Testing for Buffer Overflows by Length-Driven Symbolic Execution

Zhang Qirun,
Department of Computer Science and Engineering,
The Chinese University of Hong Kong

November 13, 2009



- 1 Introduction
 - Buffer Overflow
 - Symbolic Execution
- 2 Related work
 - Static Analysis
 - Dynamic Analysis
- 3 Our Approach
 - Motivation
 - Improvement
 - Key Idea
- 4 Q & A



- 1 Introduction
 - Buffer Overflow
 - Symbolic Execution
- - Static Analysis
 - Dynamic Analysis
- - Motivation
 - Improvement
 - Key Idea



Outline

The Essentials

■ The Definition



The Essentials

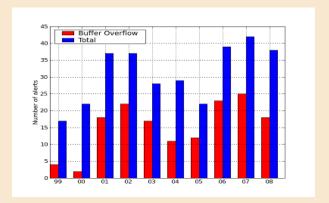
- The Definition
- Problem with C/C++

The Essentials

- The Definition
- Problem with C/C++
- US-CERT (United States Computer Emergency Readiness Team)

Why So Important?

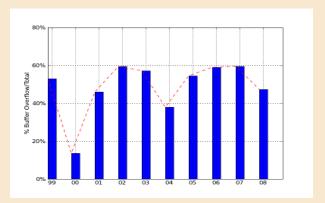
■ Number of Reported CERT Security Advisories



Why So Important?

Introduction

■ Precentage of reported Buffer Overflow



Outline

Two Kinds of Buffer Overflow

Stack Overflow



Two Kinds of Buffer Overflow

- Stack Overflow
- Heap Corruption

The power of Buffer Overflow

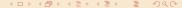
An example...



Outline

Fighting with Buffer Overflow

Choice of programming language



Fighting with Buffer Overflow

Introduction

- Choice of programming language
- Use of safe libraries

Fighting with Buffer Overflow

Introduction

- Choice of programming language
- Use of safe libraries
- Buffer overflow protection(Dynamic and Static) example...

Fighting with Buffer Overflow

- Choice of programming language
- Use of safe libraries
- Buffer overflow protection(Dynamic and Static) example...
- Pointer protection

Outline

Fighting with Buffer Overflow

- Choice of programming language
- Use of safe libraries
- Buffer overflow protection(Dynamic and Static) example...
- Pointer protection
- Address space layout randomization



Outline

What We want to do...

■ We focus on...



Symbolic Execution

Outline

What's Symbolic Execution

■ What's Symbolic Execution



What's Symbolic Execution

- What's Symbolic Execution
- Advantages

Symbolic Execution

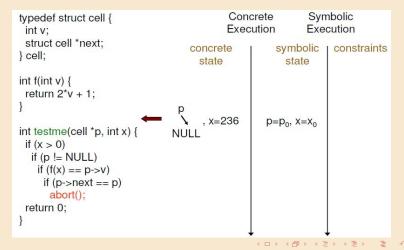
Outline

What's Symbolic Execution

- What's Symbolic Execution
- Advantages
- Disadvantages

```
typedef struct cell {
 int v:
 struct cell *next;
} cell:
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
 if (x > 0)
  if (p != NULL)
    if (f(x) == p \rightarrow v)
      if (p->next == p)
       abort();
 return 0:
```

- · Random Test Driver:
 - random memory graph reachable from p
 - random value for x
- Probability of reaching abort() is extremely low



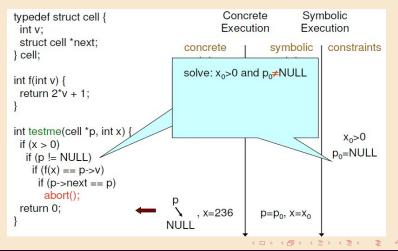
```
typedef struct cell {
                                               Concrete
                                                                 Symbolic
                                               Execution
                                                                 Execution
 int v:
 struct cell *next;
                                      concrete
                                                          symbolic
                                                                       constraints
} cell;
                                         state
                                                            state
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
 if (x > 0)
                                         x = 236
                                                        p=p_0, x=x_0
  if (p != NULL)
                                  NULL
    if (f(x) == p -> v)
     if (p->next == p)
      abort();
 return 0;
```

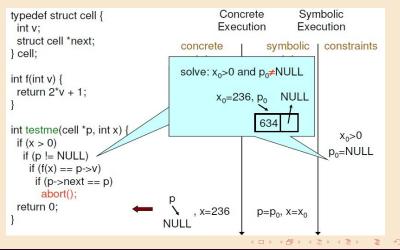
Symbolic Execution

Outline

```
Concrete
                                                                  Symbolic
typedef struct cell {
                                               Execution
                                                                  Execution
 int v:
 struct cell *next;
                                       concrete
                                                          symbolic
                                                                       constraints
} cell:
                                         state
                                                             state
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
                                                                           x_0 > 0
 if (x > 0)
  if (p != NULL)
                                          x = 236
                                                        p=p_0, x=x_0
    if (f(x) == p \rightarrow v)
                                   NULL
     if (p->next == p)
      abort();
 return 0;
                                                              圖▶ 《意》《意》
```

```
Concrete
typedef struct cell {
                                                                    Symbolic
                                                Execution
                                                                   Execution
 int v;
 struct cell *next;
                                                            symbolic
                                                                         constraints
                                        concrete
} cell:
                                          state
                                                              state
int f(int v) {
 return 2*v + 1:
int testme(cell *p, int x) {
                                                                             x_0 > 0
 if (x > 0)
                                                                         !(p_0!=NULL)
  if (p != NULL)
    if (f(x) == p \rightarrow v)
     if (p->next == p)
       abort();
 return 0;
                                                          p=p_0, x=x_0
```





```
Concrete
                                                                    Symbolic
typedef struct cell {
                                                 Execution
                                                                    Execution
 int v:
 struct cell *next;
                                                                          constraints
                                        concrete
                                                            symbolic
} cell:
                                          state
                                                               state
int f(int v) {
 return 2*v + 1;
                                         NULL
                                                         p=p_0, x=x_0,
                                             . x=236
                                                          p->v=v_0
int testme(cell *p, int x) {
                                    634
                                                         p->next=n<sub>0</sub>
 if (x > 0)
  if (p != NULL)
    if (f(x) == p \rightarrow v)
     if (p->next == p)
       abort();
 return 0;
                                                                母 ▶ 《 ≧ ▶ 《 ≧ ▶
```

```
Concrete
                                                                    Symbolic
typedef struct cell {
                                                 Execution
                                                                    Execution
 int v:
 struct cell *next:
                                        concrete
                                                             symbolic
                                                                          constraints
} cell:
                                           state
                                                               state
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
                                          NULL
                                                          p=p_0, x=x_0,
                                                                              x_0 > 0
 if (x > 0)
                                             , x=236
                                                           p->v=v_0
  if (p != NULL)
                                     634
                                                          p->next=n<sub>o</sub>
    if (f(x) == p \rightarrow v)
     if (p->next == p)
       abort();
 return 0;
```

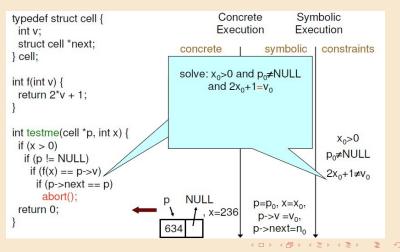
Introduction

Introduction

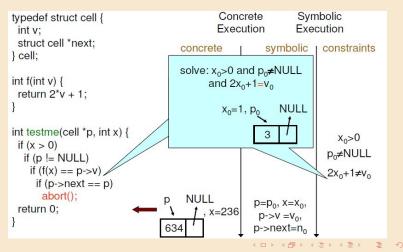
```
Concrete
                                                                      Symbolic
typedef struct cell {
                                                  Execution
                                                                     Execution
 int v;
 struct cell *next;
                                         concrete
                                                              symbolic
                                                                            constraints
} cell:
                                           state
                                                                state
int f(int v) {
 return 2*v + 1:
int testme(cell *p, int x) {
                                                                               x_0 > 0
 if (x > 0)
                                          NULL
                                                                             p<sub>0</sub>≠NULL
                                                           p=p_0, X=X_0,
  if (p != NULL)
                                                x=236
                                                            p->v=v_0
    if (f(x) == p \rightarrow v)
                                     634
                                                           p->next=n<sub>o</sub>
     if (p->next == p)
       abort():
 return 0;
```

```
Concrete
                                                                     Symbolic
typedef struct cell {
                                                 Execution
                                                                    Execution
 int v:
 struct cell *next;
                                         concrete
                                                             symbolic
                                                                           constraints
} cell:
                                           state
                                                               state
int f(int v) {
 return 2*v + 1:
int testme(cell *p, int x) {
                                                                              x_0 > 0
 if (x > 0)
                                                                            p<sub>0</sub>≠NULL
  if (p != NULL)
                                          NULL
                                                          p=p_0, x=x_0,
                                                                            2x_0+1\neq v_0
    if (f(x) == p \rightarrow v)
                                            1, x=236
                                                           p->V=V_0
     if (p->next == p)
                                     634
                                                          p->next=n<sub>0</sub>
       abort():
 return 0;
                                                                 ◆圖→ ◆量→ ◆量→
```

```
Concrete
                                                                     Symbolic
typedef struct cell {
                                                  Execution
                                                                     Execution
 int v:
 struct cell *next:
                                         concrete
                                                             symbolic
                                                                           constraints
} cell;
                                                                state
                                           state
int f(int v) {
 return 2*v + 1:
int testme(cell *p, int x) {
                                                                               x_0 > 0
 if (x > 0)
  if (p != NULL)
    if (f(x) == p \rightarrow v)
                                                                             2x_0+1\neq v_0
     if (p->next == p)
       abort():
                                          NULL
                                                           p=p_0, x=x_0,
 return 0;
                                            t , x=236
                                                            p->V=V_0
                                                           p->next=n<sub>0</sub>
```



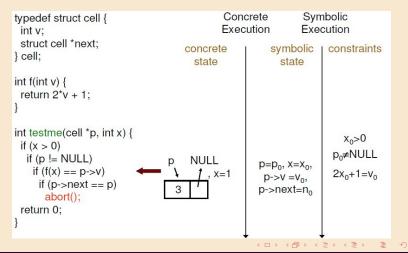
Symbolic Execution



```
Concrete
                                                                   Symbolic
typedef struct cell {
                                                Execution
                                                                  Execution
 int v;
 struct cell *next;
                                       concrete
                                                           symbolic
                                                                         constraints
} cell:
                                          state
                                                              state
int f(int v) {
 return 2*v + 1:
                                        NULL
                                                        p=p_0, x=x_0,
                                          f, x=1
                                                         p->V=V_0
int testme(cell *p, int x) {
                                     3
                                                        p->next=n<sub>0</sub>
 if (x > 0)
  if (p != NULL)
    if (f(x) == p \rightarrow v)
     if (p->next == p)
      abort();
 return 0:
                                                                圖▶ 《意》《意》
```

```
Concrete
                                                                    Symbolic
typedef struct cell {
                                                 Execution
                                                                    Execution
 int v:
 struct cell *next:
                                        concrete
                                                             symbolic
                                                                          constraints
} cell;
                                                               state
                                          state
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
                                          NULL
                                                                              x_0 > 0
                                                          p=p_0, X=X_0,
 if (x > 0)
                                             x=1
                                                           p->v=v_0
   if (p != NULL)
                                                          p->next=n<sub>o</sub>
    if (f(x) == p \rightarrow v)
     if (p->next == p)
       abort();
 return 0;
                                                         < □ > → □ > → □ > → □ > ·
```

```
Concrete
                                                                    Symbolic
typedef struct cell {
                                                 Execution
                                                                    Execution
 int v:
 struct cell *next;
                                        concrete
                                                            symbolic
                                                                          constraints
} cell;
                                          state
                                                               state
int f(int v) {
 return 2*v + 1:
int testme(cell *p, int x) {
                                                                              x_0 > 0
 if (x > 0)
                                         NULL
                                                                           p<sub>o</sub>≠NULL
                                                          p=p_0, x=x_0,
   if (p != NULL)
                                             . x=1
                                                           p->V=V_0
    if (f(x) == p \rightarrow v)
                                      3
                                                         p->next=no
     if (p->next == p)
       abort():
 return 0;
                                                               ◆圖 ▶ ◆ 圖 ▶ ◆ 圖 ▶
```

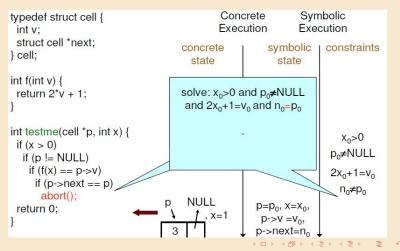


An example

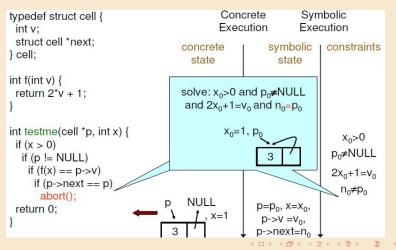
```
Concrete
                                                                      Symbolic
typedef struct cell {
                                                  Execution
                                                                      Execution
 int v;
 struct cell *next;
                                         concrete
                                                              symbolic
                                                                            constraints
} cell;
                                            state
                                                                state
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
                                                                                X_0 > 0
 if (x > 0)
                                                                             p<sub>0</sub>≠NULL
  if (p != NULL)
    if (f(x) == p \rightarrow v)
                                           NULL
                                                                             2x_0 + 1 = v_0
                                                           p=p_0, x=x_0,
     if (p->next == p)
                                            f , x=1
                                                            p->v=v_0
                                                                               n_0 \neq p_0
       abort():
                                                           p->next=n<sub>o</sub>
 return 0:
                                                                  ∢圖≯ ∢屋≯ ∢屋≯
```

Introduction

```
Concrete
                                                                        Symbolic
typedef struct cell {
                                                    Execution
                                                                        Execution
 int v:
 struct cell *next;
                                           concrete
                                                                symbolic
                                                                              constraints
} cell:
                                             state
                                                                   state
int f(int v) {
 return 2*v + 1;
int testme(cell *p, int x) {
                                                                                  x_0 > 0
 if (x > 0)
                                                                                p<sub>0</sub>≠NULL
   if (p != NULL)
    if (f(x) == p \rightarrow v)
                                                                                2x_0 + 1 = v_0
      if (p->next == p)
                                                                                  n_0 \neq p_0
       abort():
                                            NULL
                                                             p=p_0, x=x_0,
 return 0:
                                                . x=1
                                                               p->V=V_0
                                                             p->next=n<sub>0</sub>
```

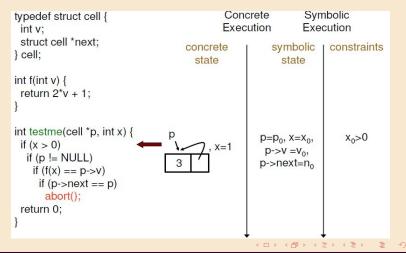


Introduction



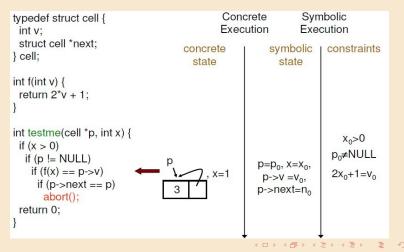
```
Concrete
                                                                    Symbolic
typedef struct cell {
                                                Execution
                                                                   Execution
 int v;
 struct cell *next:
                                                            symbolic
                                                                          constraints
                                        concrete
} cell:
                                          state
                                                              state
int f(int v) {
 return 2*v + 1:
                                                         p=p_0, x=x_0,
                                              x=1
                                                          p->v=v_0
int testme(cell *p, int x) {
                                                         p->next=n<sub>0</sub>
 if (x > 0)
  if (p != NULL)
    if (f(x) == p \rightarrow v)
     if (p->next == p)
       abort();
 return 0;
                                                                  → 《唐》《唐》
```

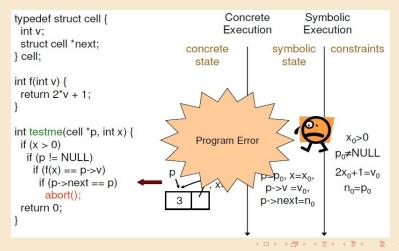
Symbolic Execution



```
Concrete
                                                                    Symbolic
typedef struct cell {
                                                 Execution
                                                                    Execution
 int v:
 struct cell *next:
                                        concrete
                                                             symbolic
                                                                          constraints
} cell:
                                          state
                                                               state
int f(int v) {
 return 2*v + 1:
int testme(cell *p, int x) {
                                                                              x_0 > 0
 if (x > 0)
                                                                           p<sub>o</sub>≠NULL
                                                          p=p_0, x=x_0,
  if (p != NULL)
                                              x = 1
                                                           p->V=V_0
    if (f(x) == p \rightarrow v)
                                                         p->next=n<sub>0</sub>
     if (p->next == p)
       abort():
 return 0;
                                                        《中》《圖》《意》《意》
```

Symbolic Execution





- - Buffer Overflow
 - Symbolic Execution
- Related work
 - Static Analysis
 - Dynamic Analysis
- - Motivation
 - Improvement
 - Key Idea



Outline

Static Analysis

Previous Work

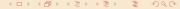


- Previous Work
 - LCLint (USENIX Security 01)



- Previous Work
 - LCLint (USENIX Security 01)
 - CSSV (PLDI 03)

- Previous Work
 - LCLint (USENIX Security 01)
 - CSSV (PLDI 03)
 - ESC/Java (PLDI 02)



- Previous Work
 - LCLint (USENIX Security 01)
 - CSSV (PLDI 03)
 - ESC/Java (PLDI 02)
 - ARCHER (FSE 03)

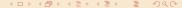
- Previous Work
 - LCLint (USENIX Security 01)
 - CSSV (PLDI 03)
 - ESC/Java (PLDI 02)
 - ARCHER (FSE 03)
- Drawbacks

Dynamic Analysis

Outline

Runtime Check

Previous Work



- Previous Work
 - Purify (Winter USENIX Conf 1992)

- Previous Work
 - Purify (Winter USENIX Conf 1992)
 - CCured (POPL 02)

Dynamic Analysis

- Previous Work
 - Purify (Winter USENIX Conf 1992)
 - CCured (POPL 02)
 - CERD (NDSS 04)

- Previous Work
 - Purify (Winter USENIX Conf 1992)
 - CCured (POPL 02)
 - CERD (NDSS 04)
- Drawbacks

Dynamic Analysis

Outline

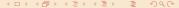
Symbolic Execution

Previous Work



Dynamic Analysis

- Previous Work
 - DART(Random Testing) (PLDI 05)



- Previous Work
 - DART(Random Testing) (PLDI 05)
 - CUTE (FSE 05)

- Previous Work
 - DART(Random Testing) (PLDI 05)
 - CUTE (FSE 05)
 - KLEE (OSDI 08)

- Previous Work
 - DART(Random Testing) (PLDI 05)
 - CUTE (FSE 05)
 - KLEE (OSDI 08)
 - Length Abstraction (ISSTA 08)

Dynamic Analysis

Outline

- Previous Work
 - DART(Random Testing) (PLDI 05)
 - CUTE (FSE 05)
 - KLEE (OSDI 08)
 - Length Abstraction (ISSTA 08)
- Drawbacks

Our Approach

Outline

- 1 Introduction
 - Buffer Overflow
 - Symbolic Execution
- 2 Related work
 - Static Analysis
 - Dynamic Analysis
- 3 Our Approach
 - Motivation
 - Improvement
 - Key Idea
- 4 Q & A

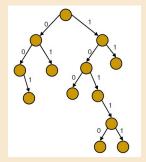


• 0

Motivation

Motivation

Motivation

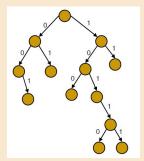


0

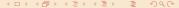
Outline

Motivation

Motivation



Three forms of Buffer overflow



Outline

Improvement

■ Compared with dynamic analysis

Improvement

- Compared with dynamic analysis
 - symbolic execution

Improvement

- Compared with dynamic analysis
 - symbolic execution
 - runtime check

Outline

Improvement

- Compared with dynamic analysis
 - symbolic execution
 - runtime check
- Compared with static analysis

- Compared with dynamic analysis
 - symbolic execution
 - runtime check
- Compared with static analysis
- Compared with Length Abstraction (Unsure -,-)

Our Approach

Key Idea (by example)

■ Sendmail 8.12.7 (daemon.c)

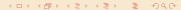
```
char buf[50];
          char *p=\&buf[0];
          char *s;
          char ibuf[50];
          int i, nleft; */
    nleft=sizeof(ibuf)-1;
    while ((i=read(s,p,nleft))>0)
        p+=i:
        nleft -= i:
        *p=' \ 0':
        if(strchr(ibuf, '\n') != NULL || nleft <= 0)
            break;
    close(s);
    if(i < 0||p = &ibuf[0])
        goto noident:
    if(*--p=='\n' && *--p == '\r')
    *++p = ' \ n':
noident:
    return 0;
```

- - Buffer Overflow
 - Symbolic Execution
- - Static Analysis
 - Dynamic Analysis
- - Motivation
 - Improvement
 - Key Idea
- 4 Q & A



Q & A

Q & A



- Q & A
- Thanks