Projeto de Programação IMD0040
Representando Comportamento de Usuários para Detecção de Ameaças Internas

Carlos Eduardo da Silva, João Carlos Xavier Júnior
Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital

Versão 0.3 - 09/05/2017

1 Introdução

Em um mundo cada vez mais conectado, a preocupação com segurança tem recebido cada vez mais atenção. Em uma pesquisa realizada em 2016 pela empresa PWC-Brasil denominada "Pesquisa Global de Segurança da Informação", somente no Brasil houve um aumento de 274% nos incidentes de segurança da informação. Esse número pode ser ainda maior, levando em consideração que muitas empresas não reportam tais ataques com receio de manchar sua reputação diante do mercado e de seus clientes.

Além dos ataques realizados por agentes externos, um outro fator preocupante são os ataques realizados por agentes internos (denominado "*insider attacker*"ou "*insider threat*") [1, 2]. Uma ameaça interna é caracterizada por usuários autorizados que exploram suas permissões de acesso para comprometer a segurança de uma organização e de seus sistemas computacionais [3, 4]. Estes usuários podem incluir empregados, terceirizados e parceiros comerciais que podem causar prejuízos de maneira intencional (para ganho pessoal ou vingança) ou através de erros (causados por negligência ou por usuários com treinamento insuficiente) [5].

Uma das maneiras de lidar com o problema de ameaças internas envolve o uso de técnicas de detecção de anomalias [1, 6]. Baseado em um log de atividades de um sistema, uma estrutura de dados pode ser utilizada para se construir um perfil que descreve as atividades de um usuário do sistema. Uma vez que se tem perfis dos usuários do sistema, é possível realizar comparações entre esses perfis para determinar como observações diárias variam em relação a observações passadas, ou identificar alterações no comportamento padrão dos usuários.

Neste contexto, este projeto consiste na implementação de um sistema orientado a objetos que permita a representação de perfis de usuários a partir de arquivos de log, e sua posterior visualização e comparação de forma a identificar usuários com comportamento suspeito. O sistema a ser desenvolvido deverá receber como entrada arquivos de log em formato csv, contendo uma sequência de atividades observadas. Após processar o arquivo de log, o sistema deverá construir

um perfil de uso para cada usuário identificado no log, e em seguida permitir a realização de comparações entre esses perfis de modo a identificar usuários que apresenta um comportamento que desvie dos outros usuários, ou de observações anteriores.

2 Modelando Perfil de Usuários

Para modelar perfis de usuários, seu sistema deverá se basear na estratégia apresentada em [1, 7, 8], onde cada usuário tem seu perfil capturado em uma estrutura baseada em árvore. A Figura 1 apresenta um exemplo desta visualização extraído de [8].

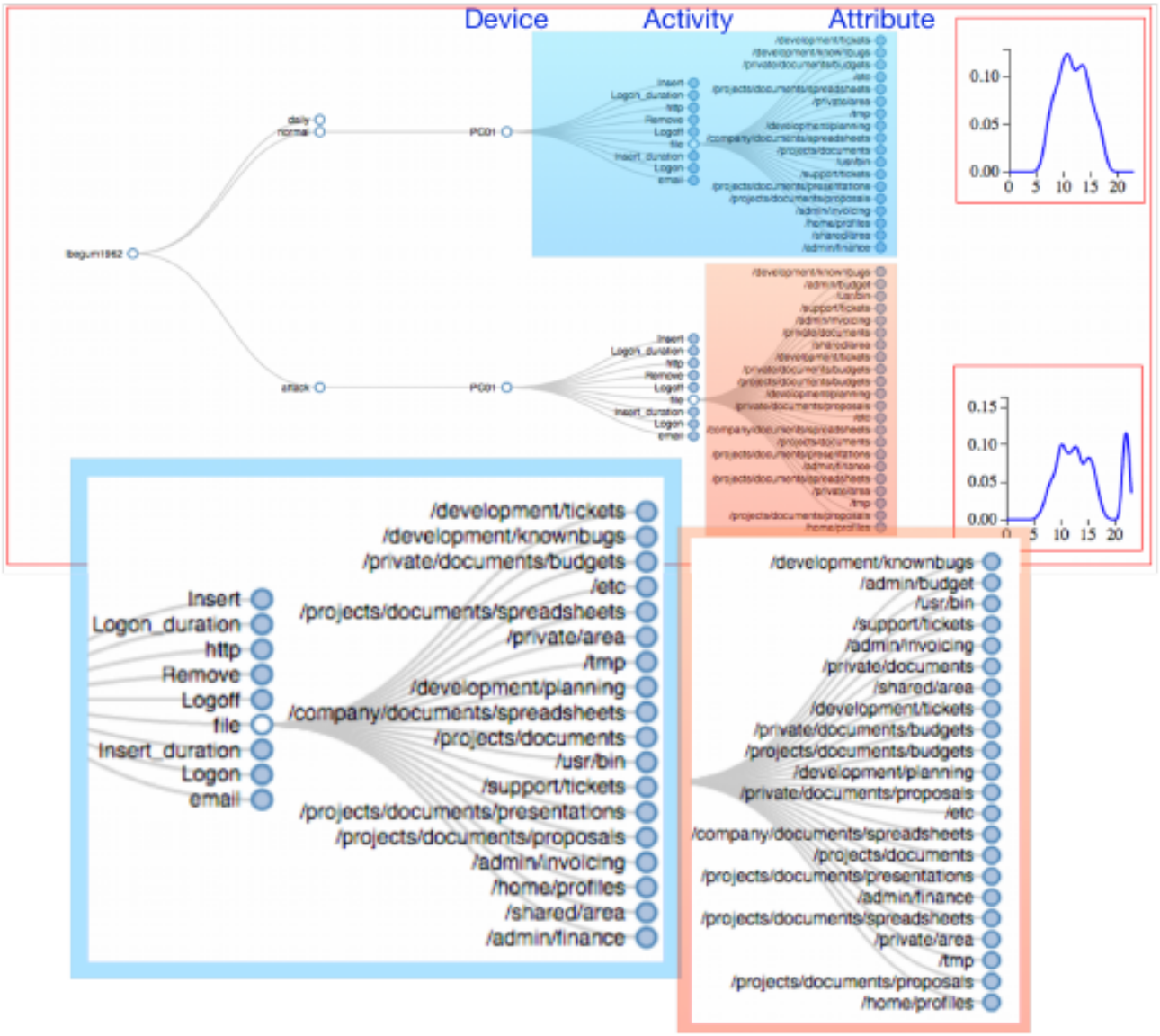

Figura 1: Exemplo da estrutura em árvore para representar perfis de usuários (extraído de [8]).

Deste modo, a raiz da árvore corresponde ao usuário sendo modelado, e contém uma série de informações sobre este usuário, tais como um identificador (ID) do usuário, e seus atributos como nome, e-mail, etc. Opcionalmente, a raíz pode corresponder também a um determinado papel, no caso onde se deseja modelar as ações de um determinado grupo de usuários.

A raiz pode ter até três nós filhos: observações para o dia corrente, observações anteriores que são consideradas normal, e observações consideradas suspeitas. Cada um desses ramos seguem a mesma estrutura hierárquica. Em seu primeiro nível encontramos todos os dispositivos (*Device*) que o usuário em questão foi observado (os dispositivos que os usuários estavam utilizando quando desempenhou a ação sendo logada), por exemplo, o computador utilizado pelo usuário.

O próximo nível (*Activity*) consiste em todas as atividades que o usuário desempenhou naquele dispositivo. Atividades possíveis incluem inserir um pen-driver (*insert*), realizar login em um computador (*logon*), enviar um e-mail (*email*), abrir um arquivo no disco (*file*) ou acessar uma página na Internet (*http*).

O próximo nível exibe um conjunto de atributos, caso seja aplicável, que correspondem ao objeto alvo da ação. Por exemplo, caso o usuário tenha acessado um website, o atributo corresponde à url acessada. Caso o usuário tenha enviado um e-mail, o atributo corresponde ao endereço de e-mail do destinatário.

Adicionalmente, cada nó de um perfil mantém um histograma com 24 posições, que representa a frequência de uso em cada hora do dia.

3 Objetivos Específicos

O projeto apresenta os seguinte objetivos específicos que deverão ser contemplados pelo seu programa:

1. Leitura de arquivos de log como entrada de dados.
2. Montagem dos perfis de usuários.
3. Visualização dos perfis de usuários.
4. Detecção de anomalias.

De modo a atender a esses objetivos, detalhamos abaixo os requisitos que deverão ser considerados.

3.1 Entrada de Dados

Seu programa deverá receber como entrada um arquivo de log de acordo com o formato definido pelo CMU-CERT[1] [9]. Cada entrada do arquivo de log corresponde a uma das diferentes atividades que podem ser desempenhadas pelos usuários. Para seu projeto, você deverá utilizar como base o arquivo disponibilizado no sigaa (este arquivo é um sub-conjunto do conjunto de teste r1 disponibilizado pelo CERT).

[1] `https://www.cert.org/insider-threat/tools/index.cfm`

O conjunto de teste contém os seguintes arquivos/diretórios:

- `device.csv`: Registra as atividades de inserir (Connect) ou remover (Disconnect) um pen-driver.
- `http.csv`: Registra as páginas Web acessadas pelos usuários.
- `logon.csv`: Registra as atividades de login (`Login`) e logoff (`Logoff`) dos usuários.
- `LDAP.csv`: Listagem de usuários ativos.

Cada arquivo segue o formato csv com os seguintes campos:

- `device.csv`: id, date, user, pc, activity (connect/disconnect)
- `http.csv`: id, date, user, pc, url
- `logon.csv`: id, date, user, pc, activity (Logon/Logoff)
- `LDAP`: employee_name, user_id, email, domain, role

ATENÇÃO: Esses dados foram gerados de maneira aleatória, principalmente os endereços Web, que não foram verificados quanto ao seu conteúdo. Portanto, NÃO TENTEM ACESSAR ESSES ENDEREÇOS, pois alguns deles podem apontar para páginas Web maliciosas.

Os campos id são únicos dentro de um mesmo arquivo, mas isso não quer dizer que eles são únicos globalmente (entre todos os arquivos).

Você deverá definir uma estrutura orientada a objetos que lhe permita processar todos os arquivos de log, e que possa ser utilizada como base para a construção dos perfis de usuários. Lembre-se de sempre começar sua leitura a partir da lista de usuários.

Dica: Você pode ser basear no projeto `weblog-analyzer` do capítulo 4 do livro do BlueJ.

3.2 Montagem dos Perfis de Usuários

Para cada usuário identificado no log, seu sistema deverá construir um perfil de atividades baseado na estrutura apresentada na Seção 2.

Seu sistema deve permitir agrupar as atividades de um usuário de acordo com um intervalo de tempo definido pelo operador do sistema. Isto é, o segundo nível da árvore deverá considerar pelo menos duas opções das apresentadas abaixo:

1. Agrupar todas as atividades observados do usuário em um único nó e considerar este o comportamento passado do usuário;
2. Agrupar as atividades observadas de acordo com o dia em que ela ocorreu. Neste caso, existirá uma sub-árvore para cada dia de atividades observadas;
3. Agrupar as atividades observadas de acordo com uma janela de tempo definida pelo operador do sistema.

Sua árvore deverá ser montada de acordo com a leitura dos arquivos de log. Para cada entrada de log, seu sistema deve comparar se existe uma entrada na ávore do usuário correspondente. Em caso negativo, basta adicionar um novo nó na posição adequada para tal. O histograma associado com este nó, e os nós superiores, deverão ser atualizados de maneira apropriada.

Seu sistema deverá incluir mecanismos que permita salvar as árvores geradas em arquivo, que podem ser carregados pelo operador posteriormente.

3.3 Visualização dos Perfis de Usuários

Seu sistema deverá possuir mecanismos para a visualização de um perfil de usuário, permitindo a um operador selecionar o usuário de interesse. Adicionalmente, seu sistema deverá permitir a um operador obter informações sobre um determinado nó além de visualizar seu histograma.

3.4 Detecção de Anomalias

Uma vez que perfis de usuários foram criados, é possível agora analisar esses dados em busca de ameaças internas, ou seja, aqueles usuários que apresentem um comportamento anômalo em relação aos outros usuários logados.

Uma das maneiras de fazer isso é cálcular uma média de todos os usuários do sistema (baseado nos histogramas do perfil) e então comparar perfis de usuários individuais com esta média, onde aqueles usuários que estiverem mais distante da média (por exemplo, fora do desvio padrão) podem ser considerados como usuários com comportamentos anômalos.

Outras possibilidades são apresentadas nos artigos [1, 7, 8]. Você deverá escolher e implementar uma abordagem para detecção de anomalias e implementar em seu sistema.

4 Modelagem e Implementação

Seu sistema deverá ser modelado de acordo com as melhores práticas de orientação a objetos, e ser implementado com a linguagem de programação Java. Lembre-se de agrupar classes com funcionalidades semelhantes em pacotes.

As estruturas de dados relacionadas à implementação de árvore e do histograma deverá ser completamente definida e implementada como parte de seu projeto.

Para aquelas estruturas que foram estudadas em EDB1 e LP1, estruturas lineares em geral, você poderá utilizar as classes da API Collection de Java.

Seu programa deverá incluir também uma interface gráfica que ofereça pelo menos as seguintes funcionalidades:

- Abrir os arquivos de log csv.
- Definir o critério de agrupamento de logs (ex: diário, agregado, ou intervalo de tempo).
- Iniciar o processamento dos logs (recomendamos que seu sistema só permita o início do processamento após todos os arquivos necessários estiverem sido abertos).

- Visualizar o perfil de um determinado usuário.
- Salvar o resultado do processamento (os perfis dos usuários) em arquivo.
- Abrir os perfis dos usuários.
- Comparar perfis de dois usuários; ou detectar usuário anômalo.

5 Entrega e Avaliação

Seu grupo deverá submeter, via sigaa, um arquivo compactado contendo:

- Manual do operador: Um manual de utilização de seu sistema, descrevendo como instalar, inicializar, e utilizar o sistema.
- Código fonte do sistema desenvolvido, incluindo um documento README.TXT contendo instruções de como se pode compilar o código fonte.
- Relatório Técnico, contendo pelo menos as seguintes seções:
  - Introdução
  - Descrição do problema abordado
  - Descrição geral das estruturas de dados utilizadas
  - Descrição da abordagem de solução do problema
  - Detalhes de projeto OO (diagramas de classes destacando as decisões de projetos tomada, bem como identificando padrões de projeto aplicados)
  - Explicação detalhada dos algoritmos utilizados (com pseudocódigo e análise de complexidade)
  - Conclusão
  - Referências

  O relatório deverá ser feito seguindo o template da Sociedade Brasileira de Computação (SBC) que pode ser encontrado no seguinte endereço: `http://www.sbc.org.br/documentos-da-sbc-summary/169-templates-para-artigos-e-capitulos-de-livros/878-modelosparapublicaode`

A avaliação do projeto considerará os seguintes critérios.

- Funcionamento do software da maneira solicitada, atendendo a todos os requisitos.
- Modelagem adequada do problema do ponto de vista algorítmico, o que engloba:
  - Escolha das estruturas de dados estudadas em EDB2, de forma a maximizar a eficiência dos algoritmos.

  - Definição de métodos(s) para processamento dos arquivos de entrada.
  - Definição de método(s) para montar o perfil dos usuários.
  - Definição de método(s) para detecção de anomalias.

- Completude do relatório técnico no que se refere aos itens solicitados para o mesmo, incluindo a adequação ao modelo proposto pela SBC. O texto do relatório técnico deverá ser coerente, coeso e objetivo, contemplando as informações suficientes e necessárias para o entendimento do software desenvolvido.
- Acompanhamento do projeto
- Qualidade do projeto desenvolvido
- Apresentação.

Acarretarão diminuição na nota:

- Presença de erros de compilação e/ou execução.
- Falta de análise de complexidade dos algoritmos implementados.
- Falta de documentação do programa em JavaDoc.
- Entrega incompleta.
- Mal uso da norma culta da Língua Portuguesa.

Alguns dos elementos de avaliação serão detalhados a seguir:

5.1 Acompanhamento do projeto

Ao longo do andamento do projeto serão definidas datas de checkpoint onde cada grupo deverá fazer uma breve apresentação de seu progresso. Cada checkpoint terá um entregável esperado.

- **Checkpoint 1**: Tarefas a serem desenvolvidas pelo grupo para o andamento do projeto com alocação de responsáveis por cada tarefa. **Deadline**: 25/05/2017
- **Checkpoint 2**: Projeto OO (diagrama de classes) inicial da solução. **Deadline**: 01/06/2017
- **Checkpoint 3**: Implementação - fase 1. **Deadline**: 06/06/2017
- **Checkpoint 4**: Implementação - fase 2. **Deadline**: 13/06/2017

5.2 Qualidade do projeto desenvolvido

Neste aspecto serão considerados os diversos conceitos abordados ao longo do semestre. Cada critério abaixo tem uma pontuação máxima de 10 pontos.

Critérios a serem considerados

- Modelagem em Classes
- Herança
- Polimorfismo
- Coesão e Acoplamento
- Tratamento de exceções
- Documentação (Javadoc)
- Apresentação
- Padrão de projeto

5.3 Apresentação

Cada dupla terá **20 minutos** para fazer sua apresentação. Nesta apresentação, os grupos deverão demonstrar o funcionamento do programa desenvolvido. Além disso, deverão estar aptos a responder questões sobre o desenvolvimento do projeto.

Importante: Não será dada uma única nota ao grupo. Cada componente do grupo receberá uma nota de acordo com seu desempenho durante a apresentação.

O programa será avaliado como um todo, ou seja, os requisitos não receberão pontuações individualmente. Dessa forma, a falta de um ou mais requisitos acarretará na perda de pontos, que poderá ser compensada (não totalmente, claro) através de outros componentes bem desenvolvidos.

Componentes adicionais serão muito bem vistos, desde que implementados de maneira racional. Lembre-se de usar o bom senso para não transformar criatividade em bagunça.

6 Dúvidas e Dicas

Se durante o desenvolvimento do projeto tiver alguma dúvida sobre a tarefa solicitada tente três possibilidades. Primeiro leia este documento, ou novamente ou pela primeira vez. Segundo, procure na Internet por mais informações. Por último, pergunte ao professor. Se a sua dúvida for interessante ela e a resposta serão acrescentadas neste documento.

7 Autoria, Política de Colaboração, Plágio e Duplicação de Material

Este trabalho poderá ser desenvolvido em duplas com dois integrantes. Eventualmente, alguns grupos poderão ser convocados para uma entrevista. O objetivo de tal entrevista é comprovar a verdadeira autoria do código entregue. Assim, qualquer um dos componentes do grupo deve ser capaz de explicar qualquer trecho do código do projeto.

O trabalho em cooperação entre alunos da turma é estimulado. Porém, esta interação não deve ser entendida como permissão para utilização de código ou parte de código de outras equipes, o que pode caracterizar a situação de plágio. Trabalhos plagiados receberão nota ZERO automaticamente, independente de quem seja o verdadeiro autor dos trabalhos infratores.

Um dos motivos mais comuns para problemas de plágio em trabalhos de programação é deixar para fazer o trabalho de última hora. Evite isso, e tenha certeza de descobrir o que você tem que fazer (que não significa necessariamente como fazer) o mais cedo o possível. Em seguida, decida o que você precisará fazer para completar o trabalho. Isto provavelmente envolverá alguma leitura e prática de programação. Se estiver em dúvida sobre o que foi pedido pelo trabalho, pergunte ao professor da disciplina.

Outra razão muito comum é trabalhar em conjunto com outros alunos da disciplina. Não faça trabalhos de programação em conjunto, ou seja, não utilizem um único PC, ou sentem lado a lado, principalmente, digitando código ao mesmo tempo. Discutam as diversas partes do trabalho, mas não submetam o mesmo código.

Não é aceitável a submissão de código com diferenças em comentários e nomes de variáveis, por exemplo. É muito fácil para nós detectar quando isso for feito, e verificaremos esse caso. Nunca deixe outra pessoa ter uma cópia de seu código, não importando o quão desesperado eles possam estar. Sempre aconselhe alguém nesta situação a buscar ajudar com o professor da disciplina.

Referências

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

[2] E. Cole, "Insider threats and the need for fast and directed response," SANS Institute InfoSec Reading Room, Tech. Rep., 2015.

[3] G. Silowash, D. Cappelli, A. Moore, R. Trzeciak, T. Shimeall, and L. Flynn, "Common sense guide to mitigating insider threats," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2012-TR-012, 2012.

[4] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes*, 1st ed. Addison-Wesley Professional, 2012.

[5] International Organization for Standardization, "ISO/IEC 27005:2011: Information technology – Security techniques – Information security risk management," 2011.

[6] C. E. da Silva, J. D. S. da Silva, C. Paterson, and R. Calinescu, "Self-adaptive role-based access control for business processes," in *The 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2017.

[7] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, no. 99, pp. 1–10, 2015.

[8] P. A. Legg, "Visualizing the insider threat: Challenges and tools for identifying malicious user activity," in *IEEE Symposium on Visualization for Cyber Security (VizSec2015)*, 2015, pp. 1–7.

[9] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2012 IEEE Symposium on Security and Privacy Workshops*, 2013, pp. 98–104.

Histórico de alterações

- Versão 0.1 (21/04/2017): Primeira versão para revisão.
- Versão 0.2 (26/04/2017): Segunda versão para revisão.
- Versão 0.3 (09/05/2017): Primveira versão pública.