Aspectos de Segurança em Programação com Java

Marcos Alexandre de Melo Medeiros

Paulo Sérgio Motta Pires

Departamento de Engenharia de Computação e Automação

DCA – UFRN

marcosam@info.ufrn.br, pmotta@dca.ufrn.br

Resumo

O objetivo deste trabalho é analisar problemas de segurança decorrentes da programação utilizando a linguagem JAVA. Após apresentar as características dessa linguagem, são discutidas as vulnerabilidades que afetam outras linguagens de programação, explicitando os motivos pelos quais elas não afetam JAVA.

Por fim, as vulnerabilidades próprias do ambiente Java são discutidas.

A Apresentação

- Introdução
- Linguagem Java
- Arquitetura de Segurança Java
- Problemas mais comuns nas outras linguagens
- Exemplo: Linguagem C
- Vulnerabilidades em Java
- Conclusão

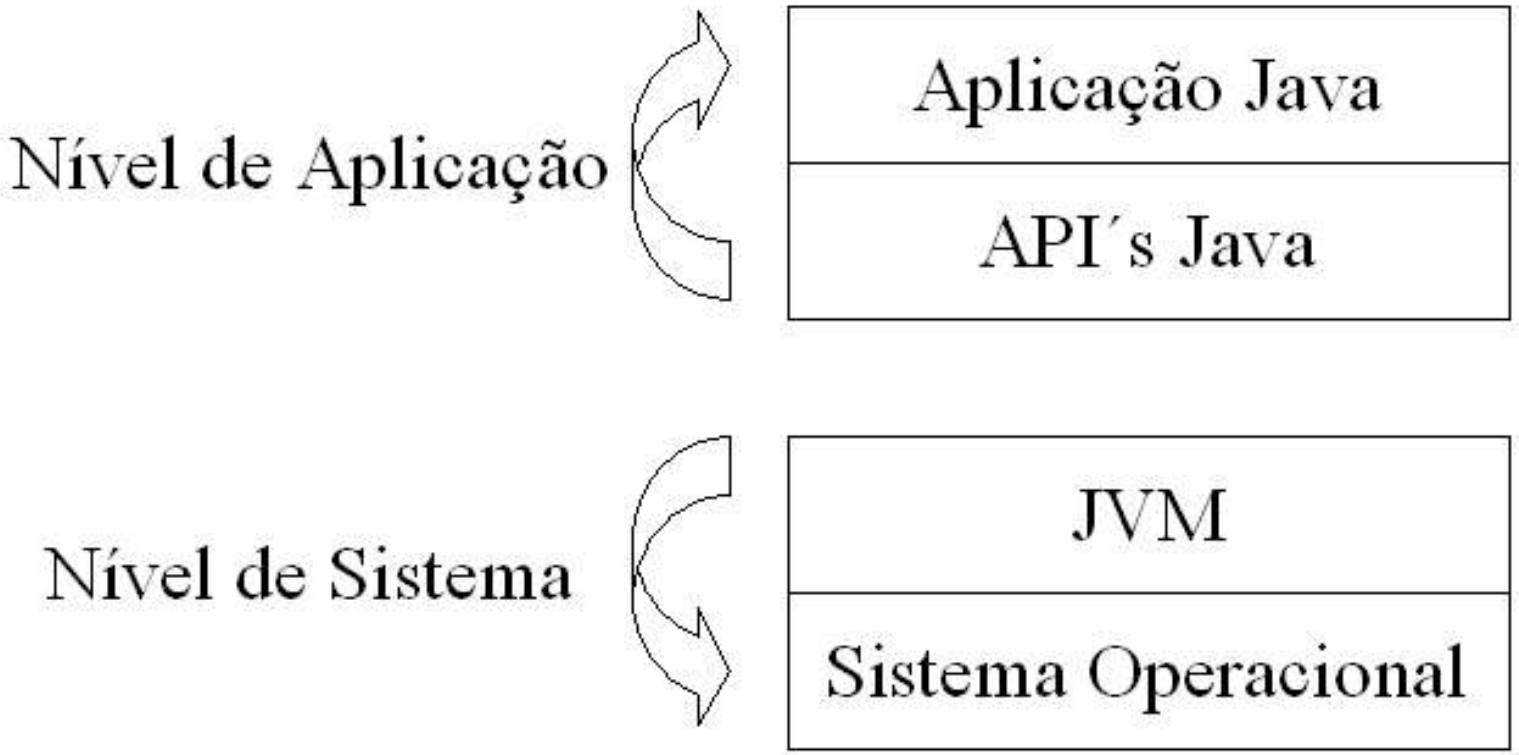
Introdução

- O que é Java?
  - Linguagem de programação baseada em C/C++
  - Criada por James Gosling
  - Resultante de uma pesquisa iniciada em 1991, mas foi anunciada formalmente em 1995
  - Orientada a objetos

Introdução

- Foi desenvolvida visando a portabilidade
- Aplicações Java rodam em diversos ambiente operacionais (computadores, celulares, etc)
- Utilizou-se o conceito de máquina virtual para isolar a aplicação do sistema operacional

Arquitetura



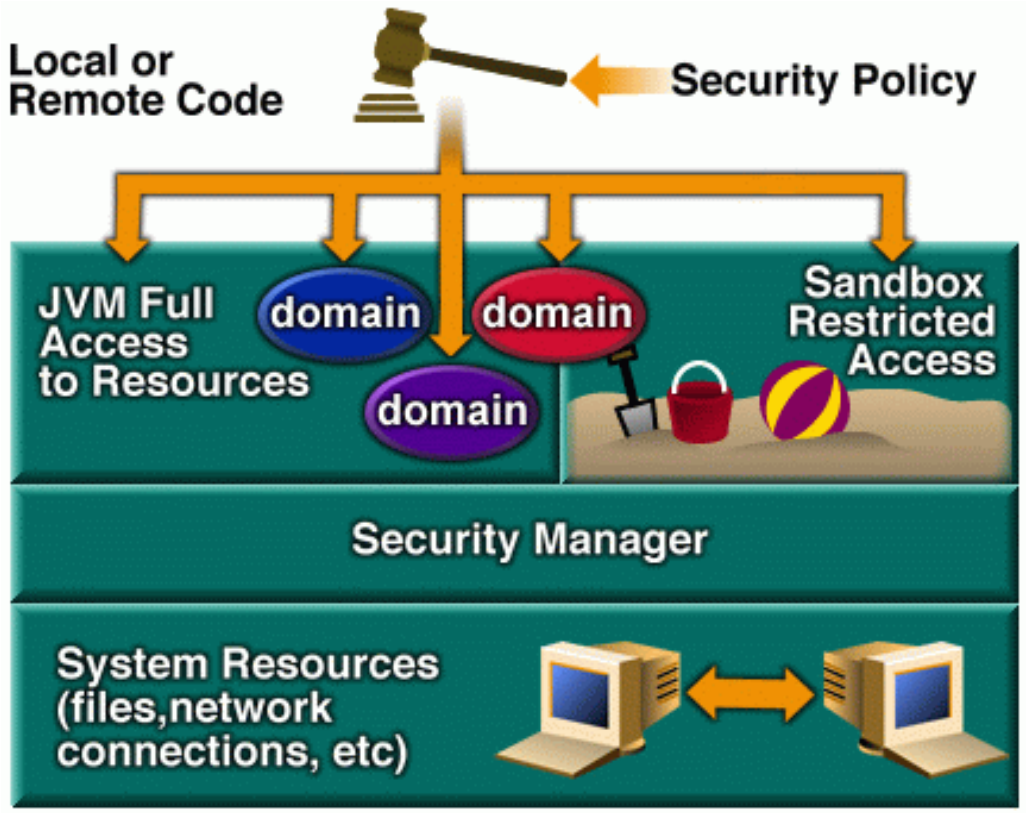
Máquina Virtual Java

- Coletor de Lixo
- Carregador de Classes
- Gerenciador de Memória
- Verificador de bytecodes
- Gerenciador de Segurança

Características da Linguagem

- Fortemente tipada
- Não permite aritmética de ponteiros
- Exige que variáveis locais sejam atribuídas antes da compilação
- Implementa verificação de limites de vetores

Arquitetura de Segurança



Sun Microsystems

- Domínios de Proteção
  - Agrupar componentes ou recursos
  - Domínios de Sistema
  - Domínios de Aplicação
- Arquivos de políticas de segurança

Problemas Mais Comuns nas Outras Linguagens

- Input Validation Error
  - Erro devido à passagem de um parâmetro em um formato não esperado e não tratado.
- Boundary Condition Error
  - Erro devido a uma tentativa de leitura/escrita em um endereço inválido ou esgotamento de um recurso do sistema.
- Buffer Overflow

Problemas Mais Comuns nas Outras Linguagens

- Access Validation Error
  - Erro devido à invocação de operações em objetos fora do domínio de acesso.
- Exceptional Condition Error
  - Erro devido ao não tratamento de uma exceção gerada por um módulo, um dispositivo ou entrada de dados do usuário.

Problemas Mais Comuns nas Outras Linguagens

- Cross Site Scripting
  - Inserção de Scripts e codigo HTML em páginas de uma rede local.
- SQL Injection
  - Inserção de comandos SQL através de concatenação de strings usando os separadores de comandos.

Exemplo: Linguagem C

- Buffer Overflow
  - Idéia
    - Extrapolar os limites de um array para executar um código malicioso
  - Teoria envolvida
    - Extrapola-se os limites de um array, alocado dinamicamente numa subrotina, que esteja na pilha de execução até atingir a posição de memória onde está armazenado o endereço de retorno dessa subrotina, que pode ser modificado para apontar para um código malicioso a ser executado.
  - Porque não afeta aplicações Java
    - Os limites dos array são sempre verificados

Exemplo: Linguagem C

- Format String
  - Idéia
    - Conseguir informações sobre os conteúdos dos endereços de memória que normalmente não são acessíveis
    - Força a queda de um programa
    - Conseguir uma “shell” de um super-usuário a partir de uma exceção provocada pela entrada de uma string mal intensionada
  - Teoria relacionada
    - Passar uma string formatada intensionamente para ler e escrever na memória.
    - Exemplo: funções xprintf e seus especificadores de formato (%s, %x, %d, %u)
  - Porque não afeta aplicações Java
    - O tratamento de exceções é bastante robusto
    - Não é possível ler fora dos limites das variáveis

Vulnerabilidades em Java

- Exposição dos Bytecodes
  - Possui grande quantidade de informações a respeito no código fonte em seu “pool” de constantes
  - FÁCIL DECOMPILAÇÃO
- Soluções
  - Adição de código não funcional ao Código Fonte
  - Ofuscação do Código Fonte
  - Criptografia dos arquivos compilados (.class)
  - Esconder os arquivos compilados (.class)

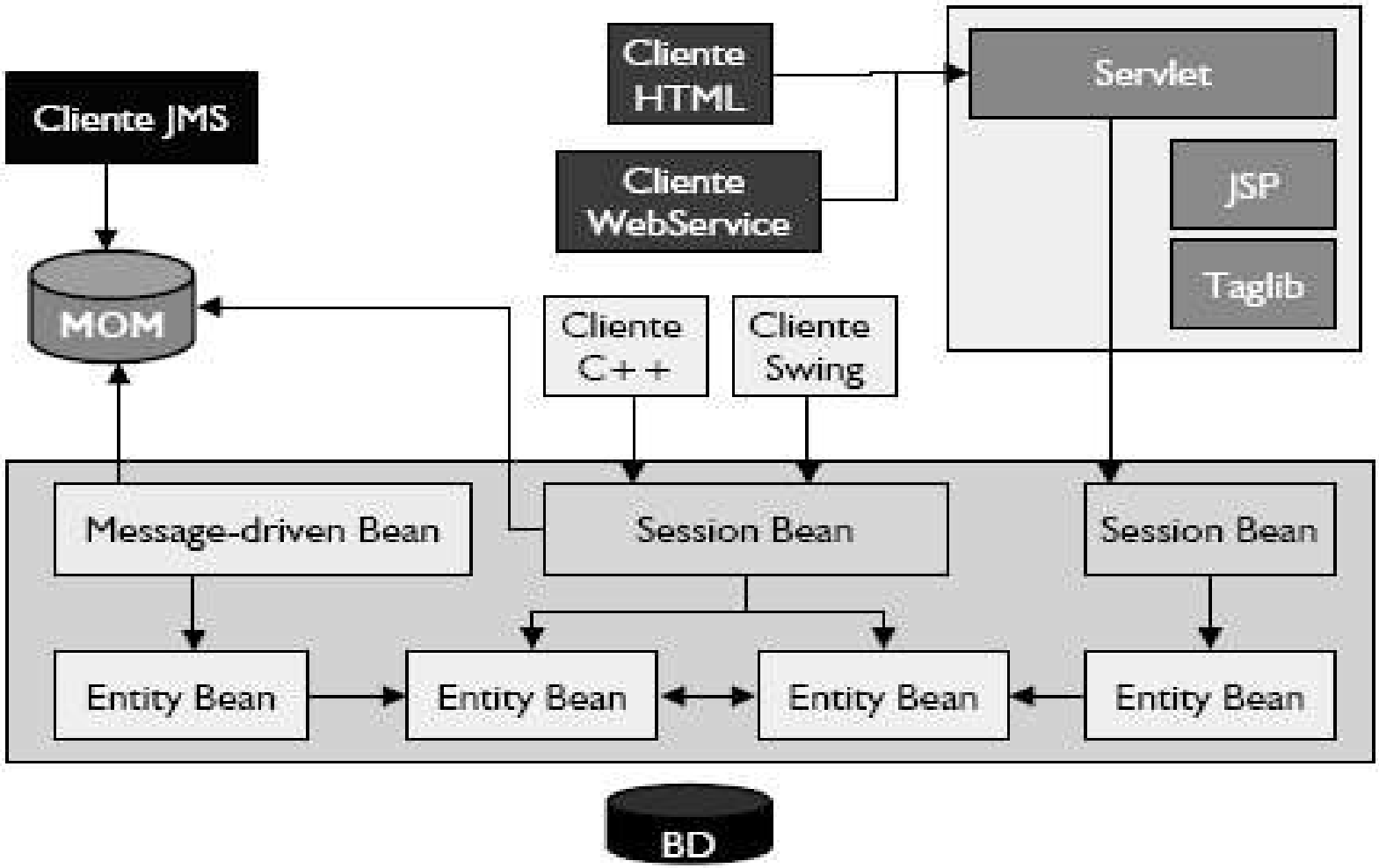
Vulnerabilidades em Java

- Objetos String e coleta de Lixo
  - São imutáveis
  - Coleta de lixo não controlável pelo usuário
  - Dados sensíveis podem flutuar na memória
- Soluções
  - Usar “array” de “char”
  - Evitar ao máximo o uso desses objetos em dados sensíveis

Vulnerabilidades em Java

- Literais de String
  - São armazenados exatamente iguais no “pool” de constantes dos arquivos compilados (.class)
  - Uma simples análise dos “bytecodes” pode fornecer dados sensíveis
- Soluções
  - Evitar o uso de literais no Código Fonte
  - Usar “array” de “char” sempre que possível

J2EE


Cliente JMS
Cliente HTML
Cliente WebService
Servlet
JSP
Taglib
MOM
Cliente C++
Cliente Swing
Message-driven Bean
Session Bean
Session Bean
Entity Bean
Entity Bean
Entity Bean
Entity Bean
BD

Conclusões

- A arquitetura da linguagem é bastante segura e permitiu isolar a aplicação do S.O.
- Java é uma linguagem mais voltada para desenvolvimento de aplicações
- As questões de segurança em Java devem ser vistas de forma diferente das demais linguagens
- O programador é o maior responsável pelas vulnerabilidades da aplicação

Contatos

Marcos Medeiros

marcosam@info.ufrn.br

Paulo Motta

pmotta@dca.ufrn.br

Departamento de Engenharia de Computação e Automação

DCA – UFRN

www.dca.ufrn.br