UNIVERSIDADE CRUZEIRO DO SUL

EDSON MOREIRA CEZAR

ÁRVORES RECURSIVAS CONSTRUÍDAS COM CHAVE VALOR

São Bernardo do Campo
2019

Trabalho de Conclusão de Curso apresentado ao Curso de Pós-Graduação da Universidade Cruzeiro do Sul, como requisito parcial para obtenção do Título de MBA em Segurança da Informação.

Orientador: Allan Pitter

São Bernardo do Campo
2019
EDSON MOREIRA CEZAR

Dedico este trabalho a todas as pessoas que de alguma maneira me auxiliaram ou passaram pela minha vida.

AGRADECIMENTO

A todas as pessoas que de alguma maneira passaram e me auxiliaram ao longo da vida.

“To deny our own impulses is to deny the very thing that makes us human.”.

Mouse - Matrix

RESUMO

Nesse trabalho vou abordar como utilizar árvores com chave e valor para efetuar tarefas de segurança da informação, como um simples bloqueio de pastas em um Sistema Operacional, as árvores geralmente são percorridas com um índex indo para o outro índex, na minha abordagem são utilizados índex e valor na construção da arvore, assim posso executar tarefas baseada no valor existente, ao invés de um nó inteiro.

Palavras-Chave: Árvores, Recursão, Algoritmos, Sistema Operacional, Segurança

ABSTRACT

In this work I will address how to use trees with key and value to perform tasks of information security, such as a simple blocking of folders in an Operating System, trees are usually traversed with an index going to the other index, in my approach are used index and value in building the tree, so I can execute tasks based on the existing value, rather than an entire node.

Keywords: Trees, Recursion, Algorithms, Operating System, Security

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

HTML - Hypertext Markup Language

HTTP - HyperText Transfer Protocol

XML - e**X**tensible **M**arkup **L**anguage

SUMÁRIO

1 INTRODUÇÃO

Na minha abordagem farei uma simples passagem pela definição sobre conceitos relacionados ao meu direcionamento, uma simples pesquisa concluindo com a solução apresentada em forma de algoritmos utilizando a linguagem JAVA.

Meu objetivo é apresentar de forma simples a utilização de árvores recursivas para a solução de um problema simples apresentado, o sistema operacional tem restrição de acesso, mas sem a utilização de um software específico não conseguimos bloquear o acesso a uma pasta, assim sendo a uma árvore de diretórios.

Meu simples exemplo de uma das maneiras que encontrei para fazer isso em um sistema que desenvolvi a algum tempo atrás, e foi um grande caso de sucesso e decidi exemplificar de uma maneira simples, abstraindo a complexidade e indo direto para a solução simplista.

Utilizei assim uma árvore recursiva e um exemplo simples da utilização de uma regra para bloqueio de acesso a uma pasta restrita específica, utilizando uma árvore com chave e valor, assim sendo consigo colocar um valor e uma regra simples atrelada a mesma.

2 COMPUTADOR MODERNO

Podemos definir que um computador moderno consiste em um ou mais processadores, alguma memória principal, discos, impressoras, teclado, mouse, tela, interfaces de rede e vários outros dispositivos de entrada / saída, ou seja, um sistema complexo. [4]

3 SISTEMA OPERACIONAL

Se todo programador tivesse que entender como todas essas coisas funcionam em detalhes, nenhum código jamais funcionaria, gerenciar todos esses componentes e usá-los de maneira ideal é um trabalho extremamente desafiador, por essa razão, os computadores são equipados com uma camada de software chamada sistema operacional, cujo trabalho é fornecer aos programas do usuário um modelo do computador melhor, mais simples e mais limpo e gerenciar todos os recursos mencionados. [4]

Podemos definir que a maioria dos usuários de computador tem alguma experiência com um sistema operacional, mas é difícil saber exatamente o que é um sistema operacional. Parte do problema é que os sistemas operacionais desempenham duas funções basicamente não relacionadas, estendendo a máquina e gerenciando recursos e, dependendo de quem está falando, você ouve principalmente uma função ou outra, a função do sistema operacional é apresentar ao usuário o equivalente a uma máquina ou máquina virtual estendida, que é mais fácil de programar do que o hardware subjacente. [5]

4 SEGURANÇA EM SISTEMAS OPERACIONAIS

Podemos dizer que muitas empresas possuem informações valiosas que desejam proteger, entre muitas outras coisas, essas informações podem ser técnicas (por exemplo, um novo design ou software), comerciais (por exemplo, estudos da concorrência ou planos de marketing), financeiras (por exemplo, planos de oferta de ações) ou legais (por exemplo, documentos sobre uma potencial fusão ou aquisição). [4]

A maioria dessas informações é armazenada em computadores, computadores domésticos também têm dados valiosos sobre eles, muitas pessoas mantêm suas informações financeiras, incluindo declarações fiscais e números de cartão de crédito, em seus computadores, até mesmo cartas de amor foram

digitalizadas, os discos rígidos atualmente estão cheios de fotos, vídeos e filmes importantes. [4]

À medida que mais e mais dessas informações são armazenadas em sistemas de computadores, a necessidade de protegê-las está se tornando cada vez mais importante. Proteger as informações contra o uso não autorizado é, portanto, uma grande preocupação de todos os sistemas operacionais. Infelizmente, também está se tornando cada vez mais difícil devido à ampla aceitação do aumento do sistema (e dos erros que o acompanham, bugs) como um fenômeno normal. [4]

O problema é causado por erros no software no computador, sistemas operacionais e aplicativos cada vez mais inchados garantem que não há escassez de bugs. Quando um bug é um bug de segurança, nós o chamamos de vulnerabilidade, se alimentamos esse software exatamente com os bytes corretos para acionar o bug, essa entrada de acionamento de bugs como essa geralmente é chamada de exploração (Exploit de vulnerabilidade). [4]

Com a Network a coisa se torna um pouco mais abrangente, quando você adiciona redes, de repente você introduz uma série de novas mensagens, existem problemas de segurança. Você não sabe quem está por aí e o que eles querem fazer. Você precisa ter muito cuidado para que as pessoas não travem sua máquina enviando pacotes maliciosos. Você não tem mais controle sobre quem está tentando entrar em contato com sua máquina. Além disso, muitas pessoas têm configurações diferentes. Com o TCP / IP, o padrão de rede, é difícil obter todos os tempos de espera. [6]

Quando procuramos proteger nossos dados, processos e aplicativos contra-ataques combinados, uma das maiores áreas em que encontramos pontos fracos está no sistema operacional que hospeda todos eles (seja um computador, roteador ou smartphone). Se não tomarmos cuidado para proteger nossos sistemas operacionais, realmente não temos base para chegar a uma postura de segurança razoavelmente forte. [7]

Há várias maneiras pelas quais podemos atenuar as várias ameaças e vulnerabilidades que podemos enfrentar a partir de uma perspectiva do sistema operacional. [7]

5 ALGORITIMOS

Uma das definições que mais gostei em minhas pesquisas de algoritmos, relata que um algoritmo pode ser definido como “uma sequência finita de instruções, cada uma com um significado claro e pode ser executada com uma quantidade finita de esforço em um período finito de tempo”, e da mesma deve ser preciso o suficiente para ser entendido pelos seres humanos. No entanto, para ser executado por um computador, geralmente precisamos de um programa que seja escrito em uma linguagem formal rigorosa; e desde computadores são bastante inflexíveis em comparação a mente humana, os programas geralmente precisam conter mais detalhes do que algoritmos. [1]

Outra de minhas abordagens preferidas é a definição de um algoritmo como uma ferramenta para resolver um problema computacional bem especificado. A declaração do problema especifica em termos gerais a relação de entrada / saída desejada. O algoritmo descreve um procedimento computacional específico para alcançar essa relação de entrada / saída. [2]

6 RECURSIVIDADE

A capacidade de um procedimento chamar-se a si mesmo, é chamada de recursão, sendo extremamente importante para os teóricos e os programadores práticos. [3]

Um tipo de procedimento particularmente interessante é o procedimento recursivo, isto é, um procedimento que se chama, direta ou indiretamente, por meio de uma cadeia de outros procedimentos. [3]

Podemos citar muitas estratégias e paradigmas de programação, muitos algoritmos úteis são recursivos na estrutura, para resolver um determinado problema, eles se chamam recursivamente uma ou mais vezes para lidar com

subproblemas intimamente relacionados. Esses algoritmos geralmente seguem uma abordagem de divisão e conquista, eles dividem o problema em vários subproblemas que são semelhantes ao problema original, mas de tamanho menor, resolvem os subproblemas de forma recursiva e combinam esses problemas soluções para criar uma solução para o problema original. [2]

O paradigma de dividir e conquistar envolve três etapas em cada nível da recursão: [2]

Divida o problema em vários subproblemas que são instâncias menores do mesmo problema, que aborda a estratégia de divisão. [2]

Conquiste os subproblemas resolvendo-os recursivamente. Se os tamanhos do subproblema forem pequenos o suficiente, no entanto, basta resolver os subproblemas de maneira direta, que aborda a estratégia de conquista. [2]

Combine as soluções para os subproblemas na solução para o problema original, assim é definida toda a estratégia final de divisão e conquista, quando ao final de dividir o problema em pequenas partes nós os reagrupamos e assim executamos o processo de solução. [2]

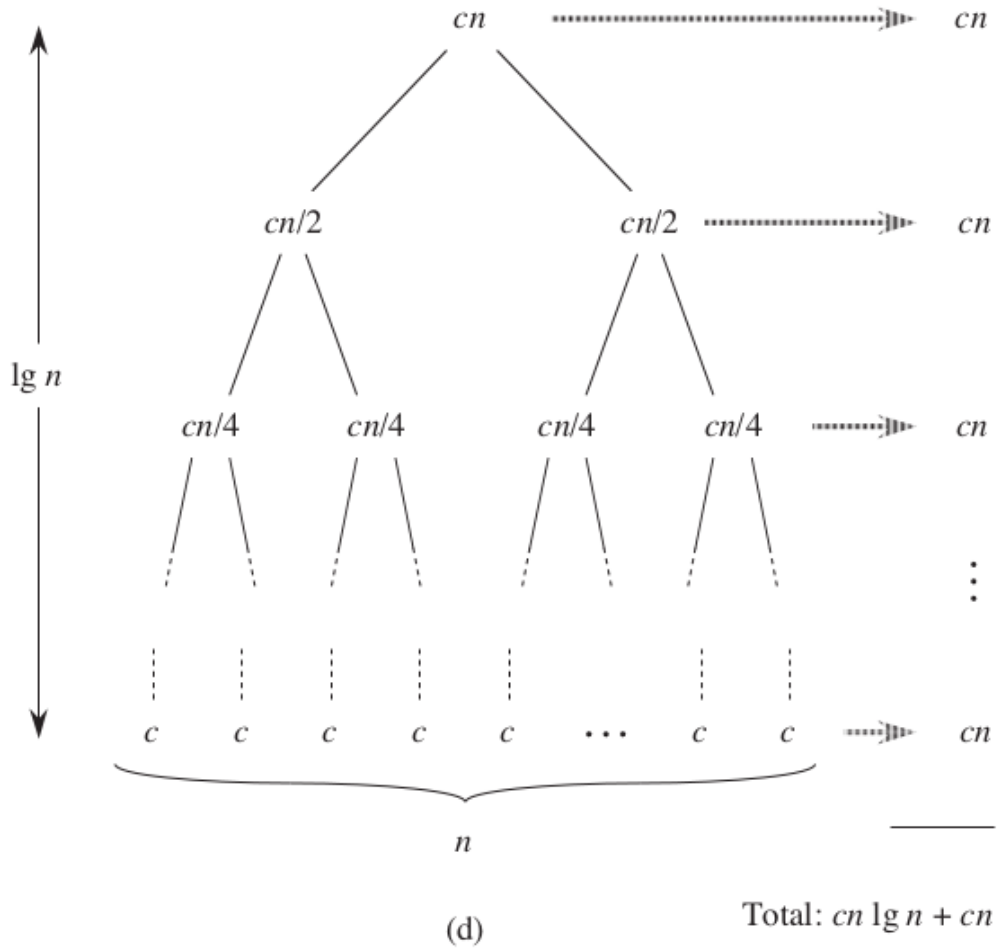
7 ARVORES

Na ciência da computação, uma árvore é uma estrutura de dados muito geral e poderosa que se assemelha a uma árvore real. Ele consiste em um conjunto ordenado de nós vinculados em um gráfico conectado, no qual cada nó tem no máximo um nó pai e zero ou mais nós filhos com uma ordem específica. [1]

Geralmente, podemos especificar uma árvore como consistindo de nós (também chamados de vértices ou pontos) e arestas (também chamadas de linhas, ou, para enfatizar o direcionamento, arcos) com uma estrutura em forma de árvore. [1]

Mais formalmente, uma árvore pode ser definida como a árvore vazia ou um nó com uma lista de árvores sucessoras. Os nós geralmente são, embora nem sempre, rotulados com um item de dados (como um número ou uma chave de pesquisa). Vamos nos referir ao rótulo de um nó como seu valor. Geralmente usamos nós rotulados por números inteiros, mas poderíamos facilmente escolher outra coisa, por exemplo, cordas de caracteres. [1]

Imagem de uma árvore recursiva[2]:



8 OPERAÇÕES EM ÁRVORES

Há muitas tarefas que podem ser executadas em uma estrutura de árvore; um comum é o de executar uma dada operação P em cada elemento da árvore.

Entende-se então que P é um parâmetro de toda a tarefa geral de visitar todos os nós ou, como é geralmente chamado, de travessia de árvores. [8]

Existem três ordens principais que emergem naturalmente da estrutura das árvores. Como a própria estrutura da árvore, elas são convenientemente expressas em termos recursivos. [8]

1. Preorder: R, A, B (visite a raiz antes das subárvores)
2. Inorder: A, R, B
3. Postorder: A, B, R (visite a raiz antes das subárvores)

Árvores binárias são frequentemente usadas para representar um conjunto de dados cujos elementos devem ser recuperados através de uma chave única. [8]

Existem seis ferramentas fundamentais que são utilizadas quando se trabalha com árvores. Essas operações são o que permitem que as árvores sejam o poderoso mecanismo da ciência da computação que elas são. [9]

As seis operações para a estrutura de dados da árvore são: [9]

1. Enumerando
2. Procurando
3. Adicionando
4. Excluindo
5. Poda
6. Enxerto

Alguns deles, como adicionar e excluir, são autoexplicativos. No entanto, conceitos como enxerto podem levar um pouco mais de estudo. [9]

Enumerando

Ao trabalhar com a estrutura de dados da árvore, uma das tarefas mais comuns é enumerar a árvore. Essencialmente, isso significa que seu algoritmo pode atravessar a árvore. [9]

Procurando

Em seguida, na lista de operações para a estrutura de dados da árvore está pesquisando. Pesquisar em árvores, especialmente em árvores de busca binária, é uma das ferramentas que torna o trabalho com essa estrutura de dados tão poderoso. [9]

Adicionando

Quando se trata de adicionar a uma árvore, o conceito é direto. Você simplesmente adiciona um novo nó à estrutura. A chave é que o elemento tem que seguir as regras da árvore. [9]

Excluindo

Ao excluir nós de uma árvore, o processo de remoção dos nós geralmente exige que você implemente um conjunto de processos para garantir que a árvore permaneça estável. Isso pode ser um processo complexo, como garantir que a árvore de pesquisa binária permaneça equilibrada após a remoção de um nó (discutiremos isso em detalhes mais adiante). Ou pode ser tão simples e atualizar uma referência em um único nó. [9]

Poda

Poda é o processo de remover uma seção inteira de uma árvore.[9]

Enxerto

Se o conceito de poda fizer sentido, você perceberá enxertos facilmente. O conceito de enxerto é adicionar uma seção inteira a uma árvore. [9]

9 – GERENCIAMENTO DE IDENTIDADE E ACESSO

Procurando unir a área de Segurança da Informação com minha abordagem um pouco de história ajudará a dar alguma perspectiva. As infraestruturas originais do Internet IAM (Identity and Access Management) foi baseada no protocolo RADIUS. Se você tiver idade suficiente, pense nos dias dos modems (ou, se não tiver, pense em uma conexão VPN ou WiFi, que ainda usa o RADIUS). [10]

Esses sistemas têm três partes: [10]

1. Um cliente RADIUS solicitando acesso à rede.
2. Dispositivo de rede que possui portas de modem ou algum outro recurso de rede.
3. Servidor RADIUS que fornece o AAA - autenticação, autorização e contabilidade (AAA—authentication, authorization, and accounting em inglês). O RADIUS foi desenvolvido pela Livingston Enterprises Inc. (agora parte da Alcatel Lucent) para controlar o acesso a "servidores de terminal" - dispositivos de rede que tinham uma alta concentração de modems. Mais tarde, tornou-se um padrão IETF. Hoje, o último A em "Triple-A" ("contabilidade") caiu da maioria dos sistemas modernos de IAM.

Avançando rapidamente alguns anos. As próximas fases do Internet IAM começaram a ocorrer quando a World Wide Web atingiu escala crítica. Acredite ou não, uma web onipresente não foi uma conclusão perdida. [10]

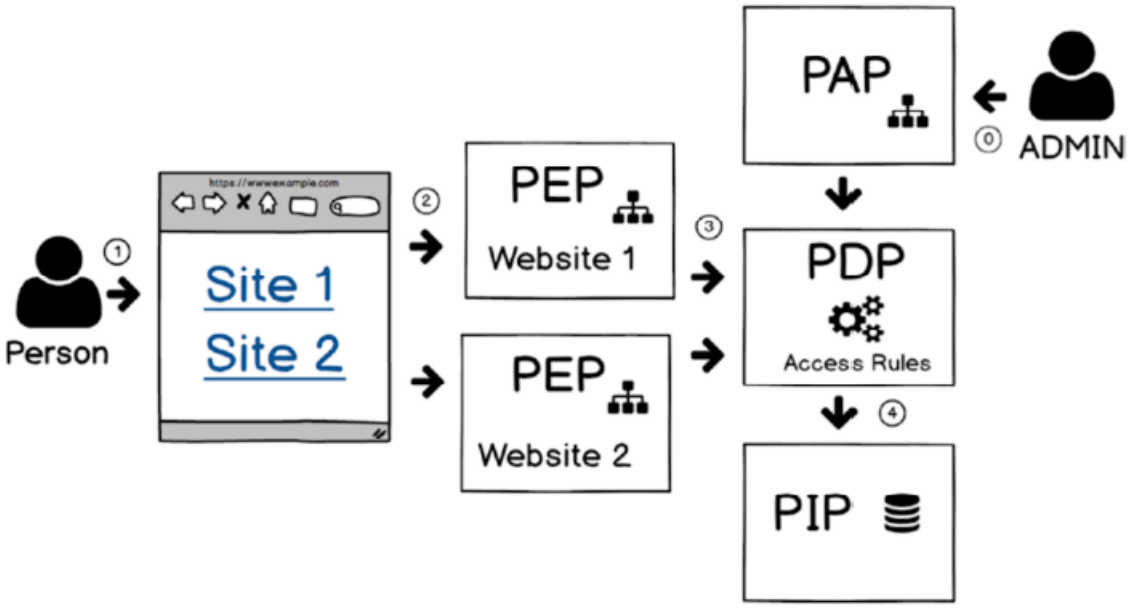
Em 1998, uma empresa chamada Netegrity (adquirida pela Computer Associates) lançou um produto chamado SiteMinder. Este era um novo tipo de servidor AA projetado para controlar o acesso a sites em vez de dispositivos de rede. O design foi semelhante ao RADIUS. [10]

Ainda dividido em três partes: [10]

1. Uma pessoa usando um navegador da Web (o cliente)
2. Um servidor da Web com o Agente do SiteMinder instalado

3. SiteMinder Policy Server central. No Policy Server, você poderia criar políticas sobre quais pessoas poderiam acessar quais servidores da web. Uma nova vantagem das plataformas web AA, como o SiteMinder, é que você poderia obter o Single Sign-On (SSO). Em outras palavras, a pessoa pode autenticar uma vez e acessar vários sites no mesmo domínio. Mais genericamente, esse padrão é comumente conhecido como "Padrão PDP-PEP". E existem algumas outras partes padrão. [10]

Imagem do padrão PDP-PEP[10]



Aqui está um breve resumo dos componentes: [10]

1. PDP (Policy Decision Point) - Conhece as políticas que as pessoas, usando quais clientes têm permissão para acessar quais recursos. De certa forma, é o cérebro do sistema. [10]
2. PEP (Policy Enforcement Point) - é responsável por consultar o PDP para descobrir se o acesso deve ser concedido. Geralmente há muitos PEPs. Por exemplo, pode haver centenas de servidores da Web em uma organização, cada um confiando no PDP para conceder acesso.
3. PAP (Policy Administration Point) - este é um tipo de interface que permite ao administrador definir as políticas no PDP. Pode ser um

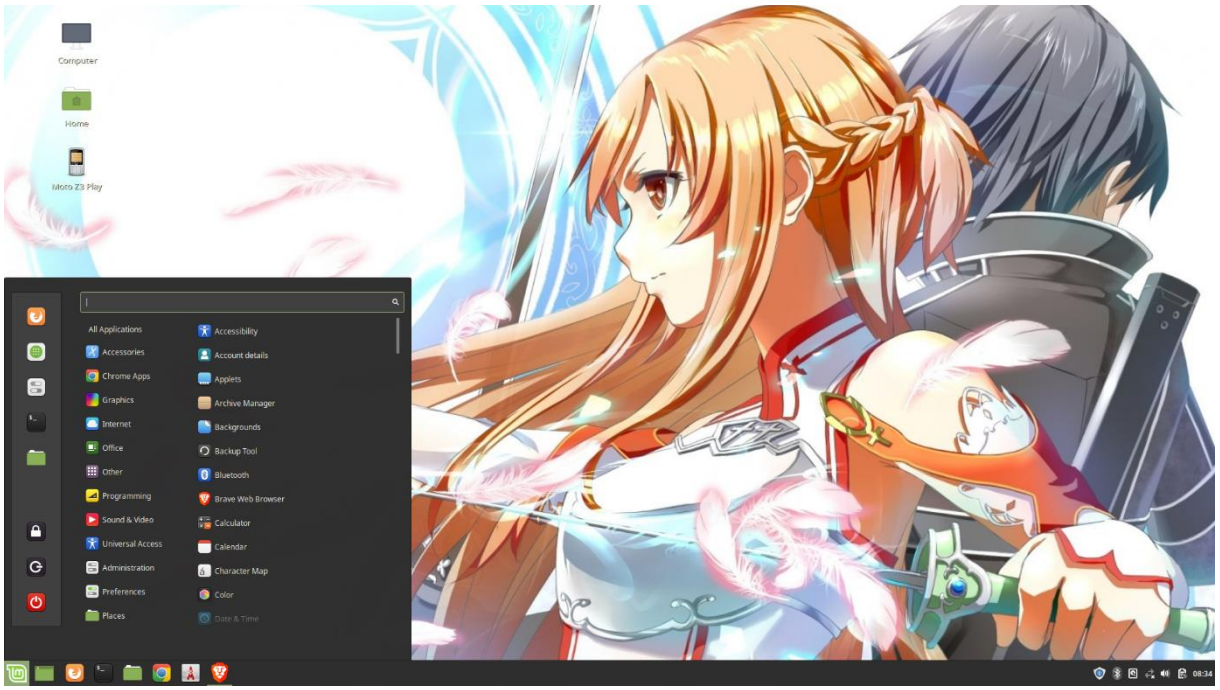
website ou uma interface de linha de comando. Sem um PAP, os administradores são forçados a gerenciar os dados da política em um banco de dados ou configurando arquivos. [10]

4. PIP (Policy and user information) - as informações do usuário persistem em algum lugar, normalmente em um banco de dados como um servidor LDAP. [10]

Finalizando uma breve história de controle de acessos, agora vamos unir minha abordagem com a área de segurança da informação, a minha simples ideia é controlar um pouco mais o acesso de sistemas por meio de árvores, usando assim uma árvore recursiva com chave e valor.

Assim posso limitar o acesso de um usuário de uma maneira simples, há um diretório, um arquivo, um simples link de acesso há um sistema.

Captura de tela do Linux Mint mostrando o menu inicial



Imaginando o sistema acima, posso gerenciar o acesso à diretórios, arquivos, paths de programas dos sistemas, usando essa mesma árvore, relacionando valores há mesma, para esse fim de gerenciamento de acesso, abordagem já utilizada com

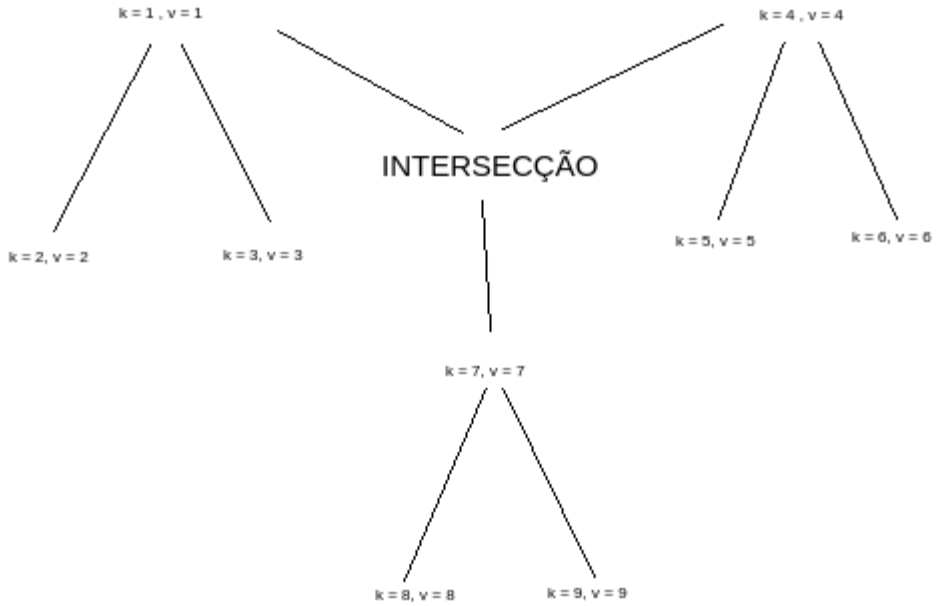
sucesso em um sistema da Alcatel Lucent no passado, já citada na breve história de Gerenciamento de Identidade e acesso.

10 ABORDAGEM

Nos próximos passos estarei explicando de maneira simples a abordagem e exemplificando-a com um simples algoritmo escrito na linguagem JAVA, minha abordagem usa para tais fins um algoritmo utilizando uma árvore recursiva.

Em minha abordagem utilizando uma chave e valor para percorrer uma árvore recursivamente, isso me permite criar árvores infinitas, assim podendo inclusive, agregar árvores umas às outras, ou então, inserir um novo membro diretamente ou excluí-lo diretamente, assim podendo moldar árvores com maior precisão.

Imagem árvore com chave e valor:



Exemplificando uma árvore recursiva diretamente, sem utilização da minha abordagem de chave e valor, utilizei um algoritmo em Java:

```
/**
 *@author Edson Moreira Cezar
```

```
**/
import javax.swing.*;
import java.awt.*;
import javax.swing.tree.*;

public class RecursiveTree extends JFrame{

        public RecursiveTree(){
  super("Simple JTree");

  Object[] languages = {
   "Languages",
   new Object[]{
    "Compiladas",
    "C++",
    "Delphi"
   },
   new Object[]{
    "Interpretadas",
    "JavaScript",
    "Python",
    "Ruby"
   },
   new Object[]{
        "Edson",
        "JavaScript",
        "Python",
        new Object[]{
                "Edson",
                "JavaScript",
                "Python",
                "Ruby"
        },
    }
  };

  DefaultMutableTreeNode root = build(languages);
  JTree tree = new JTree(root);

  Container c = getContentPane();
  c.setLayout(new FlowLayout());

  JScrollPane scrollPane = new JScrollPane(tree);
  c.add(scrollPane);

  setSize(400, 300);
  setVisible(true);
 }

 public static void main(String args[]){
  RecursiveTree app = new RecursiveTree();
  app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }

 private DefaultMutableTreeNode build(Object[] hier){

        DefaultMutableTreeNode node = new DefaultMutableTreeNode(hier[0]), son;

  for(int i = 1; i < hier.length; i++){
```

```
      Object sub_node = hier[i];
      if(sub_node instanceof Object[])
        son = build((Object[])sub_node);
      else
        son = new DefaultMutableTreeNode(sub_node);
      node.add(son);
    }

    return(node);
  }
}
```

11 CONSIDERAÇÕES FINAIS

Exemplificando uma árvore recursiva diretamente, com utilização da minha abordagem de chave e valor, utilizei um algoritmo em Java, dessa vez exemplificando de uma maneira o mais simples quanto possível, o simples bloqueio de uma pasta ou uma estrutura de acesso ao sistema operação, com uma regra escrita da maneira simples, poderia criar regras de acesso a um simples usuário bloqueando o acesso a uma estrutura inteira de arvores de diretórios:

```
import java.awt.Container;
import java.awt.FlowLayout;
import java.util.ArrayList;
import java.util.TreeMap;

import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTree;
import javax.swing.tree.DefaultMutableTreeNode;

/**
 *
 * @author Edson Moreira Cezar
 **/
public class RecursiveTreeKeyValue extends JFrame {

	ArrayList<String> blocked = new ArrayList<String>();

	public RecursiveTreeKeyValue() {
		super("Simple Tree");

		TreeMap<Object, Object> doors = new TreeMap<Object, Object>();

		TreeMap<Object, Object> room = new TreeMap<Object, Object>();

		blocked.add("block");

		room.put("First", "Entrance");
		room.put("Second", "Entrance");
```

```
        room.put("Third", "Entrance");
        room.put("Fourth", "block");

        doors.put("House", "Entrance");
        doors.put("Kitchen", "Entrance");
        doors.put("Bathroom", "Entrance");

        doors.put("son", room);

        doors.put("son2", room);

        DefaultMutableTreeNode root = build(doors, 1);
        JTree tree = new JTree(root);

        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        JScrollPane scrollPane = new JScrollPane(tree);
        c.add(scrollPane);

        setSize(400, 300);
        setVisible(true);
    }

    public static void main(String args[]) {
        RecursiveTreeKeyValue app = new RecursiveTreeKeyValue();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private DefaultMutableTreeNode build(TreeMap<Object, Object> doors, Integer a) {

        DefaultMutableTreeNode node = new DefaultMutableTreeNode(a), son;

        // Print keys and values
        for (Object door : doors.keySet()) {
            Object doorName = doors.get(door);
            String teste = door.toString();
            if (teste.contains("son")) {
                a++;
                son = build((TreeMap<Object, Object>) doorName, a);
                node.add(son);
            } else {
                son = new DefaultMutableTreeNode(door);

                if (!blocked.contains(doorName)) {

                    node.add(son);

                }
            }

        }

        return node;
    }

}
```

REFERÊNCIAS

[1] - Data Structures and Algorithms - Revised each year by John Bullinaria - School of Computer Science University of Birmingham Birmingham, UK

[2] - Introduction to Algorithms - Third Edition - 2009 Massachusetts Institute of Technology - THOMAS H. C ORMEN - Lebanon, New Hampshire, CHARLES E. L EISERSON - Cambridge, Massachusetts, RONALD L. R IVEST - Cambridge, Massachusetts, CLIFFORD S TEIN - New York, New York

[3] – STRUCTURED COMPUTER ORGANIZATION - SIXTH EDITION -ANDREW S. TANENBAUM, Vrije Universiteit, Amsterdam, The Netherlands, TODD AUSTIN, University of Michigan Ann Arbor, Michigan, United States

[4] – MODERN OPERATING SYSTEMS - FOURTH EDITION - ANDREW S. T ANENBAUM, HERBERT BOS - Vrije Universiteit, Amsterdam, The Netherlands

[5] - Operating Systems Design and Implementation, Third Edition By Andrew S. Tanenbaum - Vrije Universiteit Amsterdam, The Netherlands, Albert S. Woodhull - Amherst, Massachusetts

[6] - Just for Fun - The Story of an Accidental Revolutionary, Copyright © 2001 Linus Benedict Torvalds Co-Author: David Diamonds

[7] - The Basics of Information Security – Understanding the Fundamentals of InfoSec in Theory and Practice, Jason Andress , Technical Editor Russ Rogers

[8] - Algorithms and Data Structures Hardcover – November 1, 1985 by Niklaus Wirth (Author)

[9] - Guide to Operations for the Tree Data Structure - By Jordan Hudgens - September 27, 2016
https://www.crondose.com/2016/09/operations-for-the-tree-data-structure/

[10] - Securing the Perimeter - Deploying Identity and Access Management with Free Open Source Software, Authors: Schwartz, Michael, Machulak, Maciej