Método para identificar intrusão por anomalia em *host* com o sistema operacional Windows™ usando o algoritmo de método de *boosting*.

Rogério Winter, Carlos Henrique Quartucci Forster
Praça Marechal Eduardo Gomes, 50 - Vila das Acácias – São José dos Campos – SP – Brasil

***Resumo* — Conexões rápidas com a Internet, bem como a diversidade dos serviços Web oferecidos, facilitam a vida das pessoas e democratizam as informações. Porém, novas formas de crime surgem na intenção de comprometer os computadores e suas redes. Neste contexto, o artigo apresenta um método inovador para identificar intrusão por anomalia em host, através da utilização de processo comparativo, confrontando um sistema normal e outro invadido. Para tanto, foram realizadas medidas de diversos parâmetros do computador em diversas etapas de funcionamento do mesmo: fase inicial, fase de instalação, fase de conexão de rede, fase de operação e fase de infecção. Assim, foi possível traçar um perfil de comportamento associado a cada fase medida. Em conjunto, é utilizada a mineração de dados com a aplicação do algoritmo classificador *ADtree*. Desta forma, foi possível identificar as medidas mais relevantes para a detecção de intrusão, bem como avaliar o método proposto.**

***Palavras-Chave* — intrusion detection, anomaly intrusions, data mining.**

I. INTRODUÇÃO

A Internet começou como uma rede privada de ligação do governo, militares e pesquisadores acadêmicos. A função da segurança computacional é, intuitivamente, limitar o acesso a um sistema de computador. Com um perfeito sistema de segurança, as informações nunca seriam comprometidas porque os usuários não autorizados teriam acesso restrito ao sistema.

Parece estar além da capacidade atual construir um sistema que seja perfeitamente útil e seguro. Em vez disso, a segurança das informações é muitas vezes comprometida por falhas técnicas e por meio de ações do usuário.

A constatação de que não é possível construir um sistema perfeito é importante porque isso mostra uma necessidade maior do que apenas os mecanismos de proteção. É necessário esperar a falha do sistema e estar preparado para lidar com elas.

Também é importante perceber que os recursos de segurança não existem apenas para limitar o acesso a um sistema. O verdadeiro objetivo da execução da segurança é proteger as informações sobre o sistema, que pode ser muito mais valioso do que o próprio sistema ou de acesso aos recursos de computação.

Para [1], a maior parte dos sistemas existentes têm falhas de segurança que os tornam suscetíveis a intrusões, penetrações, e outras formas de abuso; todavia, a correção de todas estas deficiências não é viável por razões técnicas e econômicas.

Uma intrusão é definida como qualquer conjunto de ações que tentam comprometer a integridade, confidencialidade ou disponibilidade de um recurso computacional, aspectos basilares da segurança da informação.

Os sistemas de detecção de intrusão são classificados na literatura como [2]: baseado na fonte de dados e baseados no modelo de intrusões. Os sistemas de detecção baseados na fonte de dados ainda se subdividem em: baseados em *host*, baseados em *multihost* e baseados em rede. Por outro lado, os sistemas de detecção baseados no modelo de intrusões se subdividem em: detecção de intrusão por uso incorreto e detecção de intrusão por anomalia. Todos com as suas características e tecnologias próprias, porém apresentam o inconveniente de detectarem invasões onde, muitas vezes, existe uma variação no tráfego normal, conhecidos como alarmes falsos.

De modo a maximizar e melhorar a eficiência de um sistema de detecção de intrusão, o artigo propõe um método inovador para identificar intrusão por anomalia em um sistema baseado em *host* através da utilização de processo comparativo, no qual um sistema normal é confrontado com outro invadido.

Neste contexto, são construídos os perfis de utilização com o uso da monitoração de diversas características essenciais do sistema. Por conseguinte, na experimentação, é utilizado o processo de mineração de dados [3] com a aplicação do algoritmo de classificação *ADtree*[4] onde as características anteriormente selecionadas contribuíram decisivamente na detecção de intrusão por anomalia.

O texto está organizado da seguinte forma: a introdução que dá uma visão geral e salienta a finalidade do trabalho; na seção materiais e métodos é apresentada uma seleção dos softwares utilizados e a forma como os dados são obtidos. Ainda, é exposto o método utilizado como base de obtenção de parâmetros do computador. Em seguida, discorre-se sobre a análise preliminar dos dados com o processo de pré-processamento e transformação dos mesmos. Por fim, é demonstrada com uma experimentação a aplicação do classificador *ADtree* no conjunto de dados bem como a sua

eficiência para o processo de detecção de intrusão por anomalia.

II. MATERIAIS E MÉTODOS

2.1 Materiais

Os seguintes softwares foram selecionados como coadjuvantes no processo de detecção por anomalia em *host*. A escolha feita com critério, tais como: facilidade de operação, possibilidade de tratar dados multivariados, utilização em diversas versões do Windows e fundamentada em parâmetros técnicos necessários a execução dos trabalhos. O software Process Monitor [5] utilizado na monitoração do sistema durante o processo de medida dos parâmetros de máquina; software WEKA [6] [7] API Java utilizada para análise dos dados referentes à monitoração; MySQL 5.1.42[8] banco de dados relacional necessário a organização e consultas dos dados e o Visulab[9] *add-in* para o Microsoft Excel é um pacote de software experimental para a visualização comparativa dos dados multivariados.

2.2 Métodos

Conforme, anteriormente, tratado, os sistemas de detecção de intrusão baseados em fonte de dados classificam-se em: baseado em *host*, os dados de auditoria, a partir de um único *host*, são usados para detectar intrusões; baseado em *multihost*, os dados de auditoria de vários hosts são usados para detectar intrusões e baseado em rede, os dados de tráfego de rede, juntamente com dados de auditoria de um ou mais hosts, são usados para detectar intrusões.

Por outro lado, os sistemas de detecção de intrusão baseados no modelo de intrusões classificam-se em: intrusão por uso incorreto (*misuse intrusions*) e intrusão por anomalia (*anomaly intrusions*). A detecção de intrusão por uso incorreto revela intrusões, olhando para a atividade que corresponde a técnicas de intrusão conhecidas (*signatures*) ou vulnerabilidades do sistema, todavia a principal limitação deste enfoque é que as ferramentas não conseguem detectar novas formas de código malicioso ou ataques. O sistema de detecção de intrusão por anomalia baseia-se em observações dos desvios de uso normal do sistema computacional. Neste caso, é construído um perfil de utilização do sistema monitorado com a avaliação de recursos do tipo: consumo de CPU, uso de memória, quantidade de operações realizadas em um determinado tempo e, com isso, pode-se apreciar o quão discrepante são determinadas operações em relação ao perfil. Neste contexto, o trabalho edifica-se em apresentar o método de realização medidas dos parâmetros necessários a detecção de intrusão por anomalia em *host.*

Obedecendo as etapas de um processo de mineração de dados proposto por [3] [15] [16], as fases do trabalho foram: obtenção dos dados, análise preliminar e seleção, pré-processamento e transformação dos dados, aplicação de algoritmos de mineração de dados e análise dos resultados.

2.2.1 Obtenção dos dados e parâmetros medidos

As informações obtidas através da monitoração do computador com o software Process Monitor permitiram a constituição de uma base de dados de *logs* com os diversos parâmetros de operação. Diferentemente de outros softwares que são restritos a determinadas versões do sistema operacional Windows™, o Process Monitor é compatível com o Windows XP SP2 e Windows Server 2003 SP1 e as versões mais atuais do sistema operacional.

Os parâmetros medidos foram os seguintes: Time of Day, Relative Time, Duration, Process Name, PID, Event Class, Operation, Result, Image Path, User, TID, Path, Parent ID, Sequence.

* Time of Day - o parâmetro marca a hora do dia na qual foi executada determinada operação no sistema.

* Relative Time - o tempo de operação em relação ao início do processo do Monitor ou a última vez que o processo de visualização do monitor foi cancelado.

* Duration - duração de uma operação que foi concluída.

* Process Name - o nome do processo em que ocorreu o evento.

* PID - o Process ID (PID) que executou uma determinada operação.

* Event Class - a classe (File, Registry, Process) do evento.

* Operation - a específica operação de um evento. (Exemplo Read, RegQueryValue, etc.).

* Result - o código de status de uma operação concluída.

* Image Path - o caminho completo da imagem em que o processo está rodando.

* User- o nome da conta do usuário na qual o processo que executou uma operação.

* TID - O Thread ID (TID) que executa uma operação.

* Path - o caminho de um recurso que um evento referencia.

* Parent ID - a referência ao processo pai que executou uma operação.

* Sequence - posição relativa da operação no que diz respeito a todos os eventos incluídos no filtro.

"Time of Day","Relative Time","Duration","Process Name","PID","Event Class","Operation","Result","Image Path","User","TID","Path","Parent PID","Sequence"
"14:15:49.4586206","00:00:00.0008372","0.0000196","lsass.exe","736","Registry","RegOpenKey","SUCCESS","C:\WINDOWS\system32\lsass.exe","AUTORIDADE NT\SYSTEM","828","HKLM\SECURITY\Policy","680","n/a"
"14:15:49.4586480","00:00:00.0008646","0.0000101","lsass.exe","736","Registry","RegOpenKey","SUCCESS","C:\WINDOWS\system32\lsass.exe","AUTORIDADE NT\SYSTEM","828","HKLM\SECURITY\Policy\SecDesc","680","n/a"
"14:15:49.4586676","00:00:00.0008842","0.0000055","lsass.exe","736","Registry","RegQueryValue","BUFFER OVERFLOW","C:\WINDOWS\system32\lsass.exe","AUTORIDADE NT\SYSTEM","828","HKLM\SECURITY\Policy\SecDesc\(Default)","680","n/a"
"14:15:49.4586935","00:00:00.0009101","0.0000031","lsass.exe","736","Registry","RegCloseKey","SUCCESS","C:\WINDOWS\system32\lsass.exe","AUTORIDADE NT\SYSTEM","828","HKLM\SECURITY\Policy\SecDesc","680","n/a"
"14:15:49.4587025","00:00:00.0009191","0.0000064","lsass.exe","736","Registry","RegOpenKey","SUCCESS","C:\WINDOWS\system32\lsass.exe","AUTORIDADE NT\SYSTEM","828","HKLM\SECURITY\Policy\SecDesc","680","n/a"

Fig. 1. Extrato de um arquivo, no formato *csv*, resultante da monitoração.

2.2.2 Método de captura e organização dos dados

Foi criado um método que permitisse estudar os fenômenos que distinguem o comportamento de um computador em atividade normal de outro com problemas com infecção ou intrusão [12]. Para efetivação das medidas, o método foi concebido em fases onde foram estabelecidos cinco momentos de utilização de um computador: fase

inicial; fase de instalação; fase de conexão de rede; fase de operação; e fase de infecção do sistema. Esta última foi subdividida em outras seis subfases. Em todas as fases e subfases os resultados das medidas, os *logs*, foram salvos em arquivos com o formato c*omma-separated values* (*csv*), em virtude da facilidade de transportá-los para o banco de dados. O tempo de medição em cada fase é de cinco minutos, exceto para a fase de instalação que, em razão da velocidade do computador, os tempos variaram entre um minuto e cinco minutos.

* Fase inicial - é caracterizada por um processo de pós-instalação sem a aplicação de pacotes de atualização ou de segurança.

* Fase de instalação - foram realizadas instalações de software consideradas úteis para a utilização do computador. No presente trabalho foram instalados os seguintes aplicativos e versões:

TABELA I. APLICATIVOS UTILIZADOS NOS TESTES.

Aplicativo	Tipo	Versão
Br Office	Escritório	3.1.1
Adobe Acrobat Reader	Leitor de arquivos no formato PDF	9.3 3.9.1
Winrar	Compactador de arquivos	
Java Runtime Environment (JRE)	Máquina virtual Java	6 Update 20
Firefox	Navegador de páginas Web	3.5.7
Kaspersky 2010	Antivírus	9.0.0.736

* Fase de conexão de rede - o computador foi conectado à rede local e na Internet e foram realizados comandos de rede para ajustes dos parâmetros de conexão.

* Fase de operação - caracterizada pela operação propriamente dita do sistema, onde os softwares instalados foram utilizados nas tarefas específicas.

* Fase de infecção - dividida em seis subfases para avaliar os diversos tipos de infecção e apreciar os tipos de operação realizados pelo computador. A máquina de teste foi infectada com 01 *rootkit*, 04 tipos de vírus, 01 *trojan* e 01 *keylogger*.

A fim de constituir uma base de dados criada no MySQL, foi desenvolvido um aplicativo em linguagem *shell script*, no Linux, que lê os diferentes arquivos no formato *csv*, manipula determinados campos e armazena as informações.

2.2.3 Análise preliminar e seleção

A análise preliminar teve como objetivo identificar a necessidade de tratamentos ou outros processamentos nos dados existentes e selecionar períodos de amostras de dados que apresentem comportamentos relevantes relacionados à ocorrência de um evento específico.

Com a inserção dos dados no MySQL foi possível, por meio de consultas SQL, extrair informações e gerar gráficos mostrando a variabilidade dos valores na linha do tempo. Nesta fase utilizou-se a ferramenta Excel e o Visulab a fim de traçar gráficos que permitissem identificar aspectos relevantes nos eventos.

As consultas SQL selecionaram os dados de minuto em minuto.

2.2.4 Pré-processamento e transformação dos dados

Através da análise visual dos dados foi possível selecionar os atributos mais relevantes ou os que apresentavam um maior impacto na operação do sistema. Também foram tratadas as dependências e correlação linear entre as variáveis, onde a visualização através matriz de espalhamento permitiu a identificação. A partir de então os dados foram selecionados e, ainda no formato *csv*, foram carregados no software Weka. Todavia, dentro do software Weka os dados foram avaliados pelo algoritmo *InfoGainAttributeEval*[10], que mede o ganho de informação de acordo com a classe do atributo.

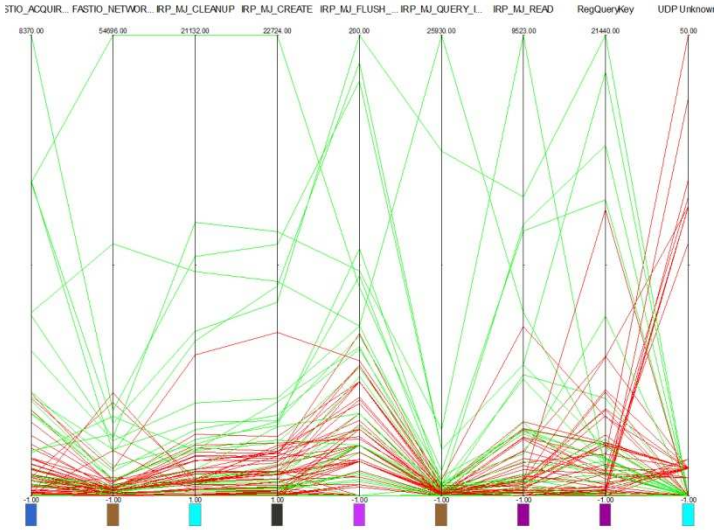

Fig.2. Gráfico de coordenadas paralelas – a cor vermelha representa a fase de infecção.

2.2.5 Aplicação de algoritmos de mineração de dados

O algoritmo de mineração de dados utilizado para o presente trabalho foi o classificador *Alternating Decision Tree (ADtree*) do software Weka. Esta versão atualmente suporta apenas os problemas de duas classes. O número de iterações de *boosting* foi ajustado manualmente para atender a complexidade e a precisão do conjunto de dados. Este algoritmo recebeu uma otimização na indução das árvores, e métodos de busca heurística foram introduzidas para acelerar a aprendizagem. Os dados foram submetidos à opção de teste do Weka [6] [7] *cross-validation* ou validação cruzada.

Na modalidade validação cruzada e a fim de obter resultados estatisticamente mais significativos, o número padrão de iterações é 10. Neste caso, a validação cruzada é realizada 10 vezes em cada dobra, e isto significa que são realizadas 100 chamadas de um classificador nos dados para treinamento e novamente cruzadas com o restante dos dados de teste.

ADTree[4][11] é uma generalização das árvores de decisão que utilizam o método de *boosting* para aprendizagem de máquina. O *boosting* é exemplo de *ensemble methods*, ou métodos que usam uma combinação de modelos como forma de melhorar a precisão da classificação ou predição. A árvore de decisão é fácil de entender e visualizar, além de ser uma abordagem clássica utilizada por muitas das ciências. Porém, ela apresenta um único caminho até a decisão, não existe nenhum nível de confiança e muitos critérios utilizados podem levar a um crescimento da árvore. Por outro lado, o *boosting* é um método de reforço que

considera que *T* hipóteses frágeis, com seus pesos, somadas podem resultar em uma hipótese mais consistente.

Pseudocódigo do Algoritmo *ADtree* proposto em [4]

Inicialize: o conjunto R para consistir única regra base, cuja precondição e condição são ambos T e cuja primeiro valor de predição é

o cálculo de α_t=2 $\ln\frac{w+(T)}{w-(T)}$

Faça t=1, 2,..., T

1. Para cada precondição base $c_1 \in P_t$ e cada condição $c_2 \in$ C calcule:

$$Z_t(c_1, c_2) = 2\sqrt{W+(c_1 \wedge c_2)W-(c_1 \wedge c_2)} + \sqrt{W+(c_{1\wedge}\neg c_2)W-(c_{1\wedge\neg}c_2)} + W(\neg c_2).$$

2. Selecione c_1, c_2 os quais minimizam $Z_t(c_1, c_2)$ e conjunto R_{t+1} para ser R_t com a adição da regra r_t cuja precondição é c_1e c_2é condição e os dois valores de precondição são:

$$a = \frac{1}{2} ln \frac{W+(c_1 \wedge c_2)}{W-(c_1 \wedge c_2)}, \quad b = \frac{1}{2} ln \frac{W+(c_1 \wedge \neg c_2)}{W-(c_1 \wedge \neg c_2)}.$$

3. Conjunto P_{t+1} para ser P_t com adição de $c_{1\wedge}c_2$ e $c_{1\wedge}\neg c_2$.
4. Atualize os pesos de cada exemplo de treinamento de acordo com a equação:

$$w_{i,t+1} = w_i, t^{e^{r_{t(x_i)}y_i}}$$

No caso de $r(_{x_i}) = 0$ o peso não é atualizado.

Saída do algoritmo: a regra de classificação que é o sinal da soma de todas as regras base em R_{T+1}:

$$classe(x) = sinal \sum_{t=1}^{T} r_t(x)$$

A maioria dos algoritmos inicia com a memorização dos dados, ao invés de aprender padrões [11]. Todavia, o *ADtree* inicia procurando a melhor constante de predição de todo o conjunto de dados. O valor inicial de predição é colocado como a raiz da árvore e esta irá aumentando de forma iterativa com a adição de novas regras base ao longo do tempo. A adição de novas regras corresponde a uma subárvore com um nó de decisão, como sua raiz, e dois nós de previsão como as folhas. Esta subárvore é adicionada como um filho de um nó de predição que pode ou não ser um nó folha. O critério de parada do algoritmo é feito pela validação cruzada que, no nosso caso, o melhor resultado foi alcançado com o valor 10. Além disso, constatou-se por ocasião da realização de testes, com quatro dobras na validação cruzada o resultado da classificação permaneceu inalterado, assim o algoritmo ficou isento ao problema de *overfitting* de modelo.

Um dos objetivos da aprendizagem de máquina é a margem, que é uma medida de quão correto está um exemplo. Ou seja, é a confiança que é medido pela diferença entre a probabilidade estimada da classe verdade e a classe mais provável que não previu a classe verdadeira. Se todas as hipóteses obtiverem um exemplo correto, provavelmente vamos obter um exemplo correto no futuro, porém se uma em cada 1000 hipóteses obtiver um exemplo correto, então, provavelmente, teremos erros no futuro. Desta forma, *boosting* nos dá uma boa margem.

No trabalho especificamente, os dados coletados em fases anteriores foram submetidos ao classificador *ADtree* apresentando os resultados abaixo:

=== Summary ===

Correctly Classified Instances	101	98.0583%
Incorrectly Classified Instances	2	1.9417%
Kappa statistic	0.9554	
Mean absolute error	0.0884	
Root mean squared error	0.1691	
Relative absolute error	19.9271%	
Root relative squared error	35.9427%	
Coverage of cases (0.95 level)	100.0000%	
Mean rel. region size (0.95 level)	72.8155%	
Total Number of Instances	103	

Fig. 3. Sumário do classificador ADtree – *cross validation*

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.9410	0.0000	1.0000	0.9410	0.9700	0.9900	normal
	1.0000	0.0590	0.9720	1.0000	0.9860	0.9900	intrusao
Weighted Avg.	0.9810	0.0390	0.9810	0.9810	0.9800	0.9900	

Fig. 4. Detalhes de precisão do classificador ADtree – *cross validation*

Confusion Matrix

a	b	<-- classified as
32	2	a = normal
0	69	b = intrusao

Fig. 5. Matriz de confusão do classificador *ADtree*

2.2.6 Análise dos resultados

Como resultado da manipulação dos dados obtidos na fase de obtenção dos dados, foram submetidos à análise com o algoritmo *ADtree* 103 instâncias amostradas cada uma com 46 atributos pertinentes a parâmetros de utilização do computador de teste.

Iniciamos pela análise do parâmetro *Kappa statistic* ou estatística *Kappa* (Fig. 3), que mede a concordância da predição com a classe verdadeira. Neste caso, observamos que o classificador em tela, na modalidade validação cruzada, apresentou uma concordância do modelo acima dos 95%. Ainda podemos observar que na mesma modalidade a taxa de instâncias classificadas corretamente ultrapassa os 98%

contra menos que 2% classificadas erradamente na presente amostra.

Quanto à precisão global do classificador (Fig 4.) ficamos com uma taxa acima dos 98% sendo que Real Positivo(TP) para a classe intrusão é de 100% e Falso Positivo (FP) para a mesma classe fica em torno dos 6%. Em consequência destes resultados, a área da curva ROC (*Receiver Operating Characteristic*) fica em torno dos 99% sendo considerado um excelente resultado. Na matriz de confusão (Fig. 5) visualizamos que a classificação permitiu identificar todas as intrusões no sistema.

Pode-se prever que o desempenho do classificador em termos de acertos aumenta na medida em que as instâncias aumentam. Contudo, o classificador *ADtree* demonstrou uma grande eficiência para classificar os eventos em normal e intrusão quando usamos o valor 10 para validação cruzada e 10 para *boosting iteration*[16].

2.2.7 Interpretação da árvore gerada pelo ADtree

A fim de interpretar a árvore gerada pelo ADtree, alguns esclarecimento são necessários ao entendimento do processo. Na apresentação da árvore de decisão, o algoritmo identificou alguns parâmetros mais relevantes para distinção entre atividade "normal" e "intrusão". Em seguida, estes parâmetros são conceituados.

Fast I/O [13] é um tipo de operação projetada especificamente para uma rápida sincronização I/O no cache de arquivos. Em operações *Fast I/O*, os dados são transferidos diretamente entre buffers de usuário e o sistema de cache, ignorando o sistema de arquivos e o controlador de armazenamento pilha.

I/O request packets (IRPs) [13] são estruturas do modo *kernel* usadas pelo *Windows Driver Model* (WDM) e drivers de dispositivo do Windows NT para se comunicarem uns com os outros e com o sistema operacional. Eles são estruturas de dados que descrevem as solicitações de I/O.

User Datagram Protocol (UDP) [14] e *Transfer Control Protocol* (TCP) são os dois protocolos da camada de transporte do TCP/IP nas versões 4 e 6. O UDP é um protocolo orientado a datagrama; sem um mecanismo de confiança que garanta a entrega do datagrama ao destinatário e sem conexão, pois não necessita manter um relacionamento entre cliente e servidor. Alguns protocolos que utilizam o UDP são: *Dynamic Host Control Protocol* (DHCP), *Domain Name Service* (DNS), *Trivial File Transfer Protocol* (TFTP) e *Simple Network Management Protocol* (SNMP).

O Windows [17] armazena dados de configuração no registro. O registro é um banco de dados hierárquico, que pode ser descrito como um repositório central de informações de configuração, na terminologia da Microsoft. Isso permite que as configurações sejam referenciadas utilizando caminhos semelhantes aos caminhos de arquivo no Windows Explorer. Diversos tipos de operações podem ser executadas no registro: consulta de parâmetros, exclusão de parâmetros, atualização de parâmetros e outros.

Para demonstrar a interpretação, é considerada a árvore apresentada na figura 6. Esta árvore é o resultado da execução do algoritmo de aprendizado por 10 iterações no conjunto de dados resultante da seção 2.2.4. Este conjunto de dados distinguiu entre a operação "normal" e "intrusão" do computador. No mapeamento, a classificação positiva corresponde à intrusão e a negativa para operação normal.

Esta árvore alternante consiste de sete nós de decisão. Sustenta-se o fato de que a descrição da árvore alternante é muito menor, é mais fácil de ler e de interpretar. Não obstante, a estrutura da árvore alternante é mais complexa do que uma árvore de decisão, porém sua interpretação é mais fácil quando comparado a uma árvore de decisão com o mesmo número de nós de decisão.

O argumento principal para a interpretação da árvore alternante repousa no fato de que a contribuição de cada nó de decisão pode ser entendida de forma isolada. Somando essas contribuições gera a previsão da classificação. Considerando o exemplo da figura isoladamente, cada um dos nós de decisão, observamos que a operação FASTIO_READ>=319.5 e UDP Unknown<0.5 são indícios de "intrusão" no sistema. Isso é indicado pelo fato de que ambos geram contribuições negativas para a soma da previsão. Assim, é facilmente possível interpretar o significado de todos os nós na árvore de decisão de forma semelhante.

Após obter o significado de cada nó de decisão isoladamente, podemos analisar as interações dos nós. Os nodos de decisão paralela, como os quatro nós no terceiro nível representa pouca ou nenhuma interação. Em outras palavras, o fato de que a operação IRP_MJ_CLOSE>=0.5 aumenta a probabilidade de que o sistema seja considerado "infectado", independentemente do número operações UDP Unknown ou da operação RegQueryKey.

A raiz (*root*) da árvore está associada com a contribuição fixa incondicional. Este número positivo indica que, segundo esse conjunto de treinamento, há mais intrusão do que normal. Isto significa que antes de testar o valor de alguma característica deve-se prever que o sistema pode apresentar algum tipo de intrusão, porém com baixo nível de confiança.

Todas as contribuições são somadas para dar a previsão final e essa previsão é o limiar da classificação. Isto significa que, se testarmos as condições em série indicadas na árvore elas acumulam evidências a favor ou contra o estado "normal". Se continuarmos em algum ponto intermediário durante este processo, temos uma soma cujo valor absoluto é grande e a contribuição total dos nós não testados é pequena, então nós não precisamos continuar nossa computação, pois o sinal atual da soma é improvável que mude.

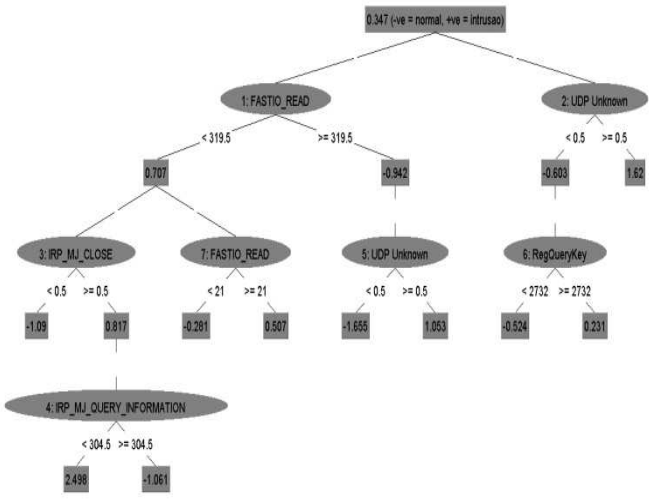

Fig. 6. Árvore de decisão gerada pelo algoritmo classificador *ADtree*

Portanto, pode-se inferir que o valor absoluto da soma pode ser considerado como uma medida de confiança na classificação e assim conhecida como margem [18] da classificação.

O fato de cada nó de decisão ter uma influência limitada na previsão aumenta a robustez da representação.

III. CONCLUSÃO

Com a detecção de intrusão, é mais difícil procurar por atividade não autorizada em um sistema, pois podem existir tantos alarmes que os mesmos se assemelham a atividade legítima. Na tentativa de não perder algo que é potencialmente malicioso, muitos dos atuais sistemas de intrusão emitem alarmes falsos, todavia substancialmente redução da sua eficácia.

Ao longo deste artigo foi analisado o efeito da utilização dos parâmetros de funcionamento do computador na detecção de intrusão por anomalia como forma de minimizar a quantidade de alarmes falsos.

É proposto um método inovador para identificar intrusão por anomalia através da utilização de processo comparativo, no qual um sistema normal é confrontado com outro invadido. Para efetivação das medidas, o método foi concebido em fases onde foram estabelecidos cinco momentos de utilização de um computador: fase inicial; fase de instalação; fase de conexão de rede; fase de operação; e fase de infecção do sistema. Esta última foi subdividida em outras seis subfases. Assim, foi possível construir os perfis de utilização, além de definir, com o uso da monitoração, as características mais relevantes do sistema.

Com a seleção dos parâmetros do computador, a experimentação utilizou o processo de mineração de dados com a aplicação do algoritmo de classificação *ADtree* onde as características anteriormente selecionadas contribuíram decisivamente na detecção de intrusão por anomalia.

O algoritmo *ADtree* é um exemplo de *ensembles methods*, pois usa a combinação de modelos. Pesquisas mostram que os *ensembles methods*, muitas vezes melhoram o desempenho em classificadores único. O *ADtree* faz uso do método *boosting*, o qual produz um conjunto de modelos ponderados por iterativa aprendizagem de um conjunto de dados, avaliando-os e realizando uma nova ponderação com base no desempenho do modelo. Assim, o método usa o conjunto de modelos de forma a prever a classe com maior peso.

O trabalho como um todo conseguiu apresentar resultados excelentes na detecção de intrusão, em parte pela inserção de um método que propiciou, de forma controlada, a monitoração do sistema computacional. Por outro lado, a utilização do algoritmo *ADtree* fortaleceu a tese de reduzir a quantidade de falsos alarmes no sistema de detecção de intrusão. Ressalte-se que árvore de decisão gerada pelo algoritmo *ADtree* (Fig. 6) segue o princípio da simplicidade na detecção de intrusão por anomalia, pois com a monitoração de 5 parâmetros de funcionamento da máquina e a realização testes decisórios cria-se um sistema de alarme com alta taxa de detecção de intrusão.

Por fim, o método de medida proposto no artigo agregado à utilização do algoritmo classificador *ADtree* revelaram eficiência e precisão na detecção de intrusão, provando que o seu uso em prol da detecção de intrusão por anomalia em host e na segurança da informação demonstrou serem perfeitamente adequados como mecanismo de proteção.

Rogério Winter, winter@ita.br, Carlos Henrique Quartucci Forster, forster@ ita.br, Tel +55-12-39475981,+55-12-39475981.

REFERÊNCIAS

[1] DENNING, Dorothy E. An Intrusion-Detection Model. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-13, NO. 2, 222-232. 1987.

[2] CROSBIE, Mark. PRICE, Katherine. Et al. Intrusion Detection Pages. Disponível em: <http://www.cerias.purdue.edu/about/history/coast_resources/idcontent/policy.html>. Acesso em: 04 fev. 2009.

[3] FAYYAD, Usama. PIATETSKY-SHAPIRO, Gregory. SMYTH, Padhraic. From Data Mining to Knowledge Discovery in Databases.PP 1-34. American Association for Artificial Intelligence. Menlo Park, CA, USA.1996.

[4] FREUND, Yoav. MASON, Llew. The alternating decision tree learning algorithm. Proceedings of the Sixteenth International Conference on Machine Learning.PP 124 - 133. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. 1999.

[5] RUSSINOVICH, Mark. COGSWELL, Bryce. Process Monitor. Disponível em: <http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx>. Acesso em: 14 abr. 2010.

[6] FRANK, Eibe. HALL, Mark. Et al. WEKA Version 3-7-1. Disponível em:<http://www.cs.waikato.ac.nz/~ml/weka/>. Acesso em: 10 mar. 2010.

[7] WITTEN Ian H. FRANK, Eibe Data Mining Practical Machine Learning Tools and Techniques. Second Edition. Elsevier. USA. 2005.

[8] MYSQL versão 5.1.42. Disponível em:< http://dev.mysql.com/downloads/>. Acesso em: 12 jan. 2010.

[9] VISULAB - Interactive data visualisation in Microsoft Excel. Disponível em: <http://www.inf.ethz.ch/personal/hinterbe/Visulab/>. Acesso em: 02 abr 2010.

[10] BOUCKAERT, Remco R. FRANK, Eibe. Et al. WEKA Manual for Version 3-7-1. Disponível em: <http://www.cs.waikato.ac.nz/~ml/weka/> Acesso em: 10 mar. 2010.

[11] JBOOST. What are ADTrees? Disponível em: <http://jboost.sourceforge.net/presentations/BoostingLightIntro.pdf>. Acesso em: 15 mai. 2010.

[12] VIELER, Ric. Professional Rootkits Wiley Publishing, Inc. Indianapolis. 2007

[13] RUSSINOVICH, Mark. SOLOMON, David A. IONESCU, Alex.Windows Internals – Covering Windows Server 2008 and Windows Vista – Fifth Edition. Microft. USA. 2009.

[14] STEVENS, W. Richard. TCP/IP Illustrated: the protocols. Volume 1. Addison Wesley Longman, Inc. Massachusetts.USA. 1997.

[15] HAN, Jiawei. KAMBER, Micheline. Data Mining: Concepts and Techniques. Second Edition. Elsevier. USA. 2006.

[16] TAN, Pang-Ning. STEINBACH, Michael. KUMAR, Vipin. Introdução ao DATAMINING Mineração de dados. Rio de Janeiro. Editora Ciência Moderna Ltda.2009.

[17] HONEYCUTT, Jerry. Windows Registry Guide. Microsoft Press. Washington .USA. 2005.

[18] SCHAPIRE, Robert E. FREUND , Yoav .BARTLETT, Peter . Lee,Wee Sun. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics October. 1998.