias que não consideram o tipo do objeto (i.e. que podem associar objetos de tipos distintos em um mesmo grupo), e que podem ser aplicadas em qualquer domínio de aplicação, denominadas SSD (*simple syscall distance*) e NCD (*normalized compression distance* (Cilibrasi e Vitányi, 2005)). Além disso, o resultado dessas técnicas é comparado aos resultados de uma distância específica, denominada CSD (*complex syscall distance*).

CSD é uma distância similar à utilizada nos experimentos conduzidos no Capítulo 5, onde o agrupamento é aplicado segundo a abordagem hierárquica de acordo com o tipo da chamada de sistema. Essa distância se baseia em uma abordagem específica do contexto, sendo aplicável apenas a chamadas de sistema. Outra diferenças entre essa distância e a utilizada anteriormente é a forma de cálculo da distância entre atributos numéricos, que é dada por $|a - b|/max(a, b)$, dado que os valores numéricos não são normalizados. A distância SSD, por outro lado, representa a chamada de sistema como uma tupla composta pelo tipo da chamada, concatenado aos valores dos parâmetros e ao valor de retorno. A distância final é obtida pela distância Euclidiana entre as duas tuplas. Dessa forma, essa distância pode ser considerada genérica dado que é aplicável a qualquer objeto que possa ser representado como uma tupla, e não apenas a chamadas de sistema. Por fim, é avaliada a distância NCD quando aplicada aos objetos submetidos a um processo de serialização. Os experimentos conduzidos mostram que o agrupamento de dados em varredura única é capaz de gerar séries que caracterizam o comportamento com precisão suficiente para a diferenciação entre situações normais e ataques, e que as distâncias genéricas apresentam resultados compatíveis aos obtidos por meio da distância específica ao contexto.

O segundo grupo de experimentos avalia a escalabilidade da abordagem quando aplicada a um conjunto de dados de maior escala. Diversos trabalhos utilizam conjuntos de dados com escala de ambiente de produção, como o apresentado por Kruegel et al. (2003) (com cerca de 500.000 objetos) e por

Warrender et al. (1999) (com 45 milhões de objetos). Esses conjuntos não são aplicáveis neste trabalho uma vez que o primeiro não é público (por conter informações pessoais) e o segundo por não apresentar informações suficientes para a aplicação da abordagem. Decidiu-se portanto, neste trabalho, por utilizar um subconjunto das saídas do módulo BSM do conjunto de dados DARPA IDEval, com cerca de 5 milhões de registros. Apesar de conter as limitações descritas no Capítulo 4, esse conjunto de dados pode ser importante para a avaliação da escalabilidade da abordagem após a avaliação da sua consistência por meio de outros experimentos. Cada registro desse conjunto é composto por uma sequência de *tokens*, e a distância entre dois registros é dada pela distância Euclidiana entre seus tokens. O objetivo desse experimento é demonstrar que a técnica é capaz de diferenciar sessões com ataques de sessões normais, mesmo quando o número de objetos no conjunto de dados é alto.

O terceiro grupo de experimentos aplica a técnica proposta aos dados coletados durante a simulação da aplicação *Web*. São gerados três conjuntos de dados, denominados *HTTP*, *XML-RPC* e *SQL*, contendo, respectivamente, as requisições enviadas pelo simulador de acessos às aplicações *Web*, as mensagens *Web Services* trocadas entre as aplicações, e os comandos *SQL* executados no gerenciador de banco de dados. Quando aplicada a esse conjunto de dados, a técnica considera apenas a distância NCD sobre os objetos serializados.

A função de distância baseada em NCD utilizada nos experimentos serializa os objetos por meio do algoritmo `pickle` da linguagem `Python` (Python, 2010), e comprime as cadeias geradas com o algoritmo `gzip` com nível de compressão 1. Todos os objetos estão dentro do limite de $32KBytes$ necessário para a estabilidade da distância NCD (Cebrián et al., 2005). Todas as distâncias retornam valores normalizados no intervalo $[0.0, 1.0]$ entre os objetos avaliados.

6.4 Resultados

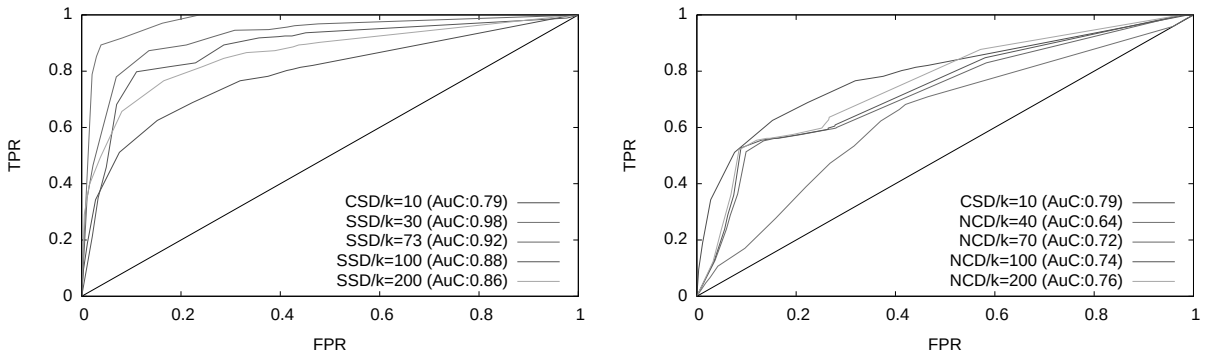
Nesta seção são apresentados os resultados da aplicação da técnica proposta sobre os três conjuntos de dados. Inicialmente são apresentados os resultados da técnica quando aplicada às chamadas de sistema, onde se observa que a técnica é efetiva mesmo quando se conta com uma caracterização simplória do contexto (i.e. quando não se utilizam distâncias específicas para um determinado domínio) e quando a geração das séries temporais ocorre em uma única varredura. Em seguida, são apresentados os resultados com o conjunto de dados DARPA IDEval, onde se avalia a escalabilidade da técnica. Por fim, são apresentados os resultados da aplicação da técnica sobre o conjunto

de dados proposto, onde se observa que a técnica é capaz de detectar ataques e anomalias no comportamento de uma aplicação *Web*.

A efetividade das técnicas é avaliada por meio de curvas ROC geradas a partir dos índices de novidade fornecidos pela técnica de detecção de novidades. Utiliza-se também a área sob essa curva como indicador de capacidade de diferenciação entre situações normais e ataques.

6.4.1 Detecção de ataques em chamadas de sistema

As Figuras 6.3a e 6.3b apresenta algumas das curvas ROC obtidas ao se utilizarem duas distâncias independentes do domínio (SSD e NCD), em relação a uma das curvas obtidas ao se utilizar a distância específica para chamadas de sistema (CSD). A curva da distância CSD apresentada exibe o resultado do experimento quando $k = 10$ (onde k é o número de centróides na etapa de agrupamento), o que representa um experimento onde o valor da área sob a curva é próximo ao valor médio observado em todos os experimentos. Observa-se nesse subconjunto dos resultados que a distância SSD apresentou resultados melhores que a técnica que considera o contexto (representado pelas curvas mais próximas ao vértice $(0, 1)$), enquanto que a distância NCD apresentou resultados ligeiramente inferiores.



(a) Distância simples (b) Distância NCD

Figura 6.3: Curvas ROC para conjunto de dados de chamadas de sistema.

A Figura 6.4 apresenta os valores da área sob a curva ROC em relação ao parâmetro k da etapa de agrupamento, para as três distâncias avaliadas (a linha horizontal apresenta o valor da média e barras verticais apresentam o desvio padrão desse valor). Para comparar os resultados das técnicas independentes de contexto (SSD e NCD) com a técnica específica para o contexto de chamadas de sistemas (CSD), são executados dois testes de hipótese que consideram a média e a dispersão desse valor para todos os experimentos executados.

No primeiro teste são comparados os resultados das distâncias NCD e CSD. Aplica-se um teste *t* de Welch bicaudal não-pareado para avaliar se as médias

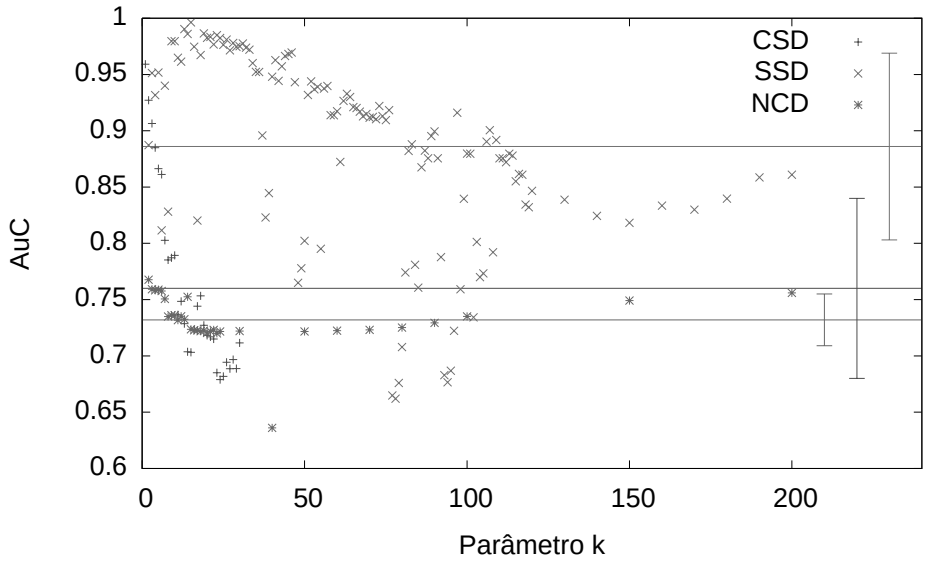


Figura 6.4: Área sob curvas ROC em relação ao parâmetro k.

são significativamente distintas. A distância NCD apresenta média de área sob a curva de $0,732$ e desvio padrão de $0,02$, enquanto a distância CSD apresenta média de $0,76$ com desvio padrão de $0,08$. Formulam-se então as hipóteses:

- Hipótese nula($H0$): $\mu_{csd} = \mu_{ncd}$
- Hipótese alternativa($H1$): $\mu_{csd} \neq \mu_{ncd}$

A diferença entre as médias apresenta um *t score* de $-1,85$, de forma que com 33 graus de liberdade se observa uma probabilidade de $0,07$ de se observar essa diferença por acaso, i.e. não é possível rejeitar essa hipótese com $\alpha < 0,05$. Dessa forma, apesar do uso da distância específica apresentar resultados ligeiramente superiores à distância NCD, não se pode afirmar que as técnicas apresentam resultados significativamente distintos.

No segundo teste são comparados os resultados da distância SSD com a distância CSD. Aplica-se novamente um teste *t* de Welch bicaudal não-pareado para avaliar se a distância SSD apresenta resultados mais efetivos que a técnica específica do contexto. A distância SSD apresenta média de área sob a curva de $0,886$ com desvio padrão de $0,08$. Formulam-se então as hipóteses:

- Hipótese nula($H0$): $\mu_{ssd} \leq \mu_{csd}$
- Hipótese alternativa($H1$): $\mu_{ssd} \neq \mu_{csd}$

A diferença entre as médias apresenta um *t score* de $-7,7$. Com 44 graus de liberdade, observa-se uma probabilidade da ordem de 10^{-9} de se observar essa diferença entre médias por acaso, de forma que é possível rejeitar a hipótese nula, uma vez que o desempenho da distância genérica foi significativamente superior ao desempenho da distância específica ao domínio.

Esses resultados experimentais corroboram a hipótese de que as técnicas que não consideram informações de contexto são capazes de caracterizar o comportamento de uma aplicação, e que as técnicas de agrupamento de varredura única podem ser utilizadas nessa etapa inicial de caracterização. Com isso, observa-se que é possível diferenciar ataques de situações normais nesse conjunto de dados, por meio de uma abordagem autônoma e independente do modelo de dados que descreve o comportamento da aplicação.

6.4.2 Avaliação da escalabilidade

Avaliada a efetividade da técnica no conjunto de dados de chamadas de sistema, deseja-se avaliar a escalabilidade da técnica aplicando-a a um conjunto de dados de maior ordem de grandeza. A Figura 6.5 apresenta algumas das curvas ROC obtidas quando a técnica é aplicada ao conjunto de dados DARPA IDEval.

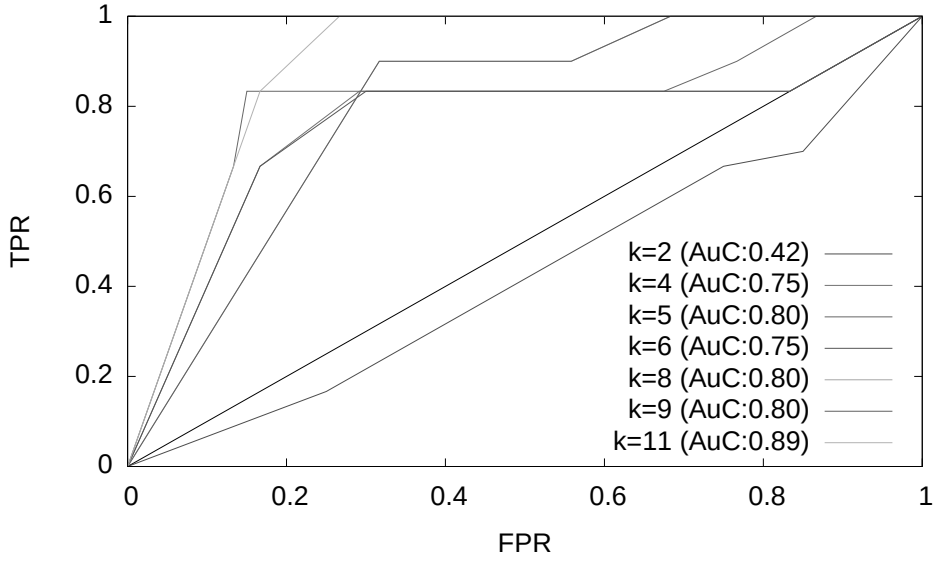


Figura 6.5: Curvas ROC para conjunto de dados DARPA IDEval.

Quando o valor do parâmetro k varia entre 2 e 14, a área sob a curva ROC varia de $0,43$ a $0,89$, com valor médio de $0,64$. Para se verificar que a técnica é efetiva nessa aplicação, pode-se avaliar se o valor dessa média é maior que $0,5$, i.e. que a técnica tem maior capacidade de diferenciação que um classificador aleatório. Para isso, é executado um teste considerando uma distribuição de probabilidades *t* de *Student* monocaudal para uma amostra, uma vez que o número de observações é reduzido e deseja-se observar se a média de uma população é maior que um valor de corte. Assim, formulam-se duas hipóteses:

- Hipótese nula($H0$): $\mu \leq 0,5$
- Hipótese alternativa($H1$): $\mu > 0,5$

Com valor médio de $0,64$, desvio padrão de $0,165$ e 12 graus de liberdade, a probabilidade da média ser menor que $0,5$ é de $0,005$, de forma que a hipótese nula pode ser rejeitada com $\alpha < 0,01$. Como o valor da média é significantemente maior que $0,5$, pode-se afirmar que a técnica é aplicável com sucesso mesmo quando os conjuntos de dados tem maior ordem de grandeza.

6.4.3 Detecção de ataques à aplicação Web

Por fim, são apresentados os resultados da diferenciação entre situações normais e ataques nos dados extraídos da aplicação *Web*. A Figura 6.6 apresenta algumas das curvas ROC obtidas quando a abordagem é aplicada ao conjunto de dados de requisições *HTTP* submetidas pelos usuários. A técnica é mais efetiva quando o parâmetro k varia de 4 a 19, quando se observam áreas sob a curva ROC maiores que $0,5$. Para esse conjunto de dados, tendo $k \leq 3$, a etapa de agrupamento não é capaz de agregar à série informações suficientes para caracterizar o comportamento da aplicação, e quando $k \geq 20$ observam-se casos de *overfitting*, i.e. a quantidade de informação adicionada à série passa a dificultar a caracterização do comportamento dos processos. Com $4 \leq k \leq 19$, observa-se uma área média sob a curva ROC de $0,68$.

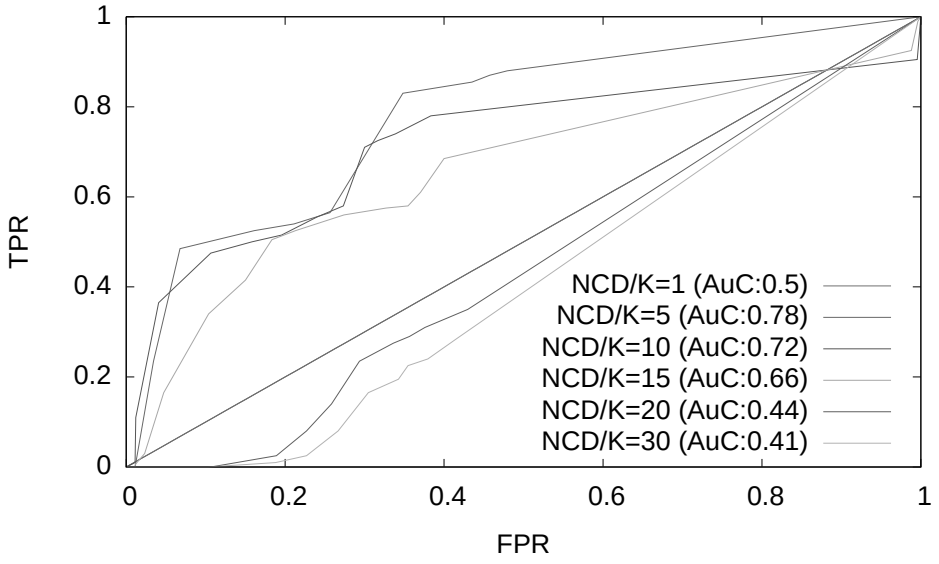


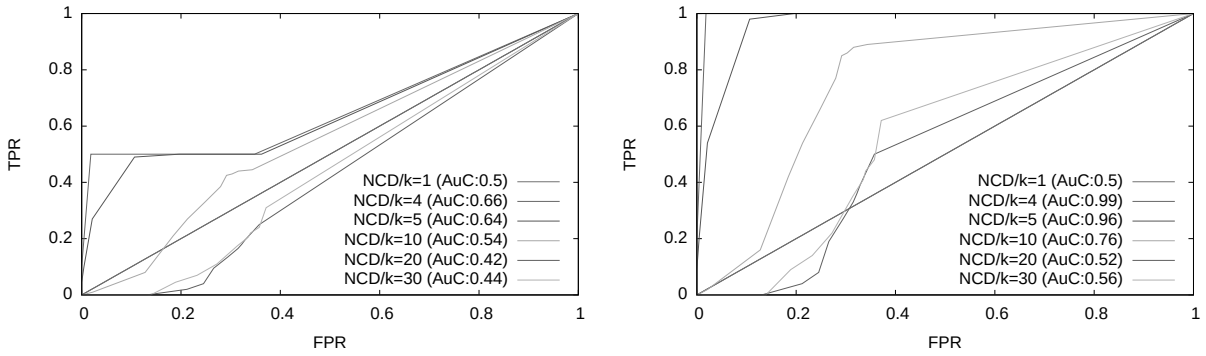
Figura 6.6: Curvas ROC para conjunto de dados *HTTP*.

Para avaliar a efetividade da técnica, utiliza-se um teste que considera uma distribuição de probabilidades *t* de *Student* monocaudal para uma amostra visando avaliar se o valor médio da área sob a curva se posiciona acima da média de um classificador aleatório. Para isso, formulam-se as hipóteses:

- Hipótese nula($H0$): $\mu \leq 0,5$
- Hipótese alternativa($H1$): $\mu > 0,5$

Quando são considerados todos os casos de teste, i.e. $1 \leq k \leq 30$, observa-se média de $0,56$ com desvio padrão de $0,14$. Com 29 graus de liberdade, a probabilidade da média ser menor que $0,5$ vale $0,008$, de forma que a hipótese nula ainda possa ser rejeitada com $\alpha < 0,01$. Esses resultados confirmam que a técnica é capaz de detectar ataques nesse contexto mais complexo, desde que se observem bons resultados na etapa de agrupamento (i.e. evitando-se generalizações excessivas e casos de *overfitting*).

A segunda etapa de testes no conjunto de dados de aplicações *Web* avalia a efetividade da técnica quando aplicada ao conjunto de dados contendo as mensagens *XML-RPC* trocadas entre as aplicações. As Figuras 6.7a e 6.7b apresenta algumas das curvas ROC geradas a partir desse experimento. São avaliadas duas situações: na primeira, assim como nos outros experimentos, consideram-se como ataques os casos de teste "acesso a senha administrativa" (i.e. ataque com injeção de *SQL*) e "extração de informações de usuários" (anomalia de comportamento do usuário). Na segunda situação, considera-se como ataque apenas o caso de teste "acesso a senha administrativa". Comparando os resultados das Figuras 6.7a e 6.7b, observa-se que ao serem consideradas as situações anômalas, a eficiência da técnica cai pela metade, pelo fato de que a técnica não foi capaz de detectar as situações anômalas como incidentes. Isso ocorre pelo fato dos casos de teste anômalos não utilizam a funcionalidade de *Web Services*, de forma que não se observa nenhuma irregularidade quando esse caso de teste é executado.



(a) Ataques e anomalias

(b) Apenas ataques

Figura 6.7: Curvas ROC para conjunto de dados XML-RPC.

Quando são considerados tanto ataques quanto anomalias, um teste com distribuição de probabilidades *t* de *Student* bicaudal de localidade para uma amostra se mostra inconclusivo para afirmar se a técnica é superior a um classificador aleatório. Por outro lado, quando se consideram apenas os casos de teste onde ocorre a injeção *SQL*, tendo $1 \leq k \leq 30$, observa-se uma média da área sob a curva de $0,64$ com desvio padrão de $0,16$. Com 29 graus de liberdade, um teste com distribuição de probabilidades *t* de *Student* monocaudal

para uma amostra apresenta probabilidade da média menor que $0,5$ na ordem de 10^{-4}. Com isso, a hipótese nula pode ser rejeitada, o que indica que é possível detectar os ataques que trafegam por esse vetor, apesar da técnica não ser capaz de detectar as anomalias desse conjunto de dados (dado que não se observam traços de qualquer tipo de comportamento anômalo nessas situações). Além disso, assim como nos testes com dados de requisições *HTTP*, observa-se na Figura 6.7 uma queda na capacidade de detecção nos casos onde ocorre *overfitting*.

A etapa final de testes nesse conjunto de dados aplica a técnica proposta aos comandos *SQL* executados pela aplicação. A Figura 6.8 apresenta alguns dos resultados da aplicação da técnica nesse conjunto de dados.

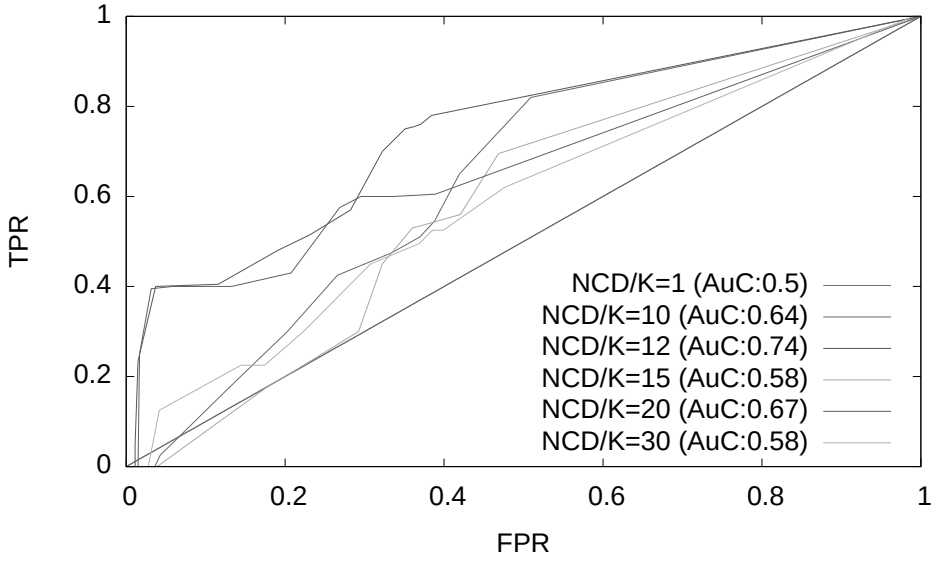


Figura 6.8: Curvas ROC para conjunto de dados *SQL*.

Ao serem considerados todos os casos de teste, i.e. $1 \leq k \leq 30$, observa-se média de $0,54$, com desvio padrão de $0,13$. Com 29 graus de liberdade, a probabilidade da média ser menor que $0,5$ vale $0,03$, de forma que a hipótese nula só pode ser rejeitada com $\alpha < 0,05$. Dessa forma, observa-se nesse conjunto de dados uma menor capacidade de diferenciação entre as anomalias e o comportamento normal que é observado nos experimentos anteriores. Atribui-se esse menor desempenho à maior quantidade de informação presente no conjunto de dados, e à leniência que a distância NCD apresenta quando os objetos apresentam maior quantidade de informação em comum. Uma alternativa para aumentar a precisão nesse caso seria o uso de uma distância específica para o domínio (por exemplo, ao utilizar-se uma distância que penalize inserções de *tokens*, o que dificulta a manipulação da estrutura das consultas), onde a perda de autonomia é acompanhada por um ganho em precisão. A ideia principal nesse caso foi apresentar o fato de que apesar de se observar menor desempenho, a técnica autônoma ainda é capaz de caracterizar o comporta-

mento do sistema com alguma precisão.

6.5 Considerações finais

Os experimentos descritos neste capítulo mostram que a abordagem de uso de agrupamento de dados e detecção de novidades é válida para a caracterização de comportamentos pouco estruturados, como de uma aplicação *Web*, com precisão suficiente para a detecção de intrusões do tipo "injeção de *SQL*", quando se utilizam algoritmos de varredura única para agrupamento de dados e detecção de novidades.

Em todas as técnicas apresentadas neste capítulo observam-se índices de área sob a curva relativamente mais baixos em comparação a outros trabalhos de detecção de intrusão (onde a área sob a curva frequentemente tem valores maiores que $0,9$), ou mesmo em relação aos experimentos conduzidos no capítulo anterior. É válido porém ressaltar que o objetivo deste trabalho é compor uma técnica genérica o suficiente para ser aplicável com pouco conhecimento sobre um determinado domínio, o que facilita a sua aplicabilidade em contextos autônomos, e que trabalhe com varredura única do conjunto de dados. Nos experimentos aqui conduzidos houve maior interesse em mostrar a efetividade da técnica (expressa pela área sob a curva maior que $0,5$, independentemente de parâmetros) do que apresentar a efetividade da técnica quando as funções são próprias ao contexto em questão e os parâmetros das técnicas são bem definidos, uma vez que se considera que em uma aplicação real não seja possível a definição precisa dos valores desses parâmetros.

A autonomia da aplicação dessa técnica é importante não apenas por simplificar a instalação de um produto em um novo contexto, mas por permitir que sejam avaliados diversos aspectos de uma aplicação sem intervenção humana. Esse aspecto viabilizaria por exemplo uma abordagem baseada em fusão de dados de múltiplos sensores: uma vez que a caracterização do comportamento de um sistema pode ser automatizada, uma abordagem onde diversos sensores monitoram diferentes aspectos de um sistema e técnicas de aprendizado de máquina são empregadas para a fusão desses sinais, seria a princípio possível a detecção de intrusões com uma abordagem totalmente independente da intervenção de um especialista.

Uma das maiores limitações dos experimentos conduzidos é o fato de se trabalhar com um ambiente simulado. Apesar de ser utilizada uma aplicação real, a simulação de ataques é feita de forma sintética, o que pode não representar a carga de trabalho ou o tipo de comportamento de um sistema real. Conforme se observa na literatura (McHugh, 2000), a questão de como se obter esse tipo de informação, com confiabilidade e sem comprometer informações

confidenciais, ainda é aberta.

CAPÍTULO

7

Conclusões

Esta dissertação de mestrado visou o estudo de técnicas de caracterização de comportamento de processos a fim detectar intrusões em nível de aplicação em sistemas *Web*. Como resultado, propôs-se uma técnica baseada em agrupamento de dados e detecção de novidades que utiliza distâncias genéricas e varredura única de conjunto de dados, que pode ser aplicada de maneira autônoma a sistemas reais.

A abordagem proposta aplica uma etapa de agrupamento de dados sobre as saídas de um sistema, de forma que essa etapa agrupa observações similares a fim de gerar uma série temporal que represente o comportamento da aplicação. Novidades observadas nessa série, que pode ter como origem um identificador incomum proveniente da etapa de agrupamento ou uma sequência inesperada de identificadores, são utilizadas para detectar anomalias, que neste trabalho são consideradas incidentes de intrusão.

Diversos experimentos foram executados a fim de avaliar os resultados da abordagem. O primeiro grupo de experimentos aplica a técnica em uma situação ideal, com pré-processamento dos dados e duas varreduras do conjunto a fim de se obterem índices mais precisos. Observa-se alta correlação entre os índices de qualidade das etapas de agrupamento de dados e detecção de novidades, o que mostra que a capacidade de diferenciação entre ataques e situações normais é diretamente proporcional à qualidade da caracterização de um sistema. O segundo grupo de experimentos generaliza a solução, utilizando-a em um conjunto de dados de maior escala, e em um conjunto de dados oriundo de uma simulação de uma aplicação *Web*. Esse grupo de experimentos mostra que a técnica é capaz de lidar com um conjunto de dados com escala de produção, além de validar a hipótese principal desta dissertação.

Observa-se ainda que é viável a proposta de um sistema autônomo de detecção de intrusão, o que permite, em trabalhos futuros, que se utilizem técnicas de fusão de alertas para, a partir de diversos aspectos de um sistema, caracterizados sem intervenção humana, se utilizar uma abordagem não supervisionada para detectar intrusões. Conforme se observa em outros trabalhos (Twycross e Aickelin, 2010), esse tipo de abordagem é bastante precisa uma vez que considera relações de causa e efeito entre anomalias observadas nos vetores de ataques e os efeitos colaterais desse tipo de operação.

A partir dos resultados dos experimentos conduzidos para a composição dessa dissertação, a seguinte publicação foi gerada até o momento: "Caracterização de Comportamento de Sistemas por meio de Agrupamento de Dados e Detecção de Novidades" – ERAD-SP 2010. Além disso, os seguintes artigos foram submetidos e estão em processo de revisão: "Intrusion Detection in Unstructured Contexts Using On-line Clustering and Novelty Detection" - *Computers and Security* (0167 – 4048), e "*Autonomic Intrusion Detection for Web Applications*" - SBRC 2011.

Referências Bibliográficas

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: *Proceedings of the 29th international conference on Very large data bases - Volume 29*, p. 81–92, 2003.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for projected clustering of high dimensional data streams. In: *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, p. 852–863, 2004.

ALBERTINI, M. K.; DE MELLO, R. F. *A self-organizing neural network to approach novelty detection* IGI Global, p. 1–20, 2009.

ANDERSON, R. J. *Security engineering: A guide to building dependable distributed systems*. 1 ed. Wiley, 2001.

APPARMOR Apparmor website. `http://www.novell.com/ linux/ security/ apparmor/`, 2009.

ASTROM, K. J.; MURRAY, R. M. *Feedback systems: An introduction for scientists and engineers*. 1 ed. Princeton University Press, 2008.

AXELSSON, S. *Intrusion detection systems: A survey and taxonomy*. Relatório Técnico, Chalmers University, 2000.

BABCOCK, B.; BABU, S.; DATAR, M.; MOTWANI, R.; WIDOM, J. Models and issues in data stream systems. In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, p. 1–16, 2002.

BRADLEY, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, v. 30, n. 7, p. 1145–1159, 1997.

BREACH *The web hacking incidents database annual report*. Relatório Técnico, Web Application Security Consortium, 2008.

BRUGGER, T. Kdd cup '99 dataset (network intrusion) considered harmful. http://www.kdnuggets.com/ news/ 2007/ n18/ 4i.html, 2007.

CAIS Vulnerabilidade no xmlrpc para php. Centro de Atendimento a Incidentes de Segurança (CAIS) da Rede Nacional de Ensino e Pesquisa (RNP), http://www.rnp.br/ cais/ alertas/ 2005/ cais-alr-20051107.html, 2005.

CAIS Ataques de injeção de código php em aplicações diversas. Centro de Atendimento a Incidentes de Segurança (CAIS) da Rede Nacional de Ensino e Pesquisa (RNP), http://www.rnp.br / cais/ alertas/ 2006/ cais-alr-20060323.html, 2006a.

CAIS Múltiplas vunerabilidades em produtos oracle. Centro de Atendimento a Incidentes de Segurança (CAIS) da Rede Nacional de Ensino e Pesquisa (RNP), http://www.rnp.br / cais/ alertas/ 2006/ oracle200601.html, 2006b.

CAIS Vulnerabilidade no microsoft exchange server owa. Centro de Atendimento a Incidentes de Segurança (CAIS) da Rede Nacional de Ensino e Pesquisa (RNP), http://www.rnp.br / cais/ alertas/ 2006/ MS06-029.html, 2006c.

CANNINGS, R.; DWIVEDI, H.; LACKEY, Z. *Hacking exposed web 2.0: Web 2.0 security secrets and solutions*. 1 ed. Mc Graw Hill, 2008.

CEBRIÁN, M.; ALFONSECA, M.; ; ORTEGA, A. Common pitfalls using the normalized compression distance: What to watch out for in a compressor. *Communications in Information and Systems*, p. 367–384, 2005.

CHARIKAR, M.; CHEKURI, C.; FEDER, T.; MOTWANI, R. Incremental clustering and dynamic information retrieval. In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, p. 626–635, 1997.

CILIBRASI, R.; VITÁNYI, P. M. B. Clustering by compression. *IEEE Transactions on Information Theory*, p. 1523–1545, 2005.

CRISCIONE, C.; SALVANESCHI, G.; MAGGI, F.; ZANERO, S. Integrated detection of attacks against browsers, web applications and databases. In: *European Conference on Computer Network Defense (EC2ND)*, p. 37–45, 2009.

CUNNINGHAM, R. K.; LIPPMANN, R. P.; FRIED, D. J.; GARFINKEL, S. L.; GRAF, I.; KENDALL, K. R.; WEBSTER, S. E.; WYSCHOGROD, D.; ZISSMAN, M. A.

Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. In: *Proceedings of Third Conference and Workshop on Intrusion Detection and Response*, p. 1–5, 1999a.

CUNNINGHAM, R. K.; LIPPMANN, R. P.; FRIED, D. J.; GRAF, I.; KENDALL, K. R.; WEBSTER, S. E.; ZISSMAN, M. A. Results of the 1999 darpa offline intrusion detection evaluation. In: *Second International Workshop on Recent Advances in Intrusion Detection (RAID 1999)*, p. 1–22, 1999b.

DARPA Information systems technology - darpa intrusion detection evaluation. `http://www.ll.mit.edu/ mission/ communications/ ist/ corpora/ ideval/ index.html`, 1999.

DASGUPTA, D.; GONZALEZ, F. An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions on Evolutionary Computation*, p. 281–291, 2002.

DHS United states department of homeland security website. `http://www.dhs.gov/`, 2010.

DODONOV, E.; DE MELLO, R. F. A novel approach for distributed application scheduling based on prediction of communication events. *Future Gener. Comput. Syst.*, p. 740–752, 2010.

ESKIN, E.; ARNOLD, A.; PRERAU, M.; PORTNOY, L.; STOLFO, S. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In: *Applications of Data Mining in Computer Security*, p. 1–20, 2002.

FORREST, S.; HOFMEYR, S. A.; SOMAYAJI, A.; LONGSTAFF, T. A. A sense of self for unix processes. In: *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, p. 120–128, 1996.

FUVEST Fuvest: Fundação universitária para o vestibular. `http://www.fuvest.br/`, 2010.

GOODALL, D. W. A new similarity index based on probability. *Biometrics, Vol. 22, No. 4*, p. 882–907, 1966.

GREENSQL Greensql website. `http://www.greensql.net/`, 2009.

GUPTA, S. K.; RAO, K. S.; BHATNAGAR, V. *K-means clustering algorithm for categorical attributes* Springer-Verlag, p. 203–208, 1999.

HAMMING, R. Error detecting and error correcting codes. *Bell System Technical Journal*, p. 147–160, 1950.

HAY, A.; CID, D.; BRAY, R. *Ossec host-based intrusion detection guide*. 1 ed. Syngress, 2008.

HE, J.; TAN, A.-H.; TAN, C.-L. Modified art 2a growing network capable of generating a fixed number of nodes. *IEEE Transactions on Neural Networks*, p. 728–737, 2004.

HOFMEYR, S. A.; FORREST, S.; SOMAYAJI, A. Intrusion detection using sequences of system calls. *Journal of Computer Security*, p. 151–180, 1998.

HUANG, Z. *Extensions to the k-means algorithm for clustering large data sets with categorical values* Kluwer Academic Publishers, p. 283–304, 1998.

ISHII, R. P.; DE MELLO, R. F. A history-based heuristic to optimize data access in distributed environments. *The 21st IASTED International Conference on Parallel and Distributed Computing and Systems*, p. 1–8, 2009.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. 1 ed. Prentice Hall, 1988.

JOOMDLE Joomdle - joomla-moodle integration website. `http://www.joomdle.com/`, 2010.

JOOMLA Joomla website. `http://www.joomla.org/`, 2010.

JÚPITERWEB Website júpiterweb - sistema de graduação. `http://sistemas2.usp.br/jupiterweb/`, 2010.

KEOGH, E.; RATANAMAHATANA, C. A. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, p. 358–386, 2005.

KEW, N. *The apache modules book: application development with apache*. 1 ed. Pearson Education, Inc., 2007.

KOLMOGOROV, A. N. Three approaches to the quantitative definition of information. *Problems of Information and Transmission*, v. 1, n. 1, p. 1–7, 1965.

KRUEGEL, C.; MUTZ, D.; VALEUR, F.; VIGNA, G. On the detection of anomalous system call arguments. In: *Proceedings of the 8th European Symposium on Research in Computer Security*, p. 326–343, 2003.

LBNL The internet traffic archive. `http://ita.ee.lbl.gov/`, 1998.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, v. 10, n. 8, p. 707–710, 1966.

LIPPMANN, R. P.; FRIED, D. J.; GRAF, I.; HAINES, J. W.; KENDALL, K. R.; MCCLUNG, D.; WEBER, D.; WEBSTER, S. E.; WYSCHOGROD, D.; CUNNINGHAM, R. K.; ZISSMAN, M. A. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. *DARPA Information Survivability Conference and Exposition*, p. 11–26, 2000.

MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, p. 281–297, 1967.

MAGGI, F.; MATTEUCCI, M.; ZANERO, S. Detecting intrusions through system call sequence and argument analysis. *IEEE Transactions on Dependable and Secure Computing*, v. 99, n. 1, p. 1–15, 2009.

MAHONEY, M. V.; CHAN, P. K. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. *Lecture Notes in Computer Science*, p. 220–237, 2003.

MARKOU, M.; SINGH, S. Novelty detection: a review, part 1: statistical approaches. *Signal Processing*, p. 2481–2497, 2003a.

MARKOU, M.; SINGH, S. Novelty detection: a review, part 2: neural network based approaches. *Signal Processing*, p. 2499–2521, 2003b.

MARSLAND, S.; SHAPIRO, J.; NEHMZOW, U. A self-organising network that grows when required. *Neural Networks*, p. 1041–1058, 2002.

MCHUGH, J. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information System Security*, p. 262–294, 2000.

MOODLE Moodle course management system website. `http://moodle.org/`, 2010.

MURCH, R. *Autonomic computing*. 1 ed. IBM Press, 2004.

NVD National vulnerability database website. `http://nvd.nist.gov/`, 2010.

OWASP *Owasp top 10 2010: The ten most critical web application security risks*. Relatório Técnico, Open Web Application Security Project, 2010.

PANG, R.; ALLMAN, M.; PAXSON, V.; LEE, J. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, p. 29 – 38, 2006.

PEREIRA, C. M. M.; DE MELLO, R. F. Behavioral study of unix commands in a faulty environment. In: *Proceedings of the 8th International Conference on Dependable, Autonomic and Secure Computing*, p. 1–6, 2009.

PYTHON Python v2.7 documentation. `http://docs.python.org/`, 2010.

ROESCH, M.; GREEN, C. *Snort users manual 2.8.4*. Sourcefire, Inc, 2009.

ROUSSEEUW, P. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, p. 53–65, 1987.

SCARFONE, K.; MELL, P. *Guide to intrusion detection and prevention systems (idps)*. National Institute of Standards and Technology, 2007.

SCARFONE, K.; SOUPPAYA, M.; CODY, A.; OREBAUGH, A. *Technical guide to information security testing and assessment*. 1 ed. NIST - National Institute of Standards and Technology, 2008.

SELENIUM Seleniumhq website. `http://seleniumhq.org/`, 2010.

SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal*, v. 27, n. 1, p. 379–423, 1948.

SHASHA, D.; WILSON, M. *Statistics is easy - synthesis lectures on mathematics and statistics*. 1 ed. Morgan and Claypool Publishers, 2008.

SURICATA The open information security foundation website - suricata. `http://www.openinfosecfoundation.org/`, 2010.

TCPDUMP Tcpdump website. `http://www.tcpdump.org/`, 2009.

TWYCROSS, J.; AICKELIN, U. Information fusion in the immune system. *Information Fusion, Volume 11, Issue 1*, p. 35–44, 2010.

TWYCROSS, J. P. *Integrated innate and adaptive artificial immune systems applied to process anomaly detection*. Tese de Doutoramento, University of Nottingham, 2007.

VENDRAMIN, L.; CAMPELLO, R. J. G. B.; HRUSCHKA, E. R. On the comparison of relative clustering validity criteria. In: *SIAM International Conference on Data Mining*, p. 733–744, 2009.

WARRENDER, C.; FORREST, S.; PEARLMUTTER, B. Detecting intrusions using system calls: Alternative data models. In: *IEEE Symposium on security and Privacy*, p. 133–145, 1999.

WASC *The web hacking incidents database annual report*. Relatório Técnico, Web Application Security Consortium, 2007.

XU, R.; WUNSCH, D. *Clustering (ieee press series on computational intelligence)*. 1 ed. Wiley-IEEE Press, 2008.

ZANERO, S.; SAVARESI, S. M. Unsupervised learning techniques for an intrusion detection system. In: *ACM Symposium on Applied Computing - SAC 2004*, p. 412–419, 2004.

Apêndice A: Descrição de ataques a aplicações *Web*

Injeção de Código: Ataque pelo qual um usuário forja requisições a uma aplicação inserindo códigos executáveis no tráfego da rede. Em aplicações *Web* é comum a injeção de códigos da infraestrutura legada (e.g. LDAP ou SQL) ou do sistema operacional onde o servidor é executado (e.g. *Shellscript*). A invasão ocorre quando uma aplicação não valida os dados recebidos do usuário e os envia para processamento sem nenhum tratamento, o que permite a um invasor a execução de código arbitrário no servidor e o acesso a informações restritas.

Cross Site Scripting (XSS): Ataque similar à uma injeção de código, tendo como diferença o fato de que o código não tratado é exibido no navegador do usuário ao invés de ser executado na infraestrutura legada. Isso permite que um invasor execute códigos *Javascript* ou altere a estrutura da página exibida a um usuário, viabilizando roubo de informações pessoais, além de permitir que o usuário seja direcionado a um *website* malicioso.

Cross Site Request Forgery (CSRF): Ataque no qual o conteúdo de uma página é adulterado para que requisições maliciosas sejam enviadas por um usuário válido ao servidor onde a aplicação é executada. Por exemplo, uma página de um sistema de *internet banking* pode ser alterada para que efetue automaticamente uma transferência de fundos no momento em que é exibida. O fato do usuário estar autenticado qualifica essa transferência como válida.

Denial of Service (DoS): Ataques de DoS são ataques que impossibilitam o uso de um sistema por usuários legítimos. Esse tipo de ataque geralmente é efetuado pela sobrecarga do sistema com requisições maliciosas.

Falhas em Autenticação: Diversas falhas nos mecanismos de autenticação podem levar a situações de ataques. Entre essas, pode-se destacar a ausência de criptografia no tráfego de rede, o uso incorreto de identificadores de sessão ou a implementação incorreta de mecanismos de segurança (como a validação de controle de acesso sendo implementada apenas no lado do cliente). A ausência de criptografia permite que todas as informações trafegadas sejam interceptadas por um invasor. O uso incorreto de identificadores de sessão permite que um invasor se passe por um usuário válido. A implementação incorreta de mecanismos de validação permite que um usuário autenticado tenha acesso a funcionalidades que normalmente não deveria ter acesso.

Drive-by Pharming: Ataque que utiliza vulnerabilidades em roteadores de uso doméstico para direcionar o usuário a *websites* maliciosos. O usuário acessa uma página com código malicioso que envia requisições ao roteador. Essas requisições reconfiguram o dispositivo para que as requisições do usuário sejam direcionadas a sites maliciosos.

Falha ao Inibir Automação: Vulnerabilidades que permitem acessos automatizado (i.e. por meio de *scripts*) a páginas *Web* quando esses acessos deveriam ser inibidos.