Effective Program Debloating via Reinforcement Learning

Kihong Heo\textsuperscript{1} 
Pardis Pashakhanloo\textsuperscript{1} 

Woosuk Lee\textsuperscript{1,2} 
Mayur Naik\textsuperscript{1} 

University of Pennsylvania\textsuperscript{1} 
Hanyang University\textsuperscript{2} 

CCS 2018
Growth of SW Complexity

Linux Kernel

Size (LOC)

Version

1.0 2.0 2.6.16 (1st LTS) 2.6.34 (5th LTS) 3.10 (10th LTS) 4.1 (15th LTS) 4.19 (19th LTS)

0M 5M 10M 15M 20M 25M 30M

*https://www.linuxcounter.net
Consequences of SW Bloat

Performance  Maintainability  Security
Consequences of SW Bloat

- Example: security vulnerability in GNU tar
Consequences of SW Bloat

- Example: security vulnerability in GNU `tar`
Consequences of SW Bloat

- Example: security vulnerability in GNU tar

How can we reverse this trend?
State-of-the-Practice

General-purpose tar
- Out-of-the-box Linux

Customized tar
- BusyBox Utility Package*

*https://busybox.net
# State-of-the-Practice

<table>
<thead>
<tr>
<th>General-purpose tar</th>
<th>Customized tar</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Out-of-the-box Linux</td>
<td>- BusyBox Utility Package*</td>
</tr>
<tr>
<td>- 97 cmd line options</td>
<td>- 8 cmd line options</td>
</tr>
</tbody>
</table>

*https://busybox.net
State-of-the-Practice

General-purpose tar
- Out-of-the-box Linux
- 97 cmd line options
- 45,778 LOC
- 13,227 statements

Customized tar
- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements

*https://busybox.net
State-of-the-Practice

General-purpose tar
- Out-of-the-box Linux
- 97 cmd line options
- 45,778 LOC
- 13,227 statements
- CVE-2016-6321

Customized tar
- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements
- No known CVEs

*https://busybox.net
State-of-the-Practice

**General-purpose tar**
- Out-of-the-box Linux
- 97 cmd line options
- 45,778 LOC
- 13,227 statements
- CVE-2016-6321

**Customized tar**
- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements
- No known CVEs

*https://busybox.net
Our Goal

General-purpose tar
- Out-of-the-box Linux
- 97 cmd line options
- 45,778 LOC
- 13,227 statements
- CVE-2016-6321

Customized tar
- BusyBox Utility Package*
- 8 cmd line options
- 1,646
- 3,287 LOC
- 518
- 403 statements
- No known CVEs

*https://busybox.net
Our Contribution

Chisel: an automated program debloating system
Our Contribution

Chisel: an automated program debloating system

• **minimality**: trim code as aggressively as possible
Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible
- **efficiency**: scale to large programs
Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible
- **efficiency**: scale to large programs
- **robustness**: avoid introducing new vulnerabilities
Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible
- **efficiency**: scale to large programs
- **robustness**: avoid introducing new vulnerabilities
- **naturalness**: produce maintainable code
Our Contribution

Chisel: an automated program debloating system

• **minimality**: trim code as aggressively as possible
• **efficiency**: scale to large programs
• **robustness**: avoid introducing new vulnerabilities
• **naturalness**: produce maintainable code
• **generality**: handle a variety of programs and specs
Example: tar-1.14

```
int absolute_names;
int ignore_zeros_option;
struct tar_stat_info stat_info;

char *safer_name_suffix (char *file_name, int link_target) {
    int prefix_len;
    char *p;
    if (absolute_names) {
        p = file_name;
    } else {
        /* CVE-2016-6321 */
        /* Incorrect sanitization if “file_name” contains ".." */
    }
    ...
    return p;
}

void extract_archive() {
    char *file_name = safer_name_suffix(stat_info.file_name, 0);
    /* Overwrite “file_name” if exists */
    ...
}

void list_archive() { ... }

void read_and(void *(do_something)(void)) {
    enum read_header status;
    while (...) {
        status = read_header();
        switch (status) {
            case HEADER_SUCCESS: (*do_something)(); continue;
            case HEADER_ZERO_BLOCK:
                if (ignore_zeros_option) continue;
                else break;
        ...
            default: break;
        }
    ...
    }

    /* Supports all options: -x, -t, -P, -i, ... */
    int main(int argc, char **argv) {
        int optchar;
        while (optchar = getopt_long(argc, argv) != -1) {
            switch(optchar) {
                case 'x': read_and(&extract_archive); break;
                case 't': read_and(&list_archive); break;
                case 'P': absolute_names = 1; break;
                case 'i': ignore_zeros_option = 1; break;
            ...
        }
    }
```
Example: tar-1.14

int absolute_names;
int ignore_zeros_option;
struct tar_stat_info stat_info;

char *safer_name_suffix (char *file_name, int link_target) {
  int prefix_len;
  char *p;

  if (absolute_names) {
    p = file_name;
  } else {
    /* CVE-2016-6321 */
    /* Incorrect sanitization if “file_name” contains “..” */
    ...
  }
  ...
  return p;
}

void extract_archive() {
  char *file_name = safer_name_suffix(stat_info.file_name, 0);
  /* Overwrite “file_name” if exists */
  ...
}

void list_archive() { ...

void read_and(void *(do_something)(void)) {
  enum read_header status;
  while (...) {
    status = read_header();
    switch (status) {
      case HEADER_SUCCESS: (*do_something)(); continue;
      case HEADER_ZERO_BLOCK:
        if (ignore_zeros_option) continue;
        else break;
        ...
      default: break;
    }
    ...
  }
  ...
}

/* Supports all options: -x, -t, -P, -i, ... */
int main(int argc, char **argv) {
  int optchar;
  while (optchar = getopt_long(argc, argv) != -1) {
    switch(optchar) {
      case 'x': read_and(&extract_archive); break;
      case 't': read_and(&list_archive); break;
      case 'P': absolute_names = 1; break;
      case 'i': ignore_zeros_option = 1; break;
      ...
    }
  }
  ...
}
Talk Outline

- Motivation
- System Architecture
- Evaluation
- Conclusion
System Architecture
System Architecture

Program Debloating

Program Spec

Checker w.r.t. Spec

Trimmer

Learner
System Architecture

Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Reduced Program

Validation

Static

Dynamic
System Architecture

Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Reduced Program

Validation

Static

Dynamic

Success

Correct Reduced Program

Augmentation

Spec

Program
Key Questions

1. How to provide high-level specification?
Key Questions

2. How to effectively reduce programs?

1. How to provide high-level specification?

Program Debloating

- Checker w.r.t. Spec
- Trimmer
- Learner

Validation

- Static
- Dynamic

Augmentation

- Success
- Correct Reduced Program
- Failure

Program

Spec
Key Questions

2. How to effectively reduce programs?

3. How to validate robustness?

1. How to provide high-level specification?

Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Validation

Static

Reduced Program

Success

Correct Reduced Program

Dynamic

Failure

Augmentation
#!/bin/bash

function compile {
    clang -o tar.debloat tar-1.14.c
    return $?
}

# tests for the desired functionalities
function desired {
    # 1. archiving multiple files
    touch foo bar
    ./tar.debloat cf foo.tar foo bar
    rm foo bar
    ./tar.debloat xf foo.tar
    test -f foo -a -f bar || exit 1

    # 2. extracting from stdin
    touch foo
    ./tar.debloat cf foo.tar foo
    rm foo
    cat foo.tar | ./tar.debloat x
    test -f foo || exit 1

    # other tests
    ...
    return 0
}

# tests for the undesired functionalities
function undesired {
    # 1. archiving multiple files
    for test_script in `ls other_tests/*.sh`
    do
        { sh -x -e $test_script; } >& log
        grep 'Segmentation fault' log && exit 1
    done
    return 0
}

compile || exit 1
core || exit 1
non_core || exit 1
#!/bin/bash

function compile {
    clang -o tar.debloat tar-1.14.c
    return $?
}

# tests for the desired functionalities
function desired {
    # 1. archiving multiple files
    touch foo bar
    ./tar.debloat cf foo.tar foo bar
    rm foo bar
    ./tar.debloat xf foo.tar
    test -f foo -a -f bar || exit 1

    # 2. extracting from stdin
    touch foo
    ./tar.debloat cf foo.tar foo
    rm foo
    cat foo.tar | ./tar.debloat x
    test -f foo || exit 1

    # other tests
    ...
    return 0
}

# tests for the undesired functionalities
function undesired {
    for test_script in `ls other_tests/*.sh`
        do
            { sh -x -e $test_script; } >& log
            grep 'Segmentation fault' log && exit 1
        done
    return 0
}

compile || exit 1
core || exit 1
non_core || exit 1

1. The program is compilable.
High-level Specification

```bash
#!/bin/bash

function compile {
    clang -o tar.debloat tar-1.14.c
    return $?
}

function undesired {
    for test_script in `ls other_tests/*.sh`
    do
        sh -x -e $test_script; } >& log
grep 'Segmentation fault' log &&
    exit 1
    done
    return 0
}

# tests for the desired functionalities
function desired {
    # 1. archiving multiple files
    touch foo bar
    ./tar.debloat cf foo.tar foo bar
    rm foo bar
    ./tar.debloat xf foo.tar
    test -f foo -a -f bar || exit 1

    # 2. extracting from stdin
    touch foo
    ./tar.debloat cf foo.tar foo
    rm foo
cat foo.tar | ./tar.debloat x
test -f foo || exit 1

    # other tests
    ...
    return 0
}

2. The program produces the same results with the desired functionalities. (e.g., using regression test suites)

# tests for the undesired functionalities
function undesired {
    for test_script in `ls other_tests/*.sh`
    do
        sh -x -e $test_script; } >& log
grep 'Segmentation fault' log && exit 1
    done
    return 0
}

compile || exit 1
core || exit 1
non_core || exit 1
```
3. The program does not crash with the undesired functionalities. (e.g., using Clang sanitizers)

```bash
#!/bin/bash

function undesired {
    # 1. archiving multiple files
    touch foo bar
    ./tar.debloat cf foo.tar foo bar
    rm foo bar
    ./tar.debloat xf foo.tar
    test -f foo -a -f bar || exit 1

    # 2. extracting from stdin
    touch foo
    ./tar.debloat cf foo.tar foo
    rm foo
    cat foo.tar | ./tar.debloat x
    test -f foo || exit 1

    # other tests
    ...
    return 0
}
```

# tests for the undesired functionalities

```bash
#!/bin/bash
function undesired {
    for test_script in `ls other_tests/*.sh`
    do
        { sh -x -e $test_script; } >& log
        grep 'Segmentation fault' log &&& exit 1
    done
    return 0
}
```

compile || exit 1
core || exit 1
non_core || exit 1
Key Questions

2. How to effectively reduce programs?

3. How to validate robustness?

Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Validation

Static

Dynamic

Success

Failure

Augmentation

1. How to provide high-level specification?
Delta Debugging (DD)

• Oracle $O$ takes a program and returns Pass or Fail
• Given a program $P$, find a 1-minimal $P^*$ such that $O(P^*) = \text{Pass}$
• 1-minimal: removing any element of $P^*$ does not pass

[Zeller and Hildebrandt, 2002]
Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]

- Oracle $O$ takes a program and returns Pass or Fail
Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]

- Oracle $O$ takes a program and returns Pass or Fail
- Given a program $P$, find a 1-minimal $P^*$ such that $O(P^*) = \text{Pass}$
- 1-minimal: removing any element of $P^*$ does not pass $O$

$P_i$ 

Candidate for $P_{i+1}$

Oracle (test script)
Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]

- Oracle $O$ takes a program and returns Pass or Fail
- Given a program $P$, find a 1-minimal $P^*$ such that $O(P^*) = \text{Pass}
- 1$-minimal: removing any element of $P^*$ does not pass $O
- Time complexity: $O(|P|^2)$
DD: Key Challenges

[Zeller and Hildebrandt, 2002]

- Oracle $O$ takes a program and returns Pass or Fail
- Given a program $P$, find a 1-minimal $P^*$ such that $O(P^*) = \text{Pass}$
- 1-minimal: removing any element of $P^*$ does not pass $O$
- Time complexity: $O(|P|^2)$
DD: Key Challenges

[Zeller and Hildebrandt, 2002]

- Oracle $O$ takes a program and returns Pass or Fail
- Given a program $P$, find a **1-minimal** $P^*$ such that $O(P^*) = \text{Pass}$
- **1-minimal**: removing any element of $P^*$ does not pass $O$
- Time complexity: $O(|P|^2)$
Our Solution: Learning-guided DD

- Learn a policy for DD using reinforcement learning (RL)
- Guide the search based on the prediction of the learned policy
- Still guarantee 1-minimality and $O(|P|^2)$ time complexity
- Discard nonsensical programs upfront (e.g., invalid syntax, no main, uninitialized variables, etc)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>&lt;0, 1, ..., 1&gt;</td>
</tr>
<tr>
<td>$P_1$</td>
<td>&lt;0, 0, ..., 1&gt;</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$P_{i-1}$</td>
<td>&lt;1, 1, ..., 1&gt;</td>
</tr>
</tbody>
</table>

Most Likely Candidate for $P_{i+1}$

Oracle (test script)
Our Solution: Learning-guided DD

- Learn a policy for DD using reinforcement learning (RL)
Our Solution: Learning-guided DD

- **Learn a policy** for DD using reinforcement learning (RL)
- **Guide the search** based on the prediction of the learned policy

---

**Table:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$&lt;0, 1, .., 1&gt;$ 👍</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$&lt;0, 0, .., 1&gt;$ 👎</td>
</tr>
<tr>
<td>$\cdots$</td>
<td></td>
</tr>
<tr>
<td>$P_{i-1}$</td>
<td>$&lt;1, 1, .., 1&gt;$ 👎</td>
</tr>
</tbody>
</table>

**Oracle (test script):**

**Most Likely Candidate for $P_{i+1}$**
Our Solution: Learning-guided DD

- **Learn a policy** for DD using reinforcement learning (RL)
- **Guide the search** based on the prediction of the learned policy
- Still guarantee **1-minimality** and **$O(|P|^2)$ time complexity**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$&lt;0, 1, .., 1&gt;$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$&lt;0, 0, .., 1&gt;$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$P_{i-1}$</td>
<td>$&lt;1, 1, .., 1&gt;$</td>
</tr>
</tbody>
</table>

Data

Most Likely Candidate for $P_{i+1}$

Oracle (test script)
Our Solution:
Learning-guided DD

- **Learn a policy** for DD using reinforcement learning (RL)
- **Guide the search** based on the prediction of the learned policy
- Still guarantee **1-minimality** and **$O(|P|^2)$ time complexity**
- Discard nonsensical programs upfront (e.g., invalid syntax, no main, uninitialized variables, etc)

---

<table>
<thead>
<tr>
<th>Feature</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$&lt;0, 1, .., 1&gt;$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$&lt;0, 0, .., 1&gt;$</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$P_{i-1}$</td>
<td>$&lt;1, 1, .., 1&gt;$</td>
</tr>
</tbody>
</table>

Data

$P_i$

Most Likely Candidate for $P_{i+1}$

Oracle (test script)
Our Solution: Learning-guided DD

![Graph showing the comparison between guided and unguided approaches. The y-axis represents Size (LOC) ranging from 0 to 6000, and the x-axis represents # Trials ranging from 0 to 5000. Two lines are depicted: Guided in green and Unguided in red. The Guided line starts higher and crosses the Unguided line around 500 trials, indicating a faster reduction in Size (LOC). The Unguided line remains stable until the Guided line intersects it, after which both lines converge. The code `mkdir-5.2.1` is placed on the right side of the graph.](image-url)
Example

```c
/* mkdir-5.2.1 */
int xstrtol(char *s, char **ptr, int strtol_base, strtol_t *val,
            char *valid_suffixes) {
    err = 0;
    assert(0 <= strtol_base && strtol_base <= 36);
    p = ptr ? ptr : &t_ptr;
    q = s;
    while (ISSPACE (*q)) ++q;
    if (*q == '-') return LONGINT_INVALID;
    errno = 0;
    tmp = strtol(s, p, strtol_base);
    if (*p == s) { ... }
    if (!valid_suffixes) { ... }
    if (**p != '\'0') { ... }
    *val = tmp;
    return err;
}
```
Example

```c
/* mkdir-5.2.1 */
int xstrtol(char *s, char **ptr, int strtol_base, strtol_t *val,
            char *valid_suffixes) {
    err = 0;
    assert(0 <= strtol_base && strtol_base <= 36);
    p = ptr ? ptr : &t_ptr;
    q = s;
    while (ISSPACE (*q)) ++q;
    if (*q == '-') return LONGINT_INVALID;
    errno = 0;
    tmp = strtol(s, p, strtol_base);
    if (*p == s) { ... }
    if (!valid_suffixes) { ... }
    if (**p != '\0') { ... }
    *val = tmp;
    return err;
}
```

**Minimal Desired Program:**

```c
1 2 3 4 5 6 7 8 9 10 11 12 13
```
Unguided Delta-Debugging  Guided Delta-Debugging
Unguided Delta-Debugging

Guided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13

included
Unguided Delta-Debugging

Guided Delta-Debugging
<table>
<thead>
<tr>
<th>Unguided Delta-Debugging</th>
<th>Guided Delta-Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 ✘</td>
</tr>
<tr>
<td>2</td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 ✘</td>
</tr>
<tr>
<td>3</td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 ✘</td>
</tr>
</tbody>
</table>

...
Unguided Delta-Debugging

Guided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

... ...

16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔
Unguided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

Guided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✘

3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✘

... ...

16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

... ...

65 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔
Unguided Delta-Debugging

Guided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13 ×

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ×

3 1 2 3 4 5 6 7 8 9 10 11 12 13 ×

...
Unguided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

... ...

16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

Guided Delta-Debugging

1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 ✗

Feature vector Label
Unguided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
...
16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✓
...
65 1 2 3 4 5 6 7 8 9 10 11 12 13 ✓

Guided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

Feature vector

Label

12

6

N

Y

✓

✗

P* should include 6 and 12
Unguided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
...
16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔
...
65 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

Guided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗

52
Unguided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
...
16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔
...
65 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

Guided Delta-Debugging

1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✗
3 ✔
12 N
12 Y

Unguided Delta-Debugging

Guided Delta-Debugging
Unguided Delta-Debugging

Guided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13 ×

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ×

3 1 2 3 4 5 6 7 8 9 10 11 12 13 ×

...  

16 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

...  

65 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

1 2 3 4 5 6 7 8 9 10 11 12 13 ×

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ×

3 1 2 3 4 5 6 7 8 9 10 11 12 13 ×

...  

7 1 2 3 4 5 6 7 8 9 10 11 12 13 ✔

...  

55
Unguided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

2

3

... 16

65

Guided Delta-Debugging

1 2 3 4 5 6 7 8 9 10 11 12 13

2

3

... 7

30

...
Unguided Delta-Debugging

Guided Delta-Debugging

5,169 trials (4,872 failures)

1,174 trials (901 failures)
Key Questions

2. How to effectively reduce programs?

3. How to validate robustness?

1. How to provide high-level specification?
Validation

- Check the **robustness** of the reduced program
  - preventing newly introduced security holes
- Sound static buffer overflow analyzer (Sparrow)
  - #alarms in tar: 1,290 → 19 (feasible for manual inspection)
- Random fuzzer (AFL)
  - no crashing input found in 3 days for tar
Augmentation

- Augment the test script with crashing inputs by AFL
- Typically converges in up to 3 iterations in practice
- But, may be incomplete

```c
/* grep-2.19 */
void add_tok (token t) {
    /* removed in the first trial and restored after augmentation */
    if (dfa->talloc == dfa->tindex) {
        dfa->tokens = (token *) realloc (/* large size */);
        *(dfa->tokens + (dfa->tindex++)) = t;
    }
}
```
Talk Outline

• Motivation

• System Architecture

• Evaluation

• Conclusion
Experimental Setup

• 10 widely used **UNIX utility programs** (13—90 KLOC)
  
  • each program has a **known CVE**
  
  • **remove unreachable code** by static analysis upfront

• Specification:

  • supporting **the same cmd line options** as BusyBox
  
  • with the **test suites** by the original developers
# Statement

## Size of Reduced Program

<table>
<thead>
<tr>
<th>Program</th>
<th>Original</th>
<th>Chisel</th>
<th>Hand-written</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip-1.05</td>
<td>6,316</td>
<td>1,575</td>
<td>2,342</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>3,422</td>
<td>186</td>
<td>141</td>
</tr>
<tr>
<td>date-8.21</td>
<td>4,100</td>
<td>913</td>
<td>107</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>10,816</td>
<td>1,071</td>
<td>355</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>4,069</td>
<td>1,042</td>
<td>1,058</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>1,746</td>
<td>142</td>
<td>94</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>3,470</td>
<td>73</td>
<td>89</td>
</tr>
<tr>
<td>sort-8.16</td>
<td>7,206</td>
<td>379</td>
<td>89</td>
</tr>
<tr>
<td>tar-1.14</td>
<td>12,780</td>
<td>538</td>
<td>403</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td>1,923</td>
<td>192</td>
<td>51</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>55,848</strong></td>
<td><strong>6,111</strong></td>
<td><strong>4,729</strong></td>
</tr>
</tbody>
</table>
## Size of Reduced Program

<table>
<thead>
<tr>
<th>Program</th>
<th>Original</th>
<th>Chisel</th>
<th>Hand-written</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip-1.05</td>
<td>6,316</td>
<td>1,575</td>
<td>2,342</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>3,422</td>
<td>186</td>
<td>141</td>
</tr>
<tr>
<td>date-8.21</td>
<td>4,100</td>
<td>913</td>
<td>107</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>10,816</td>
<td>1,071</td>
<td>355</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>4,069</td>
<td>1,042</td>
<td>1,058</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>1,746</td>
<td>142</td>
<td>94</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>3,470</td>
<td>73</td>
<td>89</td>
</tr>
<tr>
<td>sort-8.16</td>
<td></td>
<td>379</td>
<td>89</td>
</tr>
<tr>
<td>tar-1.14</td>
<td></td>
<td>538</td>
<td>403</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td></td>
<td>192</td>
<td>51</td>
</tr>
<tr>
<td>Total</td>
<td>55,848</td>
<td>6,111</td>
<td>4,729</td>
</tr>
</tbody>
</table>

Reachable code by static analysis.
# Size of Reduced Program

<table>
<thead>
<tr>
<th>Program</th>
<th>Original</th>
<th>Chisel</th>
<th>Hand-written</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip-1.05</td>
<td>6,316</td>
<td>1,575</td>
<td>2,342</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>3,422</td>
<td>186</td>
<td>141</td>
</tr>
<tr>
<td>date-8.21</td>
<td>4,100</td>
<td>913</td>
<td>107</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>10,816</td>
<td>1,071</td>
<td>355</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>4,069</td>
<td>1,042</td>
<td>1,058</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>1,746</td>
<td>142</td>
<td>94</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>3,470</td>
<td>73</td>
<td>89</td>
</tr>
<tr>
<td>sort-8.16</td>
<td></td>
<td></td>
<td>89</td>
</tr>
<tr>
<td>tar-1.14</td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td></td>
<td></td>
<td>51</td>
</tr>
<tr>
<td>Total</td>
<td>55,848</td>
<td>6,111</td>
<td>4,729</td>
</tr>
</tbody>
</table>

Reachable code by static analysis

Chisel reduced 89%
# Statement

<table>
<thead>
<tr>
<th>Program</th>
<th>Original</th>
<th>Chisel</th>
<th>Hand-written</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip-1.05</td>
<td>6,316</td>
<td>1,575</td>
<td>2,342</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>3,422</td>
<td>186</td>
<td>141</td>
</tr>
<tr>
<td>date-8.21</td>
<td>4,100</td>
<td>913</td>
<td>107</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>10,816</td>
<td>1,071</td>
<td>355</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>4,069</td>
<td>1,042</td>
<td>1,058</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>1,746</td>
<td>142</td>
<td>94</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>3,470</td>
<td>73</td>
<td>89</td>
</tr>
<tr>
<td>sort-8.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tar-1.14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uniq-8.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>55,848</td>
<td>6,111</td>
<td>4,729</td>
</tr>
</tbody>
</table>

Reachable code by static analysis

Chisel reduced 89%

Comparable to hand-written versions
# Security Hardening

<table>
<thead>
<tr>
<th>Program</th>
<th>CVE</th>
<th>Original</th>
<th>Reduced</th>
<th>Original</th>
<th>Reduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip-1.05</td>
<td>✗</td>
<td>662</td>
<td>298</td>
<td>1,991</td>
<td>33</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>✓</td>
<td>534</td>
<td>162</td>
<td>47</td>
<td>1</td>
</tr>
<tr>
<td>date-8.21</td>
<td>✓</td>
<td>479</td>
<td>233</td>
<td>201</td>
<td>23</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>✓</td>
<td>1,065</td>
<td>411</td>
<td>619</td>
<td>31</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>✓</td>
<td>456</td>
<td>340</td>
<td>326</td>
<td>128</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>✗</td>
<td>229</td>
<td>124</td>
<td>43</td>
<td>2</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>✗</td>
<td>565</td>
<td>95</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>sort-8.16</td>
<td>✓</td>
<td>885</td>
<td>210</td>
<td>673</td>
<td>5</td>
</tr>
<tr>
<td>tar-1.14</td>
<td>✓</td>
<td>1,528</td>
<td>303</td>
<td>1,290</td>
<td>19</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td>✗</td>
<td>349</td>
<td>109</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>6,752</td>
<td>2,285</td>
<td>5,298</td>
<td>243</td>
</tr>
</tbody>
</table>

# ROP Gadgets | #Alarms

- Original: 662 (55%), 1,991 (98%)
- Reduced: 298 (55%), 33 (98%)
- Original: 534 (70%), 47 (98%)
- Reduced: 162 (70%), 1 (98%)
- Original: 479 (51%), 201 (89%)
- Reduced: 233 (51%), 23 (89%)
- Original: 1,065 (61%), 619 (95%)
- Reduced: 411 (61%), 31 (95%)
- Original: 456 (25%), 326 (61%)
- Reduced: 340 (25%), 128 (61%)
- Original: 229 (46%), 43 (95%)
- Reduced: 124 (46%), 2 (95%)
- Original: 565 (83%), 48 (100%)
- Reduced: 95 (83%), 0 (100%)
- Original: 885 (76%), 673 (99%)
- Reduced: 210 (76%), 5 (99%)
- Original: 1,528 (80%), 1,290 (99%)
- Reduced: 303 (80%), 19 (99%)
- Original: 349 (69%), 60 (98%)
- Reduced: 109 (69%), 1 (98%)

Total: 6,752 (66%), 5,298 (95%)
## Security Hardening

Remove 4 and 2 CVEs in undesired and desired functionalities. 4 CVEs are not easily fixable by reduction (e.g., race condition).

<table>
<thead>
<tr>
<th>Program</th>
<th>CVE</th>
<th>#ROP Gadgets</th>
<th>#Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Original</td>
<td>Reduced</td>
</tr>
<tr>
<td>bzip-1.05</td>
<td>✘</td>
<td>662</td>
<td>298 (55%)</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>✔</td>
<td>534</td>
<td>162 (70%)</td>
</tr>
<tr>
<td>date-8.21</td>
<td>✔</td>
<td>479</td>
<td>233 (51%)</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>✔</td>
<td>1,065</td>
<td>411 (61%)</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>✔</td>
<td>456</td>
<td>340 (25%)</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>✘</td>
<td>229</td>
<td>124 (46%)</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>✘</td>
<td>565</td>
<td>95 (83%)</td>
</tr>
<tr>
<td>sort-8.16</td>
<td>✔</td>
<td>885</td>
<td>210 (76%)</td>
</tr>
<tr>
<td>tar-1.14</td>
<td>✔</td>
<td>1,528</td>
<td>303 (80%)</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td>✘</td>
<td>349</td>
<td>109 (69%)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>6,752</td>
<td>2,285 (66%)</td>
</tr>
</tbody>
</table>
## Security Hardening

Remove 4 and 2 CVEs in undesired and desired functionalities. 4 CVEs are not easily fixable by reduction (e.g., race condition).

<table>
<thead>
<tr>
<th>Program</th>
<th>CVE</th>
<th>#ROP Gadgets</th>
<th>#Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Original</td>
<td>Reduced</td>
</tr>
<tr>
<td>bzip-1.05</td>
<td>✗</td>
<td>662</td>
<td>298</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>✔</td>
<td>534</td>
<td>162</td>
</tr>
<tr>
<td>date-8.21</td>
<td>✔</td>
<td>479</td>
<td>233</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>✔</td>
<td>1,065</td>
<td>411</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>✔</td>
<td>456</td>
<td>340</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>✗</td>
<td>229</td>
<td>124</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>✗</td>
<td>565</td>
<td>95</td>
</tr>
<tr>
<td>sort-8.16</td>
<td>✔</td>
<td>673</td>
<td>5</td>
</tr>
<tr>
<td>tar-1.14</td>
<td>✔</td>
<td>1,290</td>
<td>19</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td>✗</td>
<td>349</td>
<td>149</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>6,752</td>
<td>2,285</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,298</td>
<td>243</td>
</tr>
</tbody>
</table>

### Reduced potential attack surface
Security Hardening

Remove 4 and 2 CVEs in undesired and desired functionalities. 4 CVEs are not easily fixable by reduction (e.g., race condition).

<table>
<thead>
<tr>
<th>Program</th>
<th>CVE</th>
<th>#ROP Gadgets</th>
<th></th>
<th>#Alarms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Original</td>
<td>Reduced</td>
<td>Original</td>
<td>Reduced</td>
</tr>
<tr>
<td>bzip-1.05</td>
<td>✘</td>
<td>662</td>
<td>298</td>
<td>(55%)</td>
<td>1,991</td>
</tr>
<tr>
<td>chown-8.2</td>
<td>✔</td>
<td>534</td>
<td>162</td>
<td>(70%)</td>
<td>47</td>
</tr>
<tr>
<td>date-8.21</td>
<td>✔</td>
<td>479</td>
<td>233</td>
<td>(51%)</td>
<td>201</td>
</tr>
<tr>
<td>grep-2.19</td>
<td>✔</td>
<td>1,065</td>
<td>411</td>
<td>(61%)</td>
<td>619</td>
</tr>
<tr>
<td>gzip-1.2.4</td>
<td>✔</td>
<td>456</td>
<td>340</td>
<td>(25%)</td>
<td>326</td>
</tr>
<tr>
<td>mkdir-5.2.1</td>
<td>✘</td>
<td>229</td>
<td>124</td>
<td>(46%)</td>
<td>43</td>
</tr>
<tr>
<td>rm-8.4</td>
<td>✘</td>
<td>565</td>
<td>95</td>
<td>(83%)</td>
<td>48</td>
</tr>
<tr>
<td>sort-8.16</td>
<td>✔</td>
<td>349</td>
<td>149</td>
<td>(69%)</td>
<td>60</td>
</tr>
<tr>
<td>tar-1.14</td>
<td>✔</td>
<td>64</td>
<td>345</td>
<td>(76%)</td>
<td>123</td>
</tr>
<tr>
<td>uniq-8.16</td>
<td>✘</td>
<td>603</td>
<td>349</td>
<td>(69%)</td>
<td>60</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>6,752</td>
<td>2,285</td>
<td>(66%)</td>
<td>5,298</td>
</tr>
</tbody>
</table>

Reduced potential attack surface
Make it feasible for manual alarm inspection
Reduction Time

- bzip-1.05
- chown-8.2
- date-8.21
- grep-2.19
- gzip-1.2.4
- mkdir-5.2.1
- rm-8.4
- sort-8.16
- tar-1.14
- uniq-8.16

Chisel
Perses [ICSE’18]
C-Reduce [PLDI’12]
Reduction Time

Line-based reducer ran out of time for 6 programs

Chisel  Perses [ICSE’18]  C-Reduce [PLDI’12]

Hours

bzip-1.05  chown-8.2  date-8.21  grep-2.19  gzip-1.2.4  mkdir-5.2.1  rm-8.4  sort-8.16  tar-1.14  uniq-8.16
Reduction Time

Grammar-based reducer ran out of time for 2 programs

Chisel
Perses [ICSE’18]
C-Reduce [PLDI’12]

bzip-1.05  chown-8.2  date-8.21  grep-2.19  gzip-1.2.4  mkdir-5.2.1  rm-8.4  sort-8.16  tar-1.14  uniq-8.16

Grammar-based reducer ran out of time for 2 programs
Reduction Time

7x and 4x faster than C-Reduce and Perses

Chisel

Perses [ICSE’18]

C-Reduce [PLDI’12]
Conclusion

• **Chisel**: automated software debloating system
  • **tractable search** via learning-guided delta debugging
  • **security hardening** by removing undesired features
  • **robustness** via static & dynamic analyses
  • https://github.com/aspire-project/chisel

• **In the paper,**
  • reduction algorithm details
  • learning a debloating policy
  • engineering issues and design choices

**Acknowledgment**: Total Platform Cyber Protection (TPCP)