Detecção de ataques por ROP em tempo real assistida por *hardware*

Marcus Botacin[1], André Grégio[1,2], Paulo Lício de Geus[1]

[1]Instituto de Computação - UNICAMP
{marcus,paulo}@lasca.ic.unicamp.br

[2]Universidade Federal do Paraná (UFPR)
gregio@inf.ufpr.br

09 de Novembro de 2016

Tópicos

Tópicos

Panorama

Problema

- Injeção de Código Externo.

Mitigações

- Canários de pilha.
- Páginas não executáveis (NX/XD).
- Proteções ampliadas de memória (MPX).

Um Novo Problema

- Reúso de Código.

Tópicos

Programação Orientada a Retorno (ROP)

ROP

- Encadeamento de sequências de código legítimo (*gadgets*).

Figura : ROP.

Programação Orientada a Retorno (ROP)

ROP

- ROP permite computações arbitŕarias (Turing-completa).

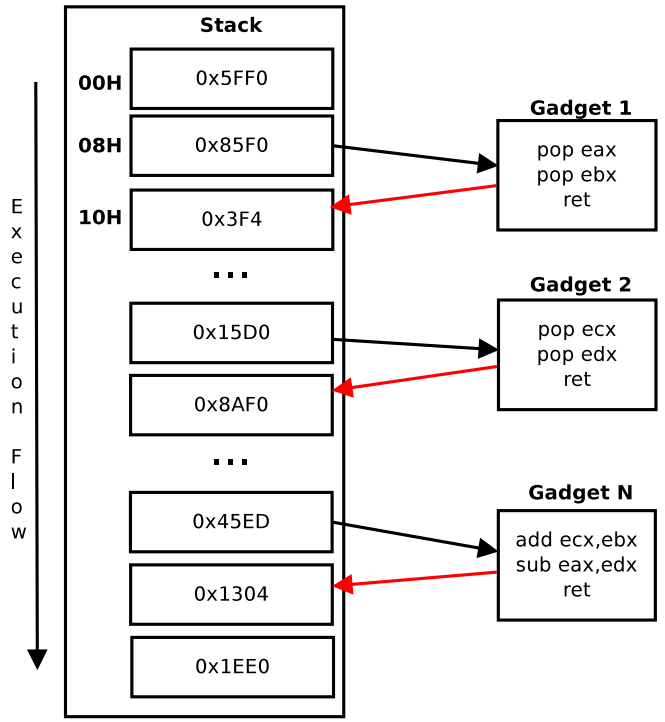

Figura : Ataque ROP.

Tópicos

Trabalhos Relacionados

Tempo de compilação

- Reescrita de código.
- Limitação: sistemas legados.

Instrumentação

- Aceita código legado.
- Efeitos Colaterais de emulação..

Trabalhos Relacionados

Reescrita de binário

- Aceita código legado.
- Sem efeitos colaterais.
- Não trata código gerado em *runtime*.

Monitoração por *hardware*

- Aceita código legado.
- Sem efeitos colaterais.
- Trata código gerado em *runtime*.
- Limitações de implementação.

Tópicos

Mecanismos

LBR vs. BTS

- Dados de *branch* fornecidos pelo processador.
- Armazenamento em registradores vs. Memória.
- *Polling* vs. Interrupção.
- Limitado vs. Ilimitado.

Dados

- Acesso em *kernel*
- `JNE,JMP,CALL,RET`

Tópicos

Implementações atuais

Kbouncer

- Limitação do uso do LBR.
- Injeção de código.
- Monitoração por processo.

ROPecker

- Limitação do uso do LBR.
- Base de código estática.

Proposta

- Sem injeção de código.
- Sem componentes estáticos.
- Monitoração de múltiplas instâncias.

Tópicos

Proposta de solução.

Funcionamento

- *Threshold* de 1 desvio.
- Interrupção para isolamento de processos.
- Introspecção de sistema para obtenção de contexto.

Arquitetura da Solução

- Cliente-Servidor / *Driver-Userland*.
- Cliente filtra processos monitorados.
- Cliente aplica políticas de integridade.

Limitações do Modelo de Ameaças

- Apenas Nível de usuário.
- Apenas instruções RET (sem *tricks*).

Proposta de solução.

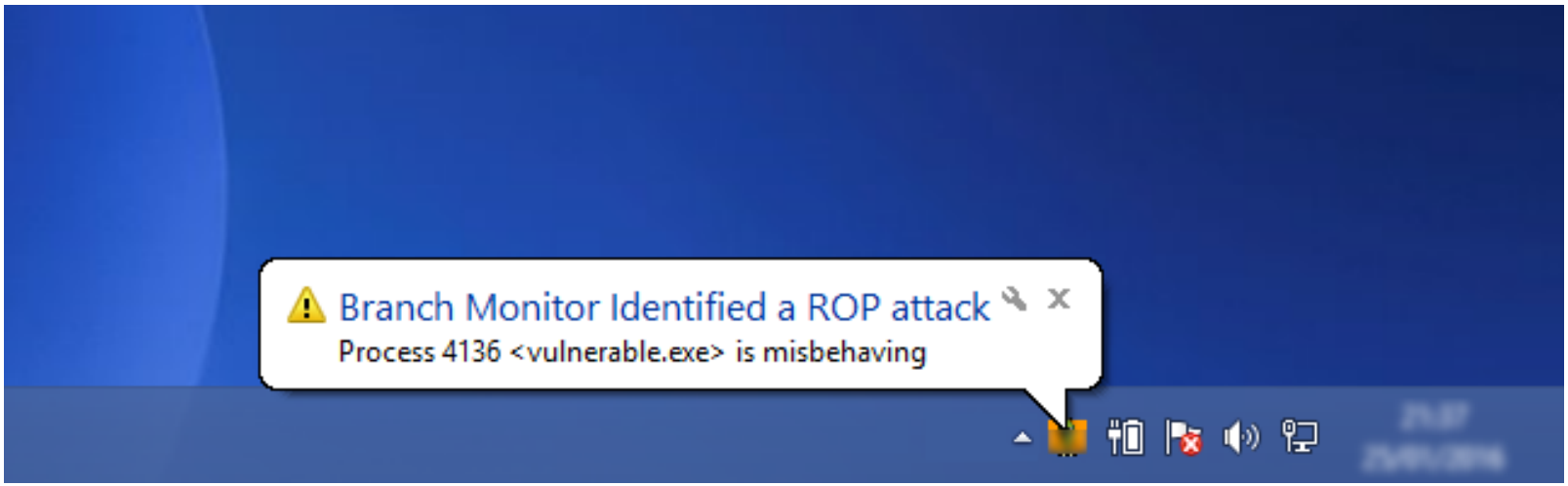

Figura : Notificação de ataque identificado emitida pelo cliente de monitoração.

Tópicos

Política CALL-RET.

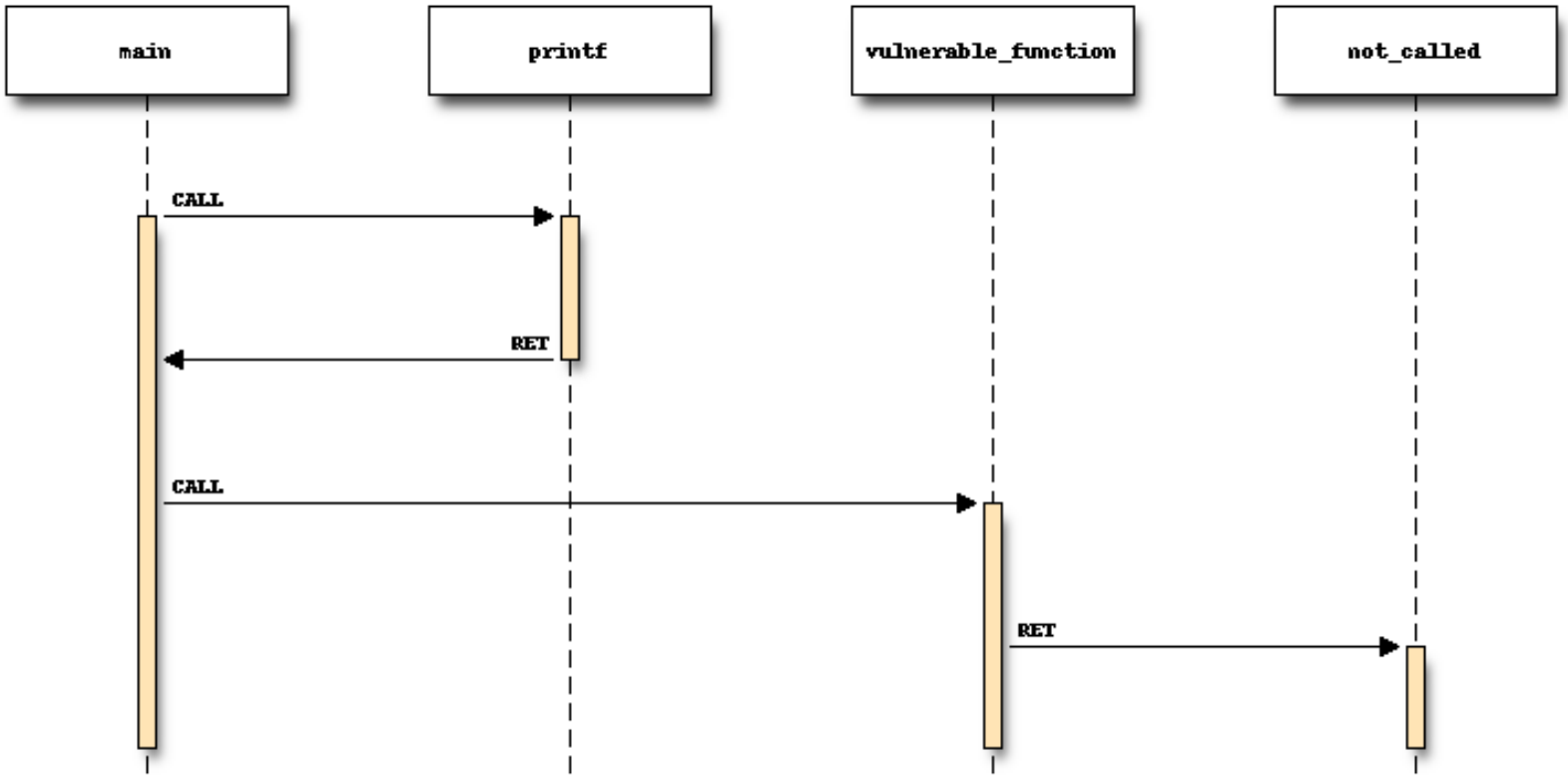

Figura : Exemplo de violação de política do tipo CALL-RET.

Como Identificar as instruções ?.

Tabela : *Opcodes* de instruções do tipo `CALL`.

Opcode	Mnemonic	Opcode	Mnemonic
0xE8	CALL rel16	0x9A	CALL ptr16:16
0xE8	CALL rel32	0x9A	CALL ptr16:32
0xFF	CALL r/m16	0xFF	CALL m16:16
0xFF	CALL r/m32	0xFF	CALL m16:32

Tabela : *Opcodes* de instruções do tipo `RET`.

Opcode	C3	CB	C2 iw	CA iw
Mnemonic	RET	RET	RET imm16	RET imm16

Política CALL-RET.

Listagem 1 : Correspondências "CALL-RET" indicando fluxo de execução íntegro.

```
1 PID 3140 FROM  6b8e7f17 INSTR e8 - CALL
2 PID 3140 TO  6b9d90c1 INSTR c3 - RET
3 PID 4196 FROM  77b2ce8e INSTR e8 - CALL
4 PID 4196 TO  77aa591e INSTR c2 - RET
5 PID 2532 FROM  3de50b6c INSTR e8 - CALL
6 PID 2532 TO  40714979 INSTR c2 - RET
```

Política do comprimento do *gadget*.

LBR Stack

	Branch	Target	
00	73802745	738028D7	
01	05F015CA	05F00E17	
02	7C348B06	7C34A028	G01
03	7C34A02A	7C34252C	G02
04	7C34252D	7C36C55A	G03
05	7C36C55B	7C345249	G04
06	7C34524A	7C3411C0	G05
07	7C3411C1	7C34B8D7	G06
08	7C34B8D8	7C366FA6	G07
09	7C366FA7	7C3762FB	G08
10	7C3762FC	7C378C81	G09
11	7C378C84	7C346C0B	G10
12	7C346C0B	7C3415A2	G11
13	7C3415A2	74F64347	
14	74F64908	752AD0A1	
15	752D6FC8	752AD0AD	

Figura : Exploit ROP (Fonte: Kbouncer).

Política do comprimento do *gadget*.

Listagem 2 : Comprimento dos blocos em programas legítimos (em número de instruções).

```
1 PID 3820 FROM 5f0dea04 TO 5f0deb30 INSTR 15
2 PID 3820 FROM 5f0deb4a TO 5f0deb53 INSTR 21
3 PID 3820 FROM 5f0dea0e TO 5f0dea17 INSTR 19
```

Política do comprimento da frequência de desvios.

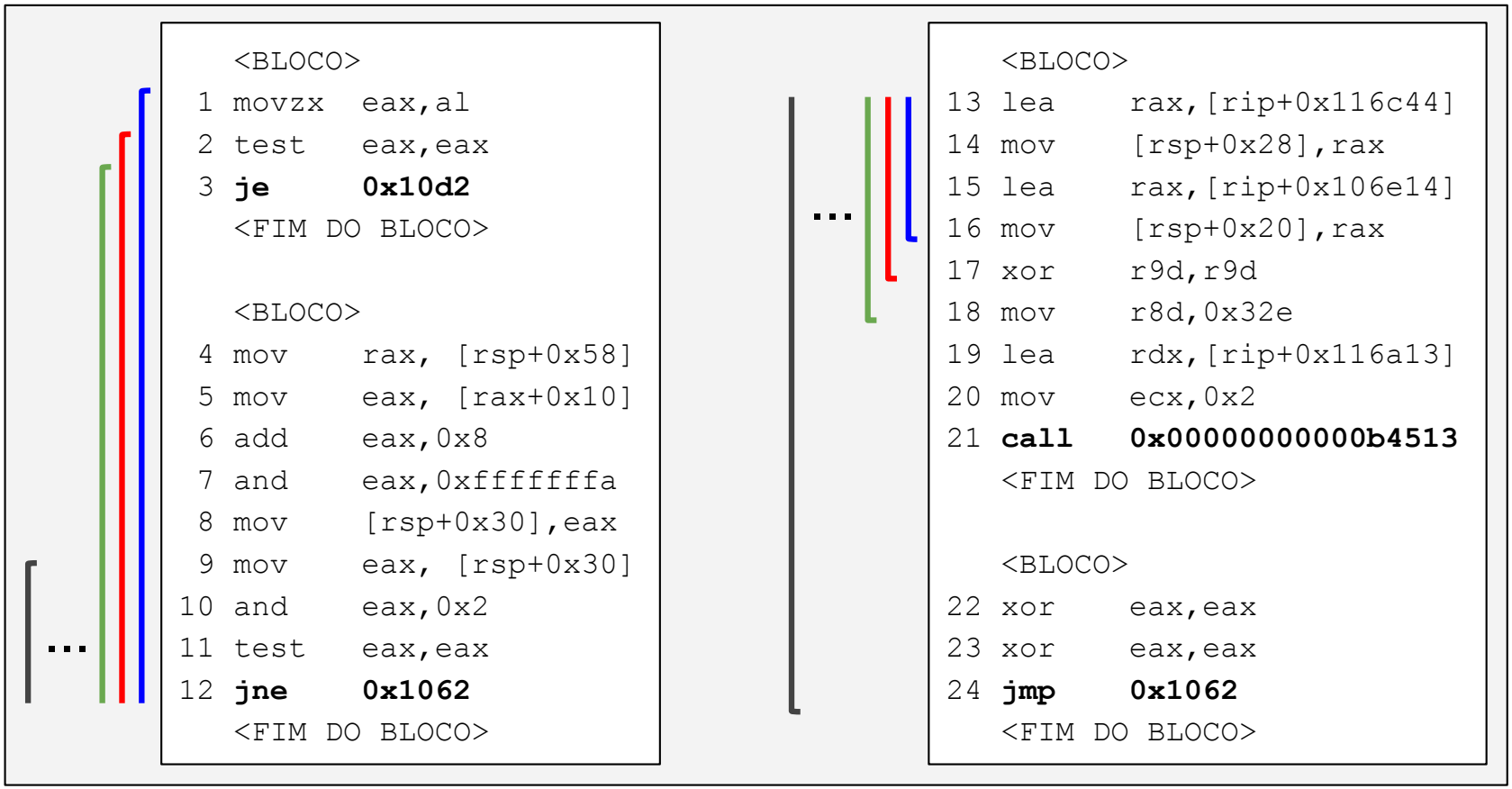

Figura : Blocos com instruções de desvios nas extremidades (em negrito) e janela deslizante de 16 instruções.

Como Obter Instruções a partir de *branches* ?

Listagem 3 : Exemplo de buffer de instruções obtido a partir dos endereços fornecidos pelo mecanismo BTS.

```
1 \xff\x15\x0a\x11\x00\x00\x48\x8d\x0d\x9f\x11\x00\
      x00
```

Listagem 4 : Conversão das instruções do buffer para opcodes.

```
1 0x1000 (size=6)   call    QWORD PTR [rip+0x110a]
2 0x1006 (size=7)   lea     rcx,[rip+0x119f]
```

Tópicos

Um *exploit* real

Tabela : Janela de instruções de desvio contendo 2 gadgets de 2 bytes e 2 instruções.

FROM	TO
—-	0x7c346c0a
0x7c346c0b	0x7c37a140
0x7c37a141	—-

Um *exploit* real

Listagem 5 : Código legítimo (alinhado) contendo uma sequência de bytes que pode ser abusada em um ataque *(gadget).*

```
1 7c346c08: f2 0f 58 c3          addsd  %xmm3,%xmm0
2 7c346c0c: 66 0f 13 44 24 04    movlpd %xmm0,0x4(%esp)
```

Listagem 6 : Código desalinhado contendo o gadget realmente executado.

```
1 0x1000 (size=1)  pop    rax
2 0x1001 (size=1)  ret
```

Um *exploit* real

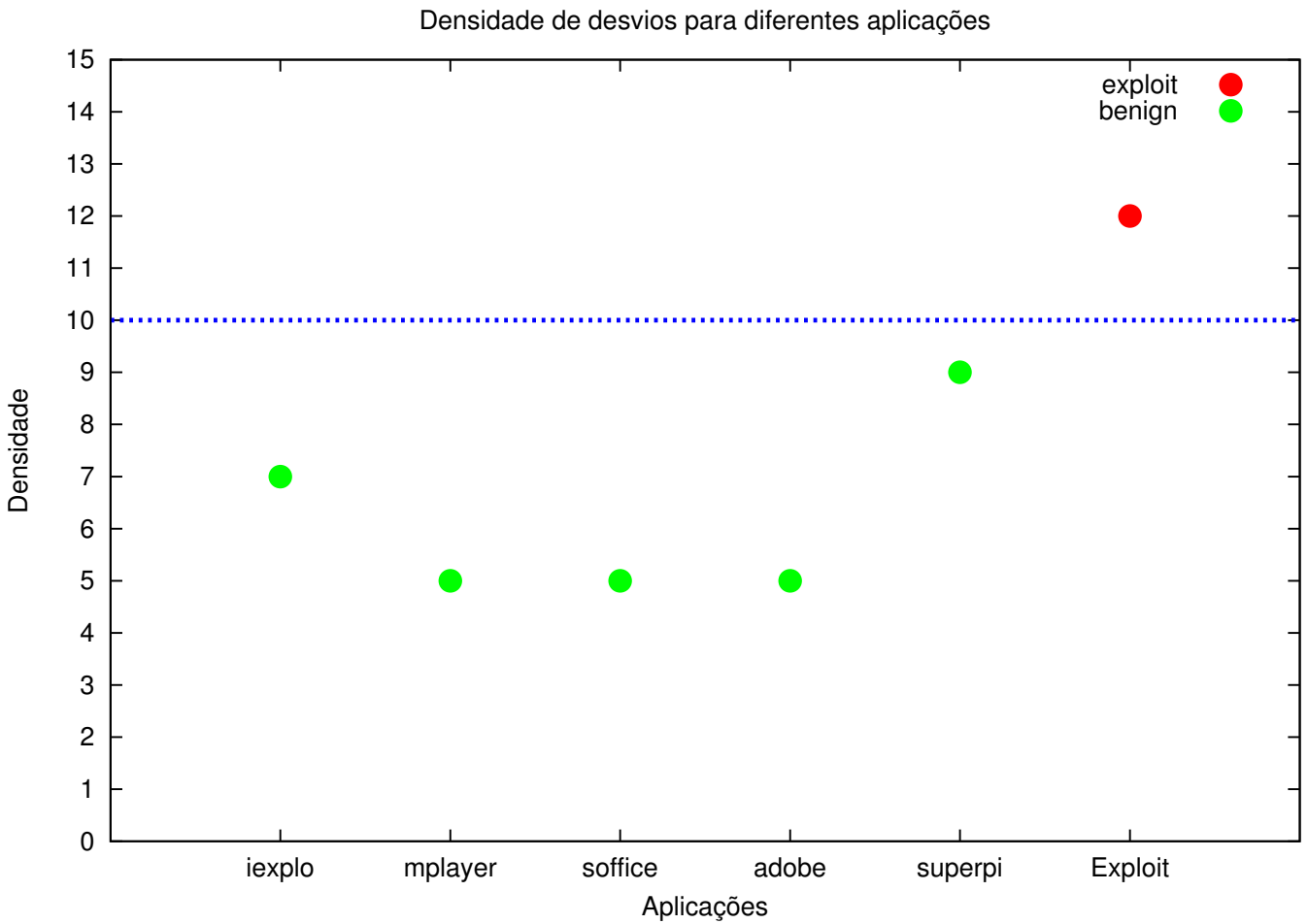

Figura : Densidade de desvios em diferentes aplicações.

Overhead

Ativação do Monitor

- 1%.

Coleta de dados

- 14% a 26%

Disassembly online

- 40%.

Tópicos

Solução Definitiva ?

Weird Machines

- *Page Fault Computing*
- *ELF Metadata Computing*
- *Look up Tables Computing*

Análise dos Resultados

Sumarização

- Ataques de Reuso cada vez mais frequente..
- Soluções exigem recompilação ou código estático.
- Soluções por *hardware* tem limitações de implementação.
- Uso de BTS e *disassembly* dinâmico é opção.
- Diferentes heurísticas para diferentes construções.
- Não temos solução definitiva.
- Soluções de monitoramento sempre serão necessárias.
- Podemos avançar estado-da-arte das implementações.

Limitações e Trabalhos Futuros

Limitações

- Análise em *kernel*.
- *Jump Oriented Programming* (JOP).
- *Loop Oriented Programming* (LOP).
- *POP+RET* como substituto a *CALL*.

Trabalhos Futuros

- Expansão das análises.

Tópicos

Conclusões

Conclusões

- Solução Baseada em BTS ao invés do LBR.
- Sem injeção de código.
- Monitoramento *system wide*
- *Disassembly* dinâmico.

Agradecimentos

- CNPq, pelo financiamento via Proj. MCTI/CNPq/Universal-A edital 14/2014 (Processo 444487/2014-0)
- CAPES, pelo financiamento via Proj. FORTE - Forense Digital Tempestiva e Eficiente (Processo: 23038.007604/2014-69).
- Instituto de Computação/Unicamp
- Departamento de Informática/UFPR

Contato:

marcus@lasca.ic.unicamp.br
paulo@lasca.ic.unicamp.br
gregio@inf.ufpr.br