Segurança da Computação

Matheus Venturyne Xavier Ferreira
Universidade Federal de Itajubá
19 de Outubro de 2015

Tópicos

- Segurança de computadores (sistemas)
  - Memória (e.g. Heap, Stack)
  - Control Hijacking
- Segurança de redes
- Segurança web

Agradecimentos

- Projeto 1 e 2 de UC San Diego CSE 127, Computer Security. Agradecimentos para o professor Hovav Shacham.
- Projeto 3 de Stanford CS 155, Computer and Network Security. Agradecimentos para os professores Dan Boneh, John Mitchell, Collin Jackson.

Projetos

- Projeto 1: fluxo de controle
  - Target 1: warmup. Buffer overflow.
  - Target 2: Buffer overflow.
  - Target 3: arithmetic overflow; buffer overflow
  - Target 4: double free (malloc/free)
  - Target 5: printf
- Projeto 2:
  - Heap spray: ataque probabilístico contra técnicas de segurança de memória.

Projeto

- Projeto 3: Web
  - Ataque 1: roubar cookie
  - Ataque 2: cross-site request forgery
  - Ataque 3: injeção SQL
  - Ataque 4: roubar senha
  - Ataque 5: profile worm

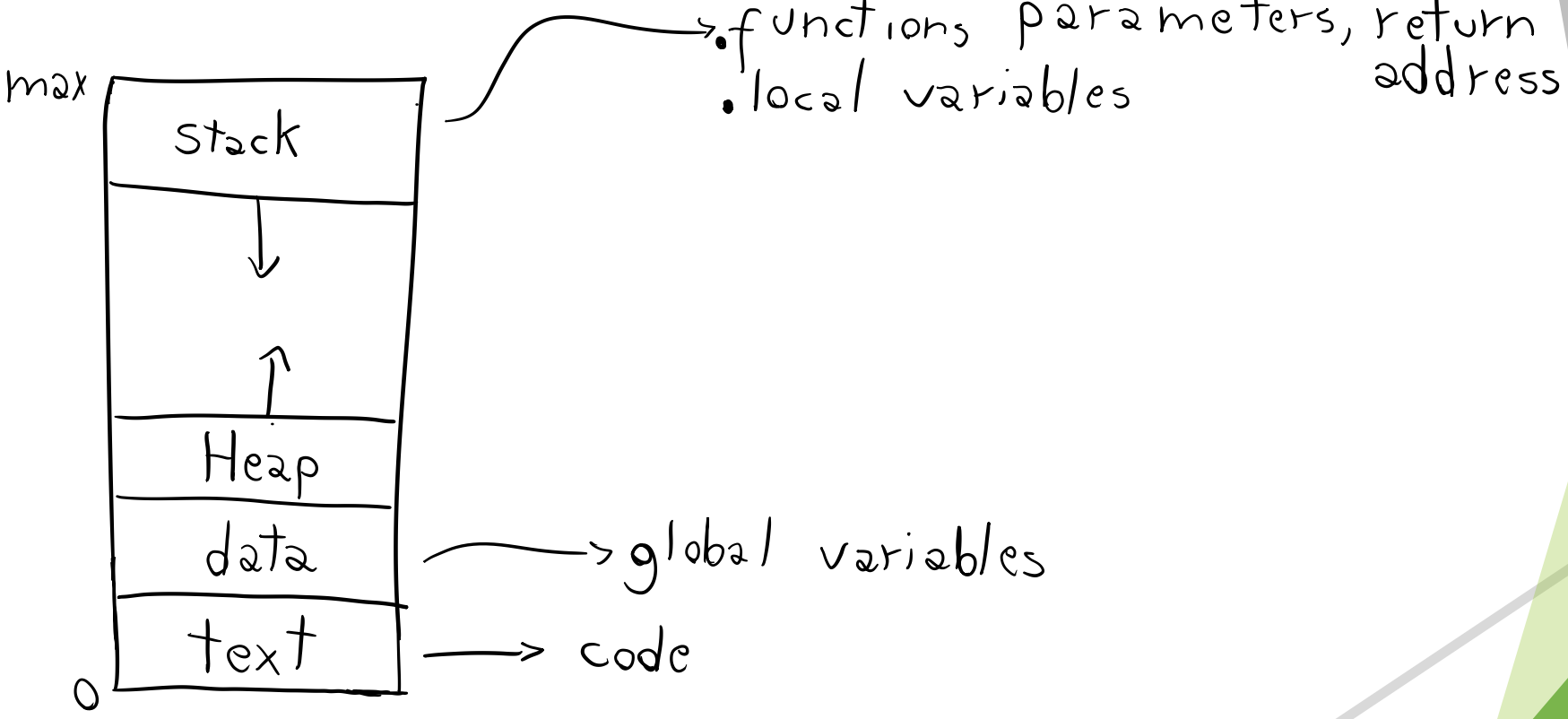
Material

- Notebook com Virtualbox
- Papel, lápis/caneta
- Material http://1drv.ms/1GbQ4Mu
  - Melhores artigos:
    - Reflection on Trusting Trust (Ken Thompson)
    - This World of Ours
    - Beware of Finer-Grained Origins
  - Leituras recomendadas para os projetos:
    - Smashing the Stack for Fun and Profit
    - Understanding the Heap by Breaking it
    - Exploiting Format String Vulnerabilities

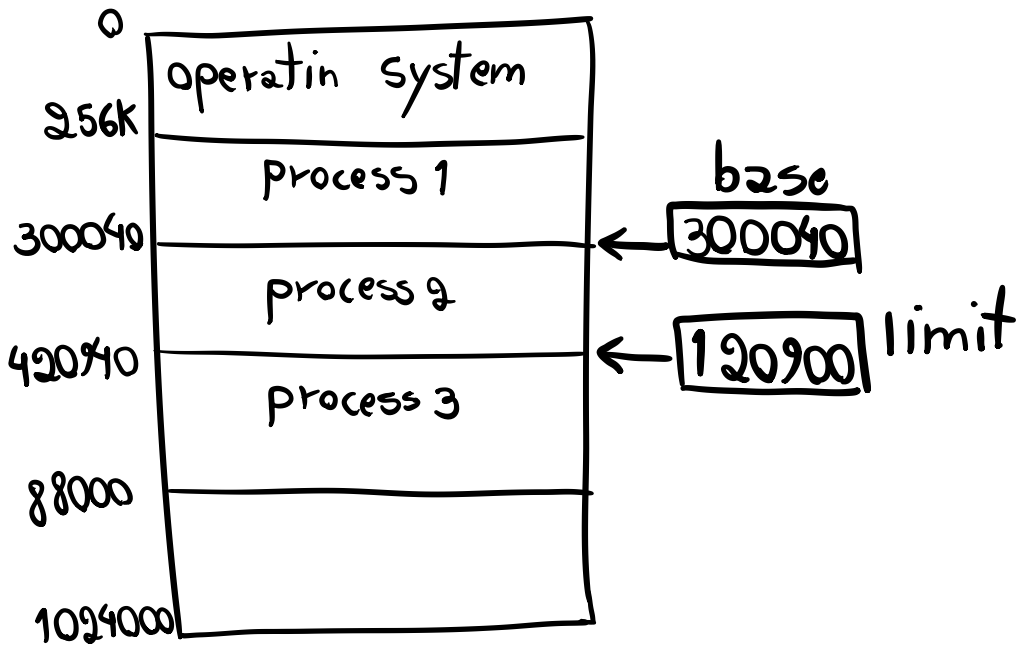
Segurança da Computação

- Alcançar funcionalidade na presença de um ataque
- Porque estudar?
  - Mais informação e funcionalidades estão se tornando online
  - Rede mundial de computadores (Internet)
  - Ataques podem ser automatizados e vendidos

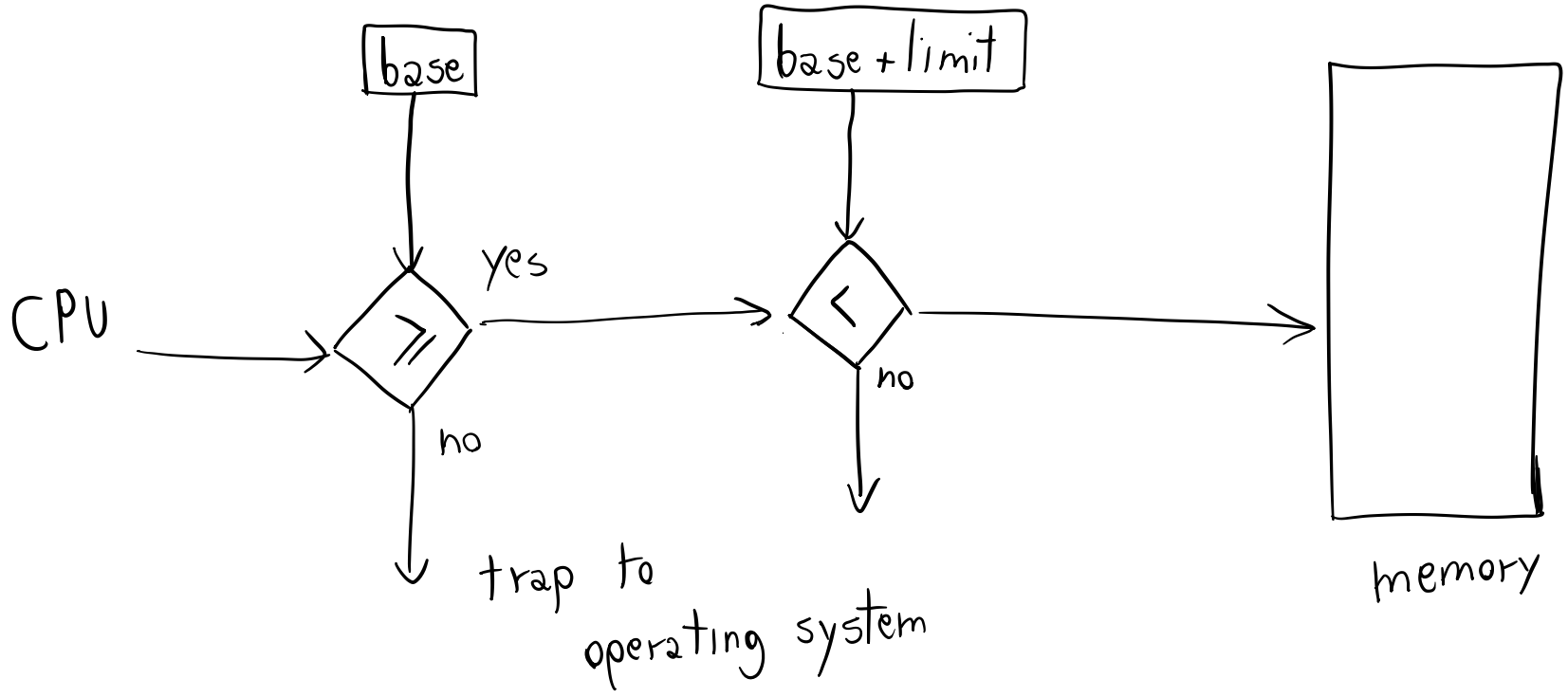
Processo



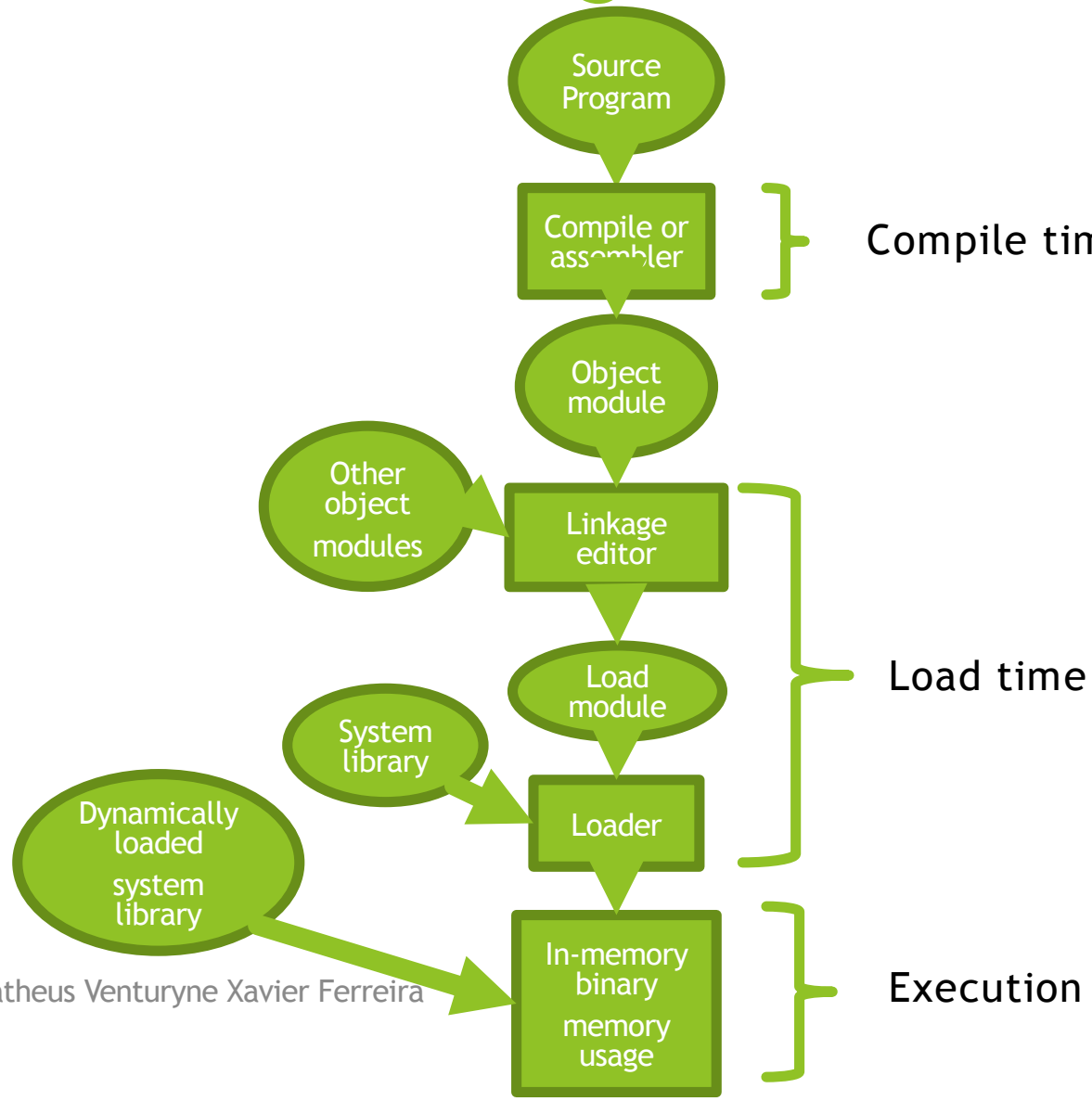
Memoria Principal



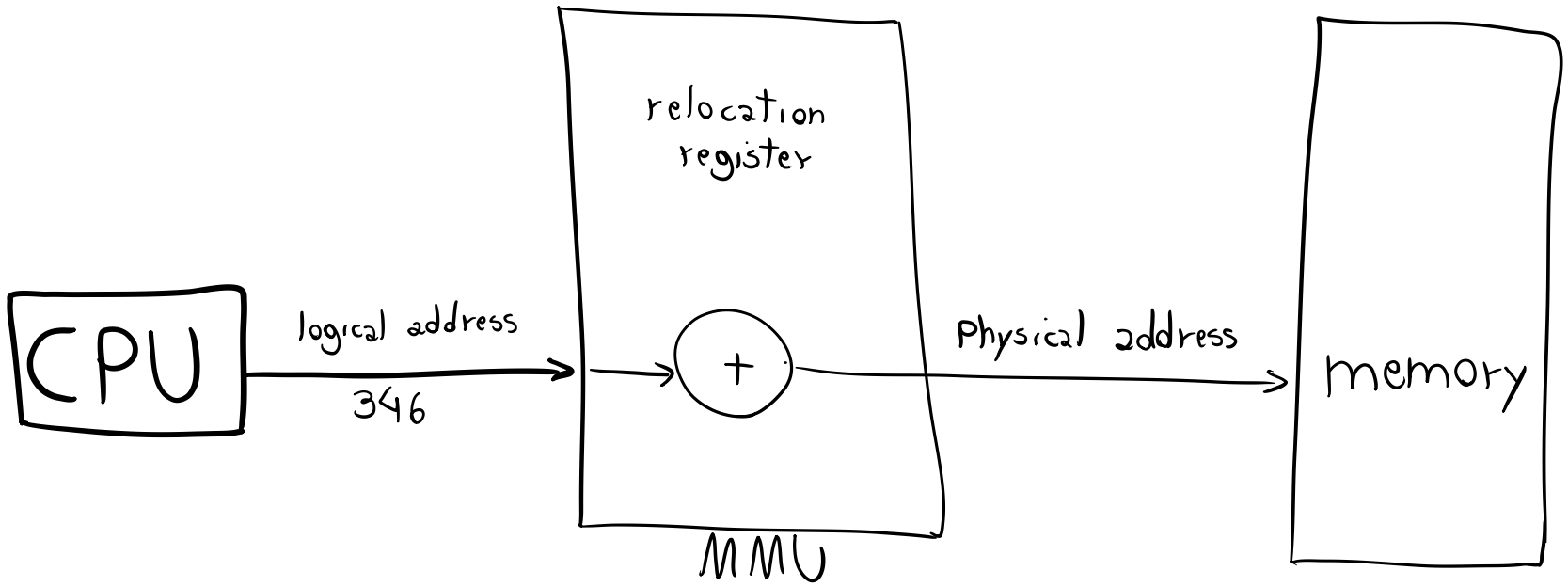
Memoria Principal



Processamento Programa do Usuário



Memoria virtual



Threat Model

- Identificar o Sistema que deve ser protegido (e.g. votação)
- Identificar os possíveis invasores (e.g. eleitor, gerente)
- Listar os objetivos
- Modelo de referencia: os componentes do Sistema e como estes interagem.
  - Estágios em que o Sistema se encontra e pode ser vítima de um ataque.
- Tipos de ataque
  - Detectável vs indetectável (legitimidade)
  - Recuperável vs irrecuperável
  - Prevenção vs detecção (custo)
  - Total vs parcial
  - Casual vs sofisticado

Tamper Sealing

- Forma de detectar um ataque
  - Tamper Resistance
    - Aumentar o tempo necessário para executar o ataque.
  - Tamper Evidence

Cavalo de Troia

- Gregos invadiram Troia com um cavalo de madeira oco
- Programa que contem código malicioso Escondido
  - Usuário acha que programa contem algo útil
  - Mas de fato, ele também faz algo prejudicial
- Programa roda como um processo no domínio do usuário
  - Pode causar dano ao ambiente do usuário
  - Pode causar dano com o nome do usuário

Trap Door

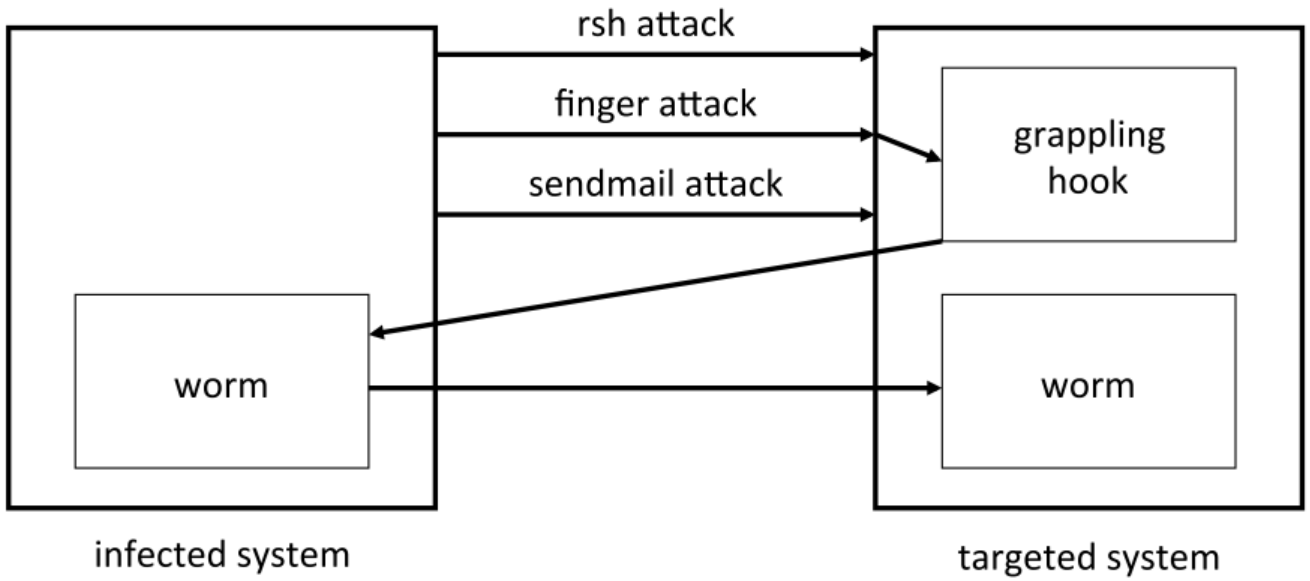
- Ponto de acesso secreto no programa
- Designer desenvolve o programa para alguém
- Uma vez carregado no Sistema, design pode acessar
- Considere se a trap door é adicionada no compilador
  - Compilador pode adicionar trap doors em programas
  - Design do compilador pode então acessar
  - Não se pode dizer do ponto de vista do código fonte do programa
  - Mesmo que um novo compilador é escrito, ele deve ser compilado!

Vírus

- Código adicionado a programa legítimo
- Quando o programa roda, o vírus roda
  - Causa dano
  - Se espalha, se fixando em outros programas
- Desinfecção
  - Checar se programas parecem normais (não modificados)
  - Checar por padrões de vírus em programas

Internet Worm

- Programa que se copia através da rede
- Internet worm (Nov 2, 1988)



Ken Thompson (Reflection on Trusting Trust)

- Passo 1: adicionar o seguinte código para login src

```
if(passwd is “let me in”)
    authorize login
```

Ken Thompson (Reflection on Trusting Trust)

- Passo 2: modificar o src do compilador para adicionar

```
If(prog being compiled is “login”)
    insert snippet (Passo 1) before compiling
```

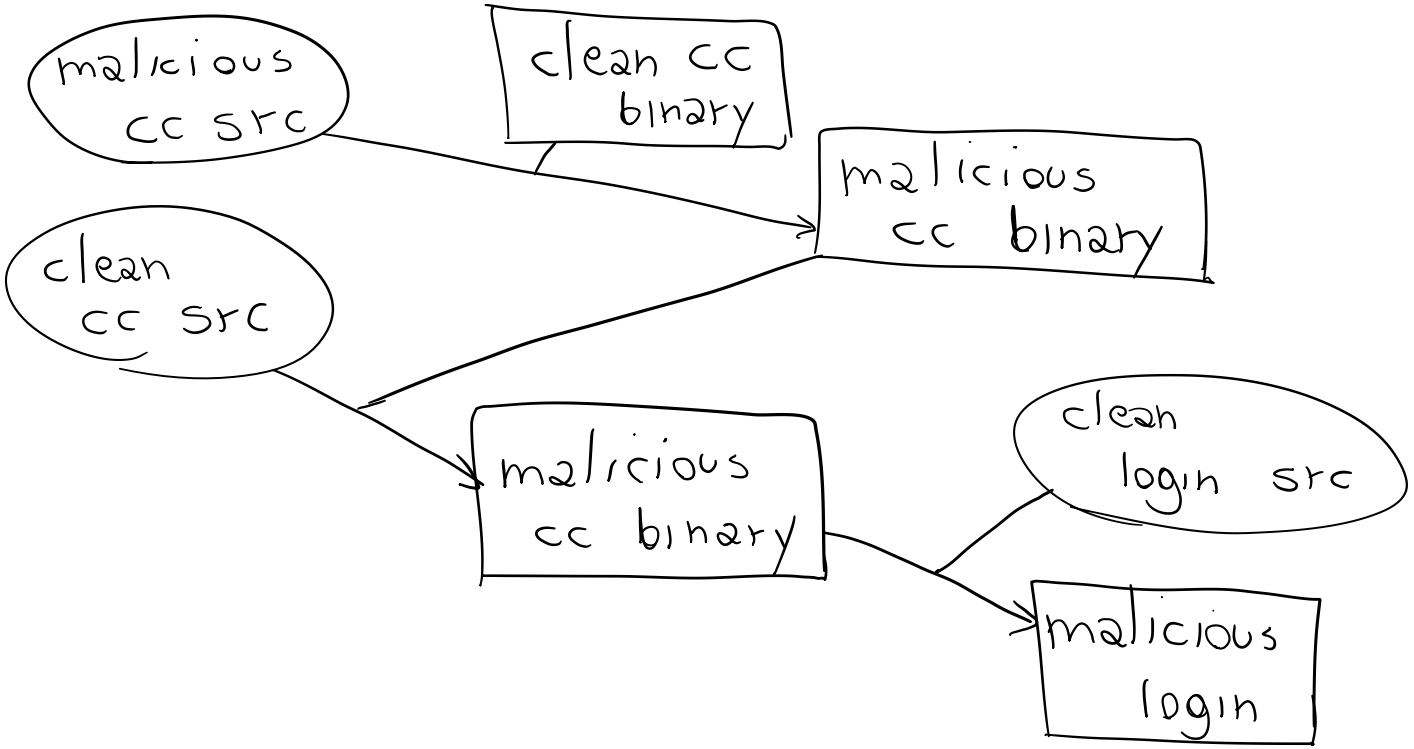
- Passo 3: modificar código fonte do compilador para adicionar

```
if(prog being compiled is “cc”)
    insert snippets (Passo 2, 3) if not already there
before compiling
```

Ken Thompson (Reflection on Trusting Trust)

- Compilar malicioso cc
- Compilar cc, login com backdoor cc
- Publicar cc e login com backdoor
- Publicar código fonte cc e login limpos

Ken Thompson (Reflection on Trusting Trust)

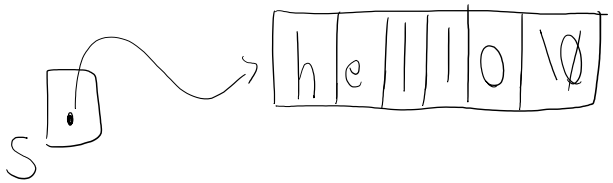


Buffer Overflow e Segurança da Memória

- C possui strings?

char *s = "hello"	char buf[6] = "hello" (know at compile time)

buf syntactic sugar to &buf[0]



Como programadores C gerenciam strings?

- Aloca buffer suficientemente grande

  char buf[800]

- Passa o endereço alocado para o buffer para funções manipuladoras de strings

  Strcpy(buf, “hello”) : dst, src

  Strcpy(char *dst, char *src) {

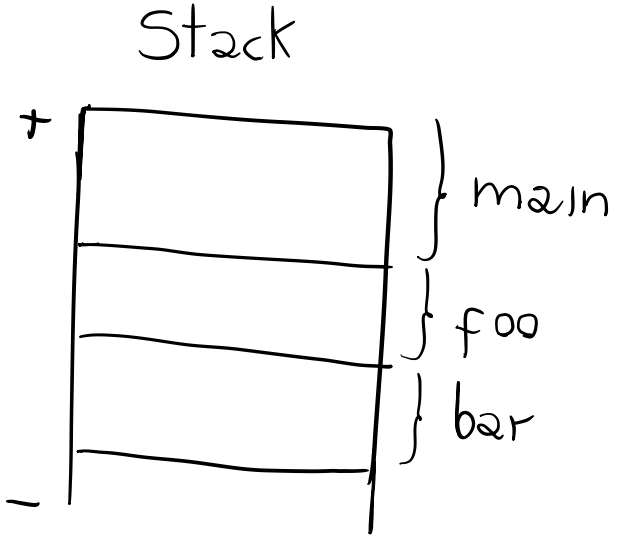
  ```
  while(*d++=*s++)
  ```

  }

- Mas o que acontece se a string é maior que o buffer?

Stack

- Estrutura de dados utilizada por um processo para gerenciar suas chamadas de função.
- Cada chamada de função recebe um activation record.

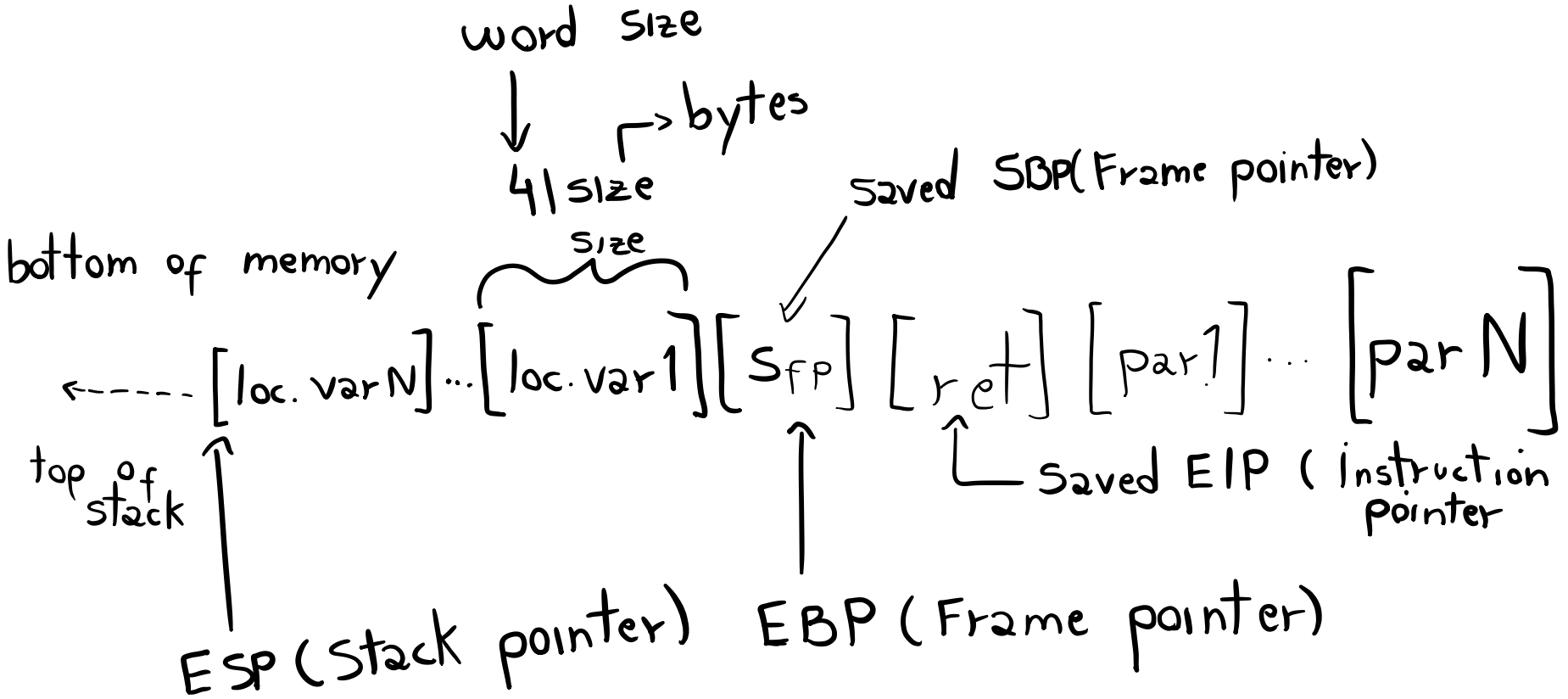


main calls foo

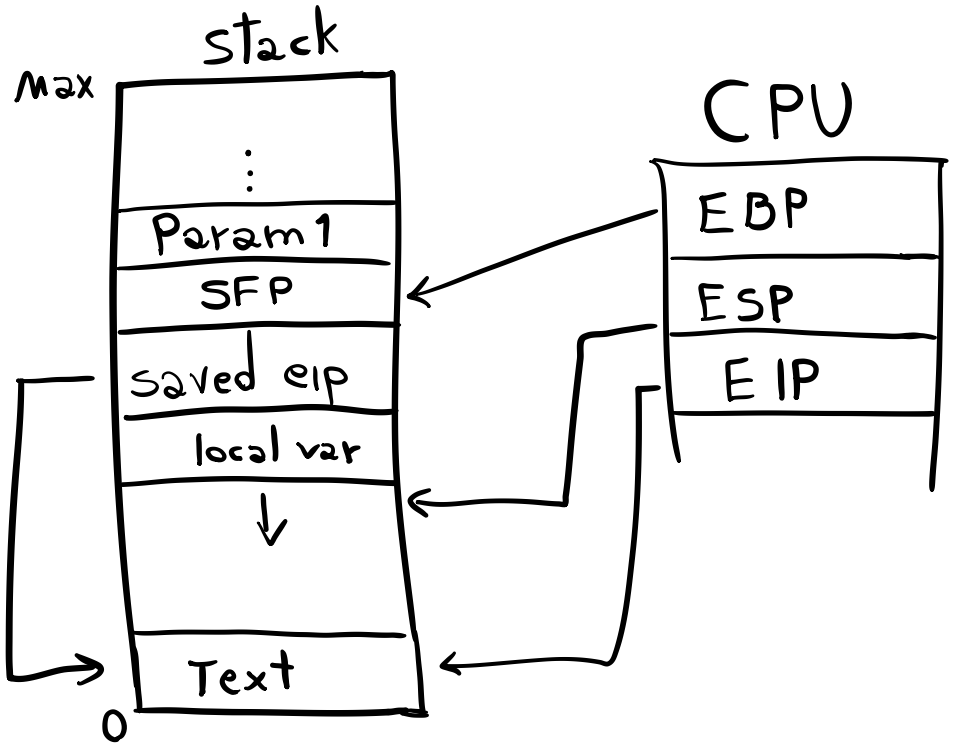
foo calls bar

$\overrightarrow{esp}$: extended stack pointer

Activation Record



Control Hijacking



Referências

- S. Inguva et al.: Source Code Review of the Hart InterCivic Voting System (Chapter 3 only)
- K. Thompson: Reflections on Trusting Trust
- A. One: Smashing the Stack for Fun and Profit
- Silberschatz; Galvin; Gagne: Operating Systems Concepts, 9th Ed, 2013
- J. Pasquale: CSE 120: Principles of Operating Systems – Lecture 13: Protection and Security. University of California, San Diego, 2014