Relatório referente ao Projeto Final
Linguagem de Programação II - IMD 0040

Erick W. M. Machado[1], Tyrone M. Damasceno[2]

[1, 2]Instituto Metrópole Digital – Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brasil

erickwilly7@gmail.com, tyronedamasceno@gmail.com

Abstract. *This report describes an application to control and detect internal attacks on systems. For this a technique of behavior analysis is used from activities logs of the users with purpose of detecting anomalous behaviors.*

Resumo. *Este relatório descreve uma aplicação que visa controlar e detectar ataques internos em sistemas. Para isso utiliza-se uma técnica de análise de comportamento a partir dos Logs de atividades dos usuários com finalidade de detectar comportamentos anômalos.*

1. Problema abordado

Nos dias de hoje existe uma grande preocupação com a segurança de sistemas, entre os riscos avaliados, um fator preocupante é a existência de ameaças realizadas por agentes internos que usam de suas permissões de acesso para comprometer a segurança de um sistema.

2. Introdução

Neste relatório pretendemos abordar uma solução desenvolvida seguindo o paradigma de orientação a objetos para análise e detecção de ameaças internas em sistemas. A solução consiste em uma análise das atividades de diversos usuários em uma rede a partir dos seus logs de atividades e identificação de comportamentos anômalos que fujam do padrão geral da rede.

A partir desta análise tem-se uma listagem de usuários considerados suspeitos, aos quais se faz interessante uma maior investigação, etapa posterior ao aqui trabalhado.

Cada usuário tem seu perfil modelado em forma de árvore, tendo esta como raiz o próprio usuário com seus dados. No nível seguinte (filho) do usuário temos uma listagem de todos os dispositivos a partir de onde ele acessou a rede e para cada dispositivo temos como filhos todas as atividades ali realizadas.

3. Estruturas de Dados

Neste projeto foram utilizadas estruturas de dados simples, listas e tabelas *hash*, além de uma estrutura em forma de árvore genérica implícita.

3.1 Listas

As listas utilizadas foram da coleção *ArrayList<>* existente no pacote *Java.util* da

linguagem de programação Java. As listas tiveram como função agrupar os filhos de cada nó de acordo com a ordem de inserção dos mesmos. Esta tem um tempo de inserção constante O(1) enquanto sua busca e remoção são de ordem linear O(n).

3.2 Tabela *hash*

A tabela *hash* utilizada foi da coleção *HashMap<>* existente no pacote *Java.util* da linguagem de programação Java. A tabela teve como função simular nossa rede para que o acesso a cada usuário fosse otimizado visto que a busca nesta estrutura é da ordem de O(1) com alta probabilidade. É importante ressaltar que uma tabela *hash* define uma situação de par chave/valor, onde como chaves temos *Strings* com os identificadores únicos dos usuários e como valor temos os seus perfis.

3.3 Árvore

A árvore utilizada não foi definida explicitamente, porém, nossa estrutura pode-se definir como árvore visto que é composta de vários nós de mesmo tipo sendo um deles uma raiz e os demais filhos, filhos de filhos e assim recursivamente. A inserção e busca na nossa árvore é da ordem de O(log n) onde "n" é a quantidade de folhas (atividades no contexto) em nossa árvore.

4. Solução do problema

Inicialmente é construída uma rede de perfis de usuário a partir da leitura dos arquivos previamente selecionados pelo cliente (os arquivos devem estar em formato .csv e seguir um padrão pré-determinado).

Após termos todos os perfis modelados com suas respectivas atividades e ter a rede construída, podemos selecionar algum usuário específico e escolher algum agrupamento de atividades deste mesmo usuário em um dia ou período. Bem como fazer uma verificação geral na rede de usuários que fujam do padrão e apresentam anomalias em suas atividades e ter uma listagem destes.

Foram definidos dois parâmetros para que um usuário fosse considerado suspeito, são estes, um determinado usuário não ter realizado nenhuma atividade ou este usuário ter feito uma quantidade de atividades maior ou igual a três vezes a média de atividades geral no sistema. É importante ressaltar que esses parâmetros foram definidos a partir de critérios pessoais dos desenvolvedores e são um modelo inicial, havendo a possibilidade de implantação de novos parâmetros.

5. Projeto OO

A implementação da solução seguiu o paradigma de orientação a objetos, apresentando diversos componentes característicos como herança, polimorfismo, modelagem em classes, tratamento de exceções e padrões de projeto.

5.1 Padrões de projeto

Na solução do projeto foi utilizado apenas um padrão de projeto, o *Singleton* que garante a existência de apenas uma instância para determinada classe. No nosso caso a classe Network, a qual definia a rede. Isso ocorre para que não haja acidentalmente a

criação de mais de uma rede e os usuários de um mesmo sistema sejam separados, o que acarretaria na inconsistência da análise de comportamento e prejudicaria nosso controle e detecção de ameaças.

5.2 Diagrama de classes

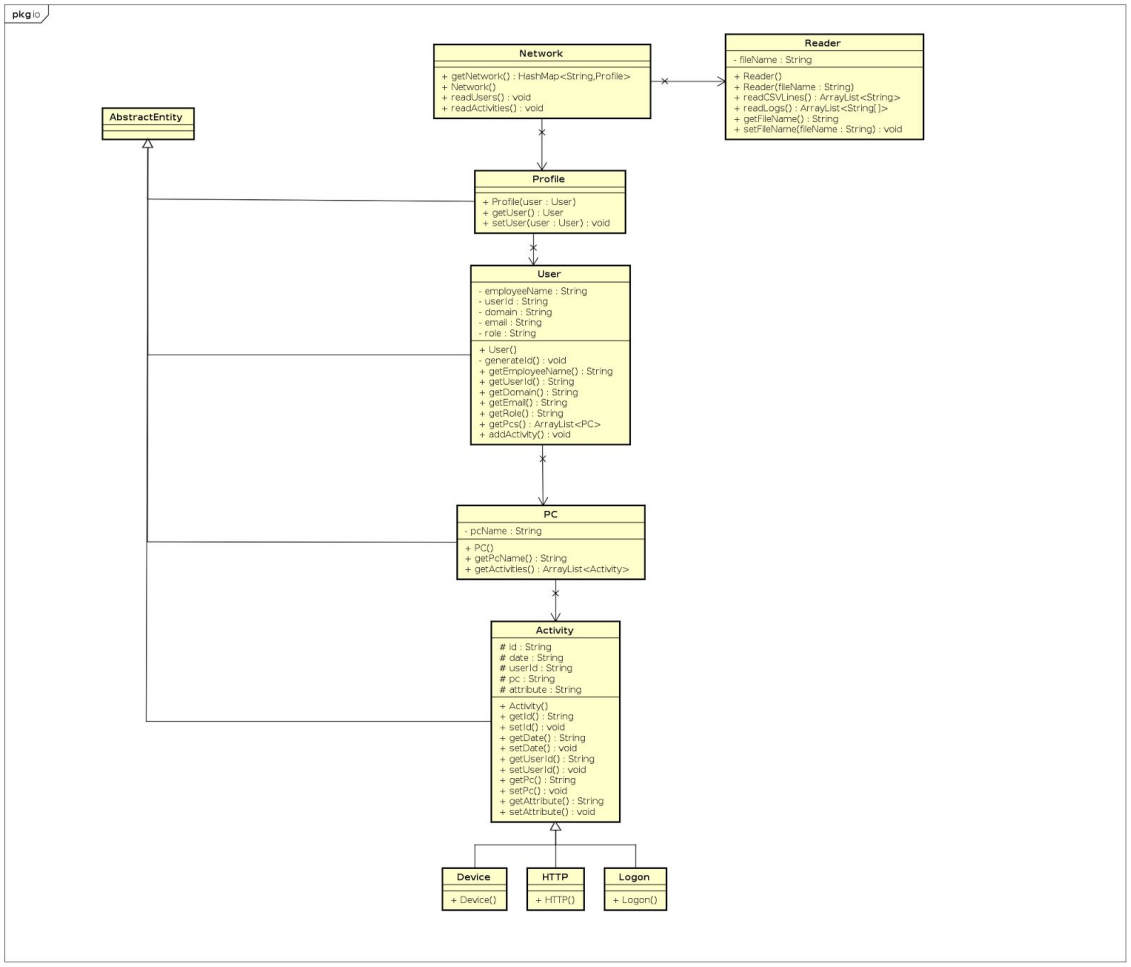

Figura 1. Diagrama UML Para Leitura e Montagem de Perfis

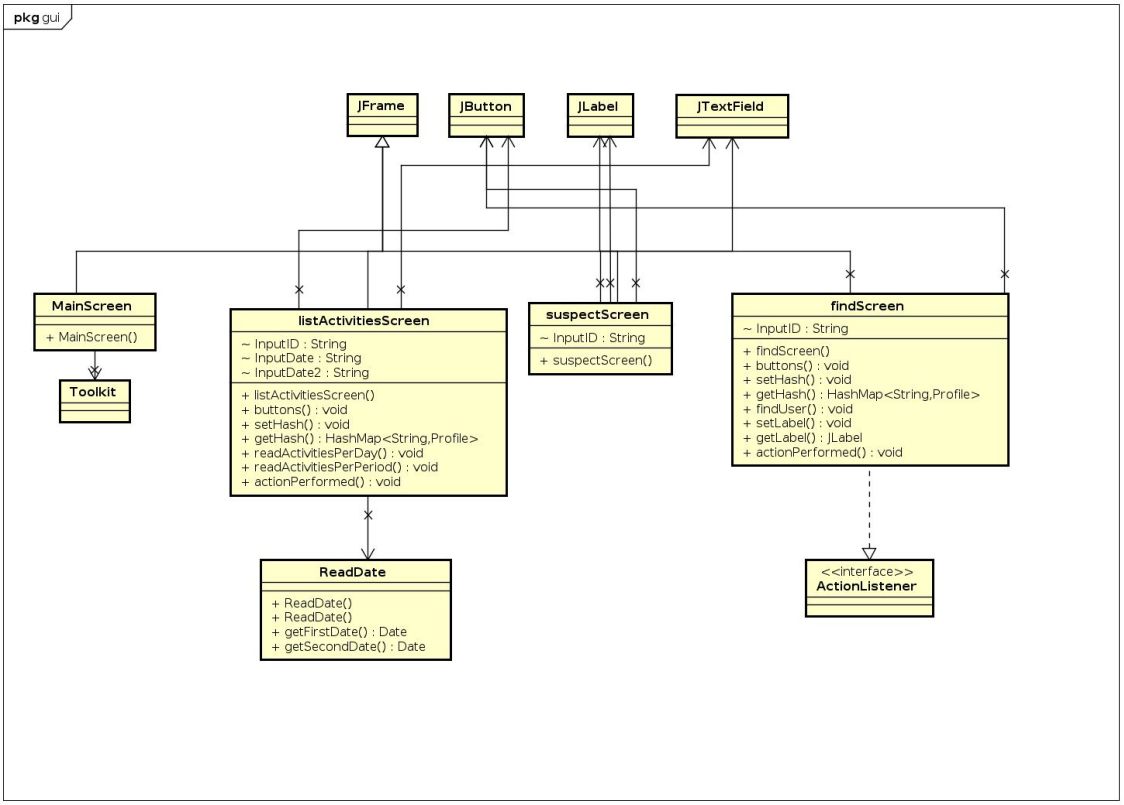

Figura 2. Diagrama UML da Interface Gráfica

7. Dificuldades e funcionalidades não desenvolvidas

No geral houveram algumas dificuldades e imprevistos durante a execução e implementação do projeto mas foram contornados com o consenso dos componentes do grupo e auxílio do professor.

Um problema enfrentado pelo grupo foi o processamento de uma grande massa de dados, visto que a máquina virtual do Java nos disponibiliza apenas 256Mb de memória e não foi implementado um uso de memória em disco, o que tornou inviável o processamento de muitos dados.

Também não houve tempo hábil para implementação do histograma de atividades dos usuários.

7. Conclusão

É visível de notar que a aplicação se trata de um protótipo com funcionalidade limitada para aplicação e uso em sistemas reais porém percebe-se que a estratégia escolhida é de grande capacidade e confiabilidade. O protótipo apresenta alguma robustez e atende às expectativas do que lhe foi requisitado.

O grupo conseguiu executar com êxito as demais especificações solicitadas na descrição do projeto e considera este como de grande valor de aprendizado e experiência no desenvolvimento de grandes aplicações, conscientes de que este é um projeto pequeno e simples ao que será encontrado fora da academia no mercado de

trabalho.

Por fim, o grupo informa que pretende aperfeiçoar a solução em momentos posteriores de maneira tal que esta venha a atender todos os pontos levantados com a maior otimização e qualidade possível. Os autores gostariam também de agradecer a oportunidade de realizar este projeto e se colocar à disposição para futuros esclarecimentos e discussões.

8. Referências

GAMMA, E.; HELM, R.; JOHNSON R.; VLISSIDES, J. (1994) “Design Patterns: Elements of Reusable Object-Oriented Software”

Leitura e escrita de arquivos de texto em Java. Disponível em: <www.devmedia.com.br/leitura-e-escrita-de-arquivos-de-texto-em-java/25529>. Acesso em 16 de Junho de 2017

BARNES, David J. (2009) “Programação orientada a objetos com Java: uma introdução prática usando o BlueJ"

SILVA, Carlos E.; XAVIER JUNIOR João C. (2017) “Representando Comportamento de Usuários para Detecção de Ameaças Internas”