# Towards Efficient Heap Overflow Discovery

Cristiano Matsui[1], Kallil Miguel[1], Rodrigo Anater[1]

Universidade Tecnológica Federal do Paraná

[1]Departamento Acadêmico de Informática

# Sumário

# Introdução

Esssa apresentação baseia-se no artigo *Towards Efficient Heap Overflow Discovery*, publicado pelos autores:

- Xiangkun Jia,
- Chao Zhang,
- Purui Su,
- Yi Yang,
- Huafeng Huanh,
- Dengguo Feng.

# Introdução

- Uma das principais vulnerabilidades exploradas em sistemas operacionais são causadas por corrupção de memória, na qual os conteúdos de um bloco de memória são modificados, causando posteriormente uma falha no programa ou um comportamento inesperado do mesmo.
- Em torno de 10% das falhas em aplicações causadas em sistemas Windows advém da corrupção da memória, seja ela acidental ou não.
- Uma das principais técnicas de corrupção de memória e também uma das mais populares é a técnica de *Heap Overflow*.

# Heap x Stack

- **Heap**
  - Acessada por ponteiros
  - *malloc()* ou *calloc()* para a alocação
  - Pode ser acessada por qualquer função, qualquer programa
- **Stack**
  - Estrutura do tipo LIFO
  - Armazena variáveis temporárias
  - Não requer alocação manual

# Heap x Stack

- O Heap é uma região de memória reservada pelo sistema operacional com o objetivo de armazenar os dados referentes a variáveis alocadas dinamicamente. Como por exemplo utilizando a função *malloc*.
- A alocação dinâmica tem como vantagem poder alocar espaço de memória durante a execução da aplicação, já que não se pode prever qual a quantidade de memória que será utilizada, e também permite a liberação de memória ao chamar a função *free*.

# Heap x Stack

- A Stack é uma região especial de memória administrada e otimizada pelo processador
- Uso de push/pop para armazenar e remover dados da stack
- Existe um limite de tamanho de variável que pode ser armazenado no stack (não existe este limite para o heap)

# Heap Overflow

- A memória é alocada de forma dinâmica no heap pela aplicação que está sendo executada
- Input malicioso procura escrever em uma posição indevida (sobrescrevendo o próximo frame pointer, por exemplo)
- Quando essa ocorrência é realizada, o objetivo é normalmente executar um determinado código ou alterar a execução de uma aplicação
- Exemplo: *Off-by-one attack*

# Heap Overflow: Proteção

- Proteção em tempo real
  - Diehard
  - Dieharder
  - HeapTherapy
  - HeapSentry
- Overhead em tempo de execução
- Negação de serviço

# Detecção e Prevenção

- Existem ferramentas para a detecção de vulnerabilidades de heap overflow:
  - Online: AddressSanitizer, Fuzzers (acesso ao heap)
  - Offline: DIODE, Dowser, BORG (alocação no heap)
- Problema: raíz do heap overflow (alocação **E** acesso)
- Inconsistências espaciais entre operações de alocação e acesso

# Detecção e Prevenção: Soluções atuais

- Se baseiam em gerar um grande número de casos de teste
- Passiva
- Não detecta todas as falhas
  - Falhas que estão no mesmo caminho
- Solução: HOTracer

# HOTracer

- Possui como principal objetivo rastrear as operações realizadas no heap (acesso e alocação), e produzir como saída casos em que a entrada é uma prova concreta resultante de um heap overflow.

- Funcionamento:
  - Pré-processamento das entradas;
  - Identificar operações no heap a partir dos caminhos de execução;
  - Rastrear atributos espaciais e atributos problemáticos;
  - Construir condições de vulnerabilidade;
  - Comparar as condições com caminhos de restrições;
  - Gerar saídas como provas concretas de vulnerabilidade;

# HOTracer

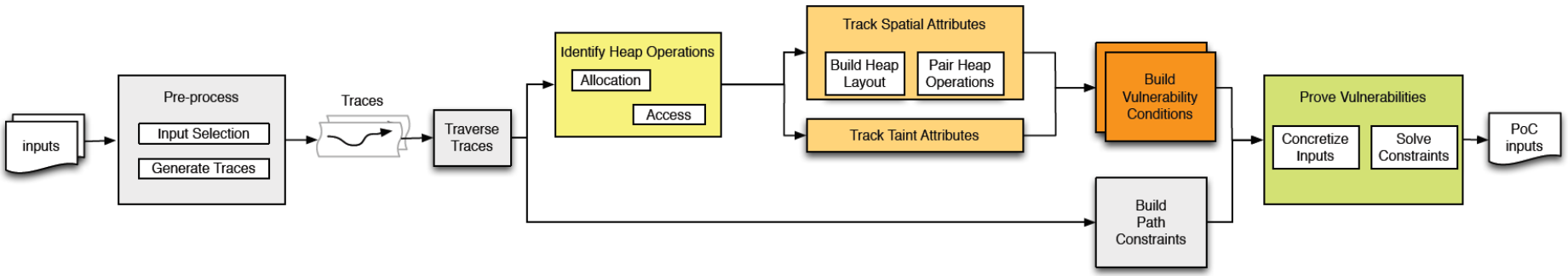


Figura: Diagrama de blocos do HOTracer

# Resultados

- Protótipo baseado em QEMU
- Testado em 17 aplicações reais
- Encontradas 47 vulnerabilidades previamente desconhecidas

# Resultados

| ID (count) | Application | version | input | bug status |
|---|---|---|---|---|
| new (1) | Feiq | 3.0.0.2 | tcp | reported |
| new (1) | WMPlayer | 12.0.7601 | mp4 | reported |
| new (3) | VLC | 2.2.1 | mp4 | fixed |
| new (1) | VLC | 2.2.4 | mp4 | reported |
| new (2) | iTunes | 12.4.3.1 | mp4 | reviewing |
| new (1) | ffmpeg | c0cb53c | mp4 | CVE |
| new (6) | QQPlayer | 3.9(936) | mp4 | rewarded |
| new (1) | QQMusic | 11.5 | m4a | rewarded |
| new (1) | BaiduPlayer | 5.2.1.3 | mp4 | reviewing |
| new (2) | RealPlayer | 16.0.6.2 | mp4 | CVE |
| new (1) | MPlayer | r37802 | mp4 | reported |
| new (3) | KMPlayer | 3.9.1.138 | mp4 | fixed |
| new (4) | KMPlayer | 4.1.1.5 | mp4 | reported |
| new (7) | Potplayer | 1.6.60136 | mp4 | fixed |
| new (2) | Potplayer | 1.6.62949 | mp4 | reported |
| new (5) | Splayer | 3.7 | mp4 | reported |
| new (2) | MS Word | 2007,10,16 | rtf | reviewing |
| new (1) | WPS Word | 10.1.0.5803 | doc | reported |
| new (2) | OpenOffice | 4.1.2 | doc | reviewing |
| new (1) | IrfanView | 4.41 | m3u | fixed |

# Resultados

Teste de falsos negativos

- Testado em 8 vulnerabilidades conhecidas
- Encontradas 6 vulnerabilidades
- Outras 2 passíveis de validação

# Resultados

| ID | Application | version | input |
|---|---|---|---|
| CVE-2010-1932 | Xnview | 1.97.4 | mbm |
| CVE-2011-5233 | irfanview | 4.30 | tif |
| OSVDB-83812 | ZipItFast | 3.0 pro | zip |
| CVE-2014-1761 | Microsoft Word | 2010 | rtf |
| EDB-ID-39353 | VLC | 2.2.1 | mp4 |
| EDB-ID-17363 | 1ClickUnzip | 3.0.0 | zip |
| CVE-2010-2553 | MediaPlayer | 9.00.00.4503 | avi |
| CVE-2015-0327 | Adobe Flash | 13sa | swf |

# Towards Efficient Heap Overflow Discovery

Cristiano Matsui[1], Kallil Miguel[1], Rodrigo Anater[1]

Universidade Tecnológica Federal do Paraná

[1]Departamento Acadêmico de Informática