Insider Threat - Analisando Comportamento de Usuários para Detecção de Ameaças Internas

Abraão Vitor L. Dantas[1], **Henrique David de Medeiros**[1]

[1] Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brasil

abraaovld@gmail.com, henriquemed101@gmail.com

1. Introdução

Atualmente com a revolução das tecnologias de informação, a preocupação com segurança da informação por parte das empresas tem aumentado consideravelmente. Em uma pesquisa realizada no ano de 2016 pela empresa PWC-Brasil, no Brasil houve um aumento de 274% nos incidentes de segurança da informação. E nessa estatística não se contabiliza os casos em que as empresas não relatam ataques, que se imagina serem muitos.

Apesar da existência de ataques externos, um dos que mais preocupa atualmente são os ataques internos (denominado ”insider attacker ”ou ”insider threat”) [Legg et al. 2015], [Cole 2015]. Tais ameaças internas são caracterizadas por usuários que fazem parte de empresas e que por terem certas permissões de acesso as exploram para comprometer a segurança da organização e de seus sistemas [Silowash et al. 2012], [Cappelli et al. 2012].

O nosso trabalho visa então ser capaz de detectar ataques internos através da análise de logs de atividade dos usuários em busca de anomalias [Legg et al. 2015]. Com esta análise cria-se uma árvore capaz de agrupar todas as atividades dos usuários e assim construir um perfil deste, permitindo fazer comparações e identificar comportamentos anômalos.

2. Abordagem de solução

O projeto foi desenvolvido para realizar a leitura de arquivos de log de usuários, dentre eles uma pasta **LDAP** com arquivos, com extensão csv, de informações dos usuários em períodos diferentes, **device.csv** com as informações dos dispositivos que foram conectados, **http.csv** com urls acessadas pelos usuários e **logon.csv**, com os dispositivos em que foram realizados logon e logoff.

O sistema foi desenvolvido em linguagem de programação Java, utilizando as IDEs (Integrated Development Environment) Netbeans e Eclipse para a escrita do código fonte. Foram utilizadas bibliotecas externas como o JFreeChart, para a geração do gráfico visualmente; e da Commons Math para realizar o cálculo do *IQR(interquartile range)*.

O projeto completo é feito com base nos diagramas de classe presentes nas figuras 1 e 2. Existe uma classe chamada **Node** cuja as demais classes que compõem a estrutura de dados herdam, contendo apenas dois atributos, é uma classe bem simples cujo o construtor obriga que todos os filhos tenham um id, este é único entre objetos das classes User e Device.

Para a execução do projeto temos a classe **Tree_insiders** que será a classe responsável por manter a estrutura da árvore na memória, sendo responsável pela construção

e leitura da árvore. Já a classe **FileReader** é a responsável por fazer a leitura dos arquivos de log e através de um objeto da **Tree_insiders** chamar os métodos de criação. Finalmente a tarefa de construir os histogramas de cada usuário e fazer a análise destes fica a cargo da classe **Analyzer** e seus métodos.

Durante o desenvolvimento foram adotados padrões de projetos no que diz às estruturas de dados lineares que servem de auxílio, bem como na distribuição de pacotes de código fonte e sua organização.

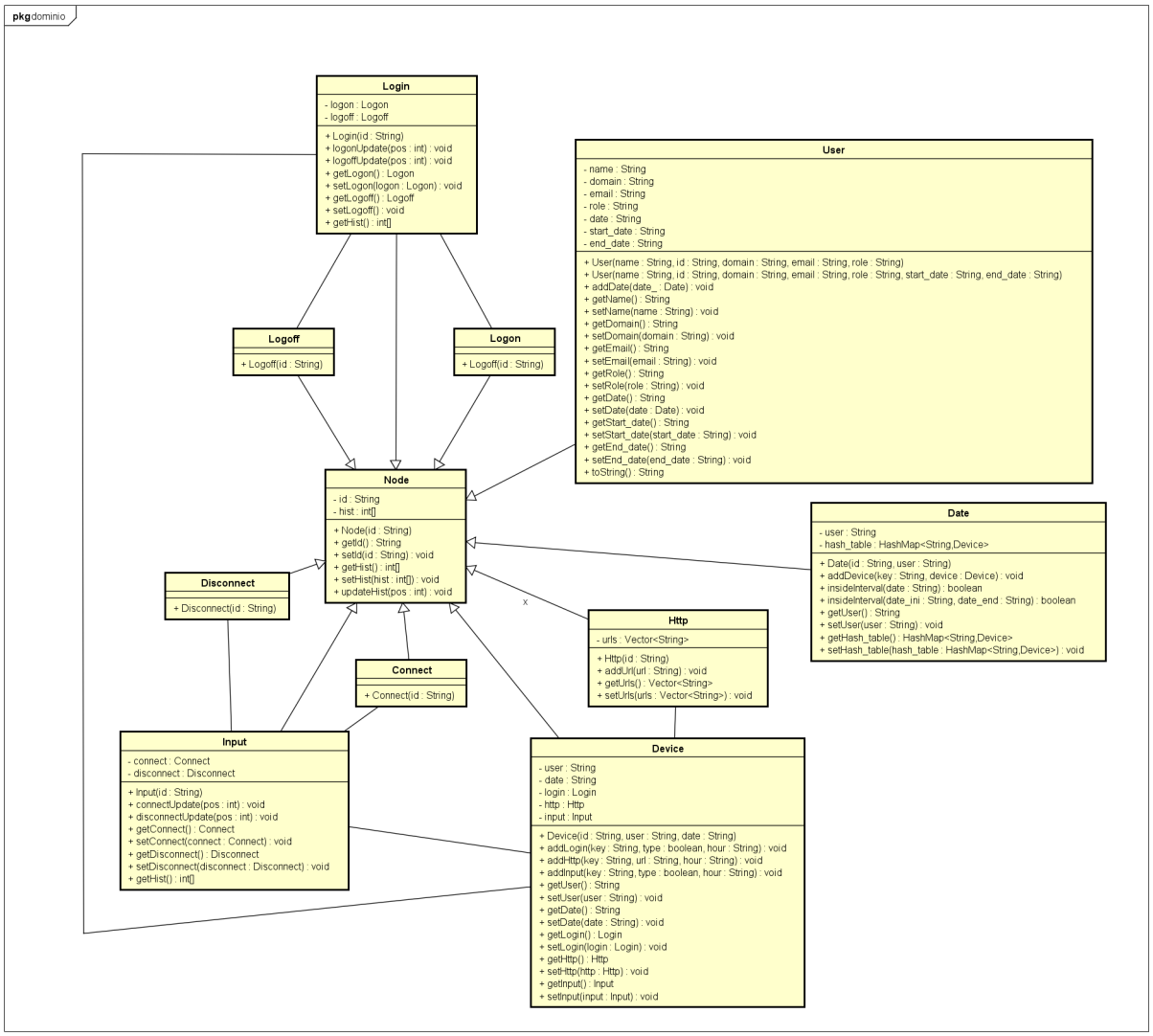

Figura 1. Diagrama de Classes 1

3. Estrutura de dados e Algoritmos Utilizados

Para a criação e análise dos perfis de usuário a estratégia abordada foi a de criar uma árvore genérica combinada a outra estrutura linear, no caso uma tabela de dispersão cujo o motivo da escolha será exposto mais a frente. O formato base da estrutura pode ser visualizado na figura 3.

Como é possível observar na figura 3 a tabela de dispersão aparece no 1º e 3º nível da árvore, isso se dar pelo fato que no primeiro nível podem existir *n* usuários, assim como

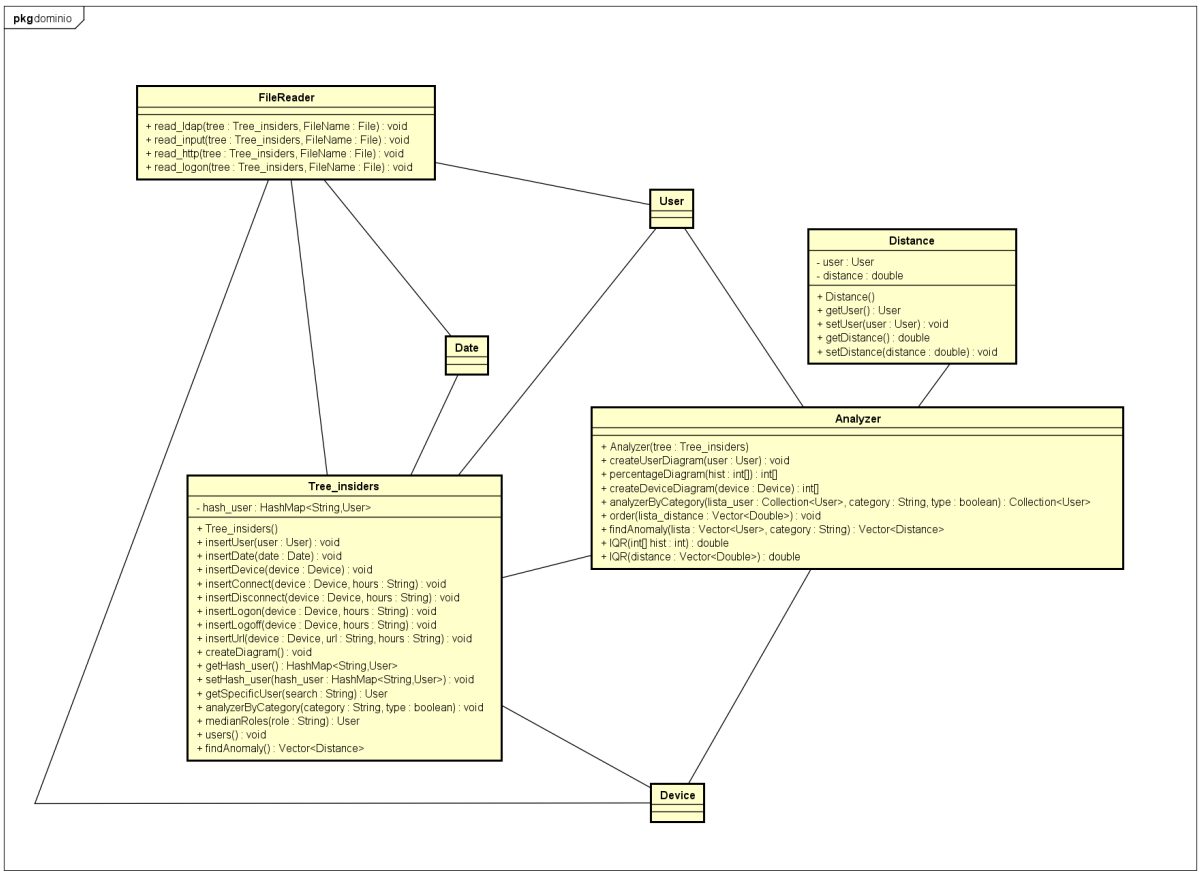

Figura 2. Diagrama de Classes 2

cada usuário pode ter usado *n* dispositivos. Como os outros nós são estáticos não se faria necessário usar uma estrutura auxilixar.

O motivo de escolha da tabela de dispersão para o projeto foi devido a complexidade de inserção e busca de $\mathcal{O}(1)$ que a estrutura possui, levando em consideração que cada usuário, bem como dispositivo, possui um *id* único, o número de colisões da tabela seria bem reduzido tornando o seu uso viável. Deste modo o uso desta em combinação com uma árvore genérica permitiu inserções e leituras de maneira muito eficiente dos dados presentes nos arquivos de log.

3.1. Detectando Anomalias

Uma vez criada a árvore contendo os perfis dos usuários é necessário que se passa uma análise em busca de possíveis anomalias no comportamento destes. Considerando que cada usuário mantém um histograma de 24 posições onde cada uma representa uma hora do dia e contém o número de acões naquela hora, podemos então organizar o usuário por sua função dentro da empresa e traçar uma média para que seja calculada a distância entre o perfil de um usuário para o perfil médio, através da equação de distância Euclidiana.

$$D(h_A, h_M) = \sqrt{\sum_{n=0}^{23} (h_A[n] - h_M[n])^2} \tag{1}$$

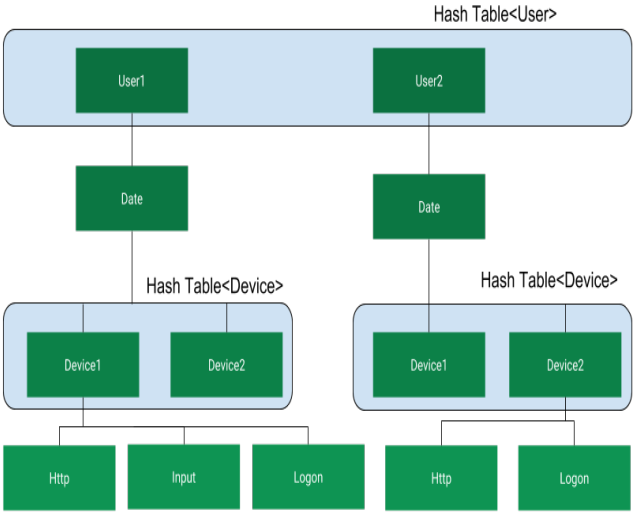

Figura 3. Estrutura de Dados

Onde h_A é o histograma de um usuário e h_M é o histograma médio de um papel. Uma vez que se possui todas as distâncias de todos os usuários que desempenham o mesmo papel, utilizamos então um algoritmo de IQR (Variância Interquartil) com estas distâncias para encontrar o valor de variância, chegando a uma segunda equação para determinar que trata-se de uma anomalia:

$$D(h_A, h_M) \geq 1.5 * IQR \tag{2}$$

Ou seja, para que o perfil de um usuário seja tratado como uma anomalia é necessário que a distância entre seu perfil para o da média seja $1.5x$ maior que o valor de IQR.

4. Conclusão

Com a escolha de uma árvore genérica utilizando tabelas de dispersão para auxiliar, o desempenho na criação e leitura dos perfis mostrou-se bastante satisfatório. Além disso cuidados na hora da abertura do arquivo foram cruciais para o funcionamento correto da aplicação sem que houvesse falta de mémoria. Com um tema bem atual, como é o caso de segurança, foi possível exercitar todos os conceitos aprendidos na disciplina de Linguagem de Programação 2 e ir até um pouco mais além, já que foram utilizadas bibliotecas externas para enriquecer o software, além disso foi possível aplicar conhecimentos oriundos de outras disciplinas como Estruturas de Dados Básicas 1 e Estruturas de Dados Básicas 2.

Referências

Cappelli, D. M., Moore, A. P., and Trzeciak, R. F. (2012). *The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud)*. Addison-Wesley.

Cole, E. (2015). Insider threats and the need for fast and directed response. *SANS Institute InfoSec Reading Room, Tech. Rep*.

Legg, P. A., Buckley, O., Goldsmith, M., and Creese, S. (2015). Caught in the act of an insider attack: detection and assessment of insider threat. In *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*, pages 1–6. IEEE.

Silowash, G., Cappelli, D., Moore, A., Trzeciak, R., Shimeall, T. J., and Flynn, L. (2012). Common sense guide to mitigating insider threats 4th edition. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.