DICOS : Discovering Insecure Code Snippets from Stack Overflow Posts by Leveraging User Discussions

Hyunji Hong,  Seunghoon Woo,  Heejo Lee
Korea University

Annual Computer Security Applications Conference (ACSAC 2021)
Goal

• Discovering insecure code snippets from Stack Overflow posts

Motivation

• Developers copy and paste code snippets from online Q&A fora
• Reusing code snippets without understanding the code implication
  • compromise the security of the software
  • propagation of insecure code snippets
Motivating example

Post #122721

How do I trim leading/trailing whitespace in a standard way?

Is there a clean, preferably standard method of trimming leading and trailing whitespace from a string in C? I'd roll my own, but I would think this is a common problem with an equally common solution.

If you can modify the string:

```c
char *trimwhitespace(char *str)
{
    char *end;
    // Yrle leading space
    while (isspace((unsigned char)*str)) str++;
    if (str == 0) // All spaces?
        return str;
    // Yrle trailing space
    end = str + strlen(str) - 1;
    while (end > str && !isspace((unsigned char)(*end))) end--;
    // Write new null terminator character
    *end++ = ' ';
    return str;
}
```

Question

Answer post uploaded

Answer post revised

Jun. 2018

Security issues revealed by comments

Sep. 2008

Question uploaded

History of post #122721
## Motivating example

*Security issues are revealed by user discussions*

### Code Snippets

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>char *trimwhitespace(char *str) {</td>
</tr>
<tr>
<td>2</td>
<td>char *end;</td>
</tr>
<tr>
<td>3</td>
<td>// Trim leading space</td>
</tr>
<tr>
<td>4</td>
<td>while(isspace(*str)) str++;</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>// Trim trailing space</td>
</tr>
<tr>
<td>7</td>
<td>end = str + strlen(str) - 1;</td>
</tr>
<tr>
<td>8</td>
<td>while(end &gt; str &amp;&amp; isspace(*end)) end--;</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>// Write new null terminator character</td>
</tr>
<tr>
<td>11</td>
<td>*(end[1]+1) = 0;</td>
</tr>
<tr>
<td>12</td>
<td>return str;</td>
</tr>
<tr>
<td>13</td>
<td>}</td>
</tr>
</tbody>
</table>

### Comments

- **Line 4**: `isspace()` may cause undefined behavior
- **Line 12**: "*str" may cause a null pointer dereference

---

Motivating example

`trimwhitespace` function is intended to trim leading and trailing whitespace from strings. However, the function could be prone to undefined behavior due to the use of `isspace()` in line 4. Additionally, dereferencing `*str` in line 12 could result in a null pointer dereference.
Motivating example

```c
char *trimwhitespace(char *str) {
    char *end;
    // Trim leading space
    while (isspace(*str)) str++;
    // Trim trailing space
    end = str + strlen(str) - 1;
    while (end > str && isspace(*end)) end--;
    // Write new null terminator character
    *(end[1]+1) = 0;
    return str;
}
```

Fix for Security Issue

```c
char *trimwhitespace(char *str) {
    char *end;
    // Trim leading space
    while (isspace(*str)) str++;
    // Trim trailing space
    end = str + strlen(str) - 1;
    while (end > str && isspace(*end)) end--;
    // Write new null terminator character
    *(end[1]+1) = 0;
    return str;
}
```
Discovering Insecure Code Snippets

- An approach for discovering insecure code snippets in Stack Overflow posts

- **Key ideas**
  - an accurate approach by examining the change history of Stack Overflow posts for discovering insecure code snippets
    - Phase 1: extracting the change history from the post
    - Phase 2: analyzing the diffs using selected three features
    - Phase 3: determining whether the post contains insecure code snippets
P1. Extracting the change history of a post

1) Collect Stack Overflow posts
2) Extract all the change histories of each post
3) Extract **Diffs** between the oldest and the latest revision of the post
P1. Extracting the change history of a post

Code snippet pairing problem

Simple diffing code snippets can extract the erroneous change history

* CS = Code Snippet
P1. Extracting the change history of a post

Code snippet pairing

Generating new code snippets by extracting functions

* CS = Code Snippet
P1. Extracting the change history of a post

Code snippet pairing

Pairing in order of highest score based on similarity score

* CS = Code Snippet
P2. Analyzing the extracted change history

Feature selection
Large-scale empirical study using CVE vulnerabilities

Initial feature selection (related approaches)
F1. Changes in security-sensitive APIs ✔
F2. Changes in security-related keywords ✔
F3. Changes in control flows ✔
F4. Changes in literals
F5. Changes in identifiers
F6. Changes in function calls (APIs)

1) Select F1

Security patch 17%
General patch 4%

4 times higher
P2. Analyzing the extracted change history

Feature selection
Large-scale empirical study using CVE vulnerabilities

Initial feature selection (related approaches)
- F1. Changes in security-sensitive APIs
- F2. Changes in security-related keywords
- F3. Changes in control flows
- F4. Changes in literals
- F5. Changes in identifiers
- F6. Changes in function calls (APIs)

2) Select F2, F3

- Security patch more than 60%
- General patch less than 26%
P2. Analyzing the extracted change history

Feature selection
Large-scale empirical study using CVE vulnerabilities

Initial feature selection (related approaches)
F1. Changes in security-sensitive APIs ✔
F2. Changes in security-related keywords ✔
F3. Changes in control flows ✔
F4. Changes in literals
F5. Changes in identifiers
F6. Changes in function calls (APIs)

We select three features
F1. Changes in security-sensitive APIs
F2. Changes in security-related keywords
F3. Changes in control flows
P2. Analyzing the extracted change history

Analyzing code snippets

- Dicos checks changes in **security-sensitive APIs** and **control flows**

![code snippet Diffs]

- Checks if deleted code lines contain **security-sensitive APIs**
- Checks whether the diffs contain a change in **control flows** or **conditional statements**
P2. Analyzing the extracted change history

Analyzing descriptions and comments

- Dicos checks whether `security-related keyword pair` is included in the diffs

[Example]

- Fix typos in comments and improve readability
- Fixed math to handle `negative` angles...

Check if (noun, verb) or (modifier, verb) in each sentence
P3. Determining insecure code snippets

if two or more features are detected → insecure post

{...} changes in security-related keyword pair

& changes in security-sensitive APIs

& changes in control flows
Evaluation

Dataset collection

- Google BigQuery, SOTorrent dataset (version. 2020-12-31)
  - We collected Stack Overflow posts tagged with C, C++, Android
  - We extracted a total of 1,958,283 Stack Overflow answer posts
  - 668,520 (34%) posts contain at least one change history
Evaluation

- Dicos discovered **12,458 insecure posts**

- **Accuracy measurement**

  G1. All posts with **three selected features**
  
  G2. Top 400 posts with **two selected features**
  
  G3. Randomly selected 200 posts with **two features**
  
  G4. Top 400 posts with only **one feature**
  
  G5. Top 200 posts **without features**

<table>
<thead>
<tr>
<th>ID</th>
<th>#Total Posts</th>
<th>#TP</th>
<th>#FP</th>
<th>#TN</th>
<th>#FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>788</td>
<td>757</td>
<td>31</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>G2</td>
<td>400</td>
<td>346</td>
<td>54</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>G3</td>
<td>200</td>
<td>162</td>
<td>38</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>G4</td>
<td>400</td>
<td>N/A</td>
<td>N/A</td>
<td>318</td>
<td>82</td>
</tr>
<tr>
<td>G5</td>
<td>200</td>
<td>N/A</td>
<td>N/A</td>
<td>185</td>
<td>15</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1,988</strong></td>
<td><strong>1,265</strong></td>
<td><strong>123</strong></td>
<td><strong>503</strong></td>
<td><strong>97</strong></td>
</tr>
</tbody>
</table>

- **Precision**: 0.91
- **Recall**: 0.93
- **Accuracy**: 0.89

Accuracy measurement results for C, C++ and Android posts
Findings

Q1. Are older posts more likely to provide insecure code snippets?

Q2. Are accepted answer posts more secure than non-accepted posts?

Q3. What types of insecure code snippets were discovered?
Findings

Q1. Are older posts more likely to provide insecure code snippets?

Year distributions of secure and insecure posts discovered by Dicos
Findings

Q1. Are older posts more likely to provide insecure code snippets?

About 2% of Insecure Posts are uploaded each year

Year distributions of secure and insecure posts discovered by Dicos
Findings

Q2. Are accepted answer posts more secure than non-accepted posts?

Ratio of insecure posts between accepted and non-accepted posts discovered by Dicos

Accepted answer post

Non-accepted answer posts
Findings

Q2. Are accepted answer posts more secure than non-accepted posts?

Ratio of insecure posts between accepted and non-accepted posts discovered by Dicos

Accepted (1.67%) ☑️
Non-accepted (1.99%)

almost same
Findings

Q3. What types of insecure code snippets were discovered?

Types of discovered insecure code snippets:

- Undefined behavior: 367
- Null-terminated string issue: 175
- Memory leak: 91
- Buffer overflow: 90
- Initialization issue: 49
- Infinite loop: 23
- Out-of-bounds error: 17
- Others: 68
Findings

Q3. What types of insecure code snippets were discovered?

Dicos covers various types of insecure code snippets

Types of discovered insecure code snippets

<table>
<thead>
<tr>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undefined behavior</td>
<td>367</td>
</tr>
<tr>
<td>Null-terminated string issue</td>
<td>175</td>
</tr>
<tr>
<td>Memory leak</td>
<td>91</td>
</tr>
<tr>
<td>Buffer overflow</td>
<td>90</td>
</tr>
<tr>
<td>Initialization issue</td>
<td>49</td>
</tr>
<tr>
<td>Infinite loop</td>
<td>23</td>
</tr>
<tr>
<td>Out-of-bounds error</td>
<td>17</td>
</tr>
<tr>
<td>Others</td>
<td>68</td>
</tr>
</tbody>
</table>
Conclusion

- We present Dicos, an accurate approach for discovering insecure code snippets in Stack Overflow posts by leveraging user discussions.

- Equipped with insecure code snippet discovery results from Dicos:
  - improve the credibility of Stack Overflow by addressing discovered insecure code snippets
  - create a safe code snippet reuse environment
Q&A

Thank you for your attention!
• Dicos repository (https://github.com/hyunji-Hong/Dicos-public)

CONTACT
• Hyunji Hong (hyunji_hong@korea.ac.kr)
• Computer & Communication Security Lab (https://ccs.korea.ac.kr)