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)

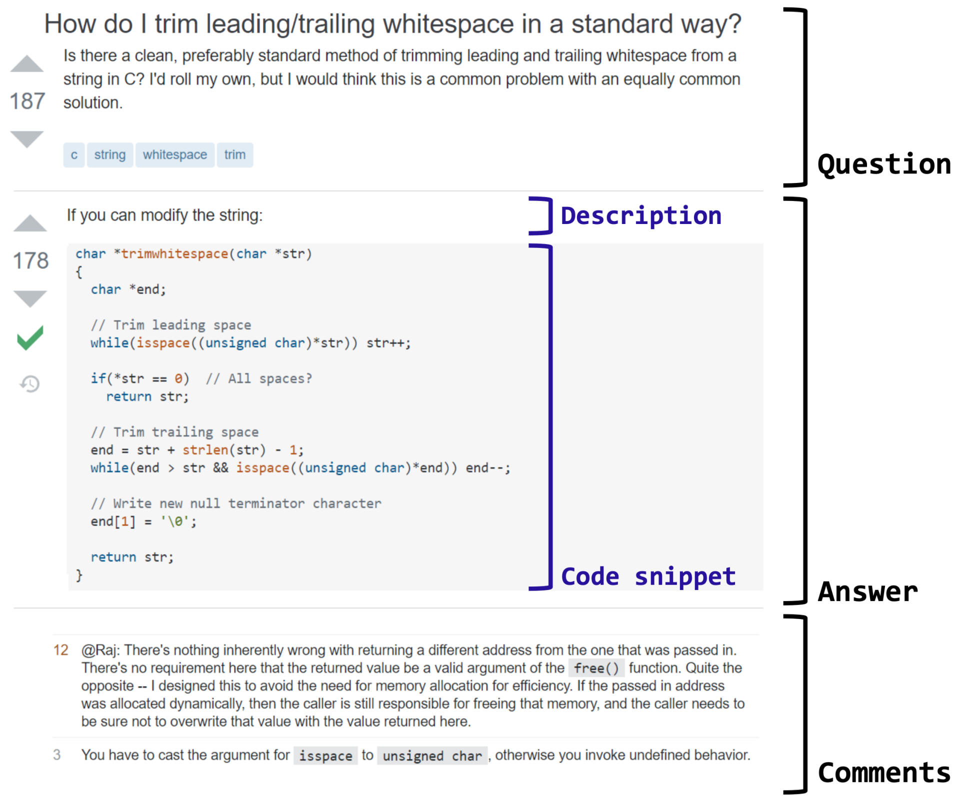
Goal

- **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?

187

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.

c string whitespace trim

178

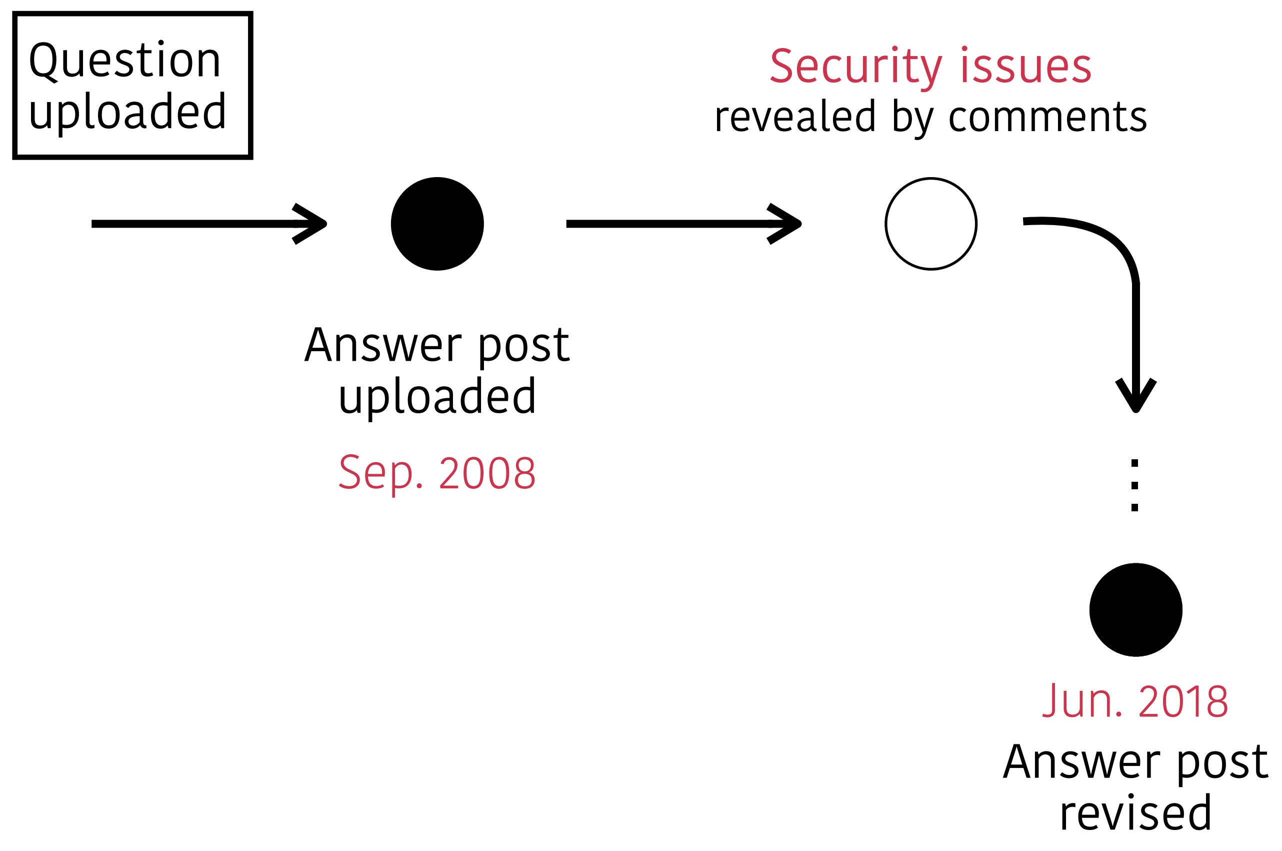
If you can modify the string:

```c
char *trimwhitespace(char *str)
{
  char *end;

  // Trim leading space
  while(isspace((unsigned char)*str)) str++;

  if(*str == 0)  // All spaces?
    return str;

  // Trim trailing space
  end = str + strlen(str) - 1;
  while(end > str && isspace((unsigned char)*end)) end--;

  // Write new null terminator character
  end[1] = '\0';

  return str;
}
```

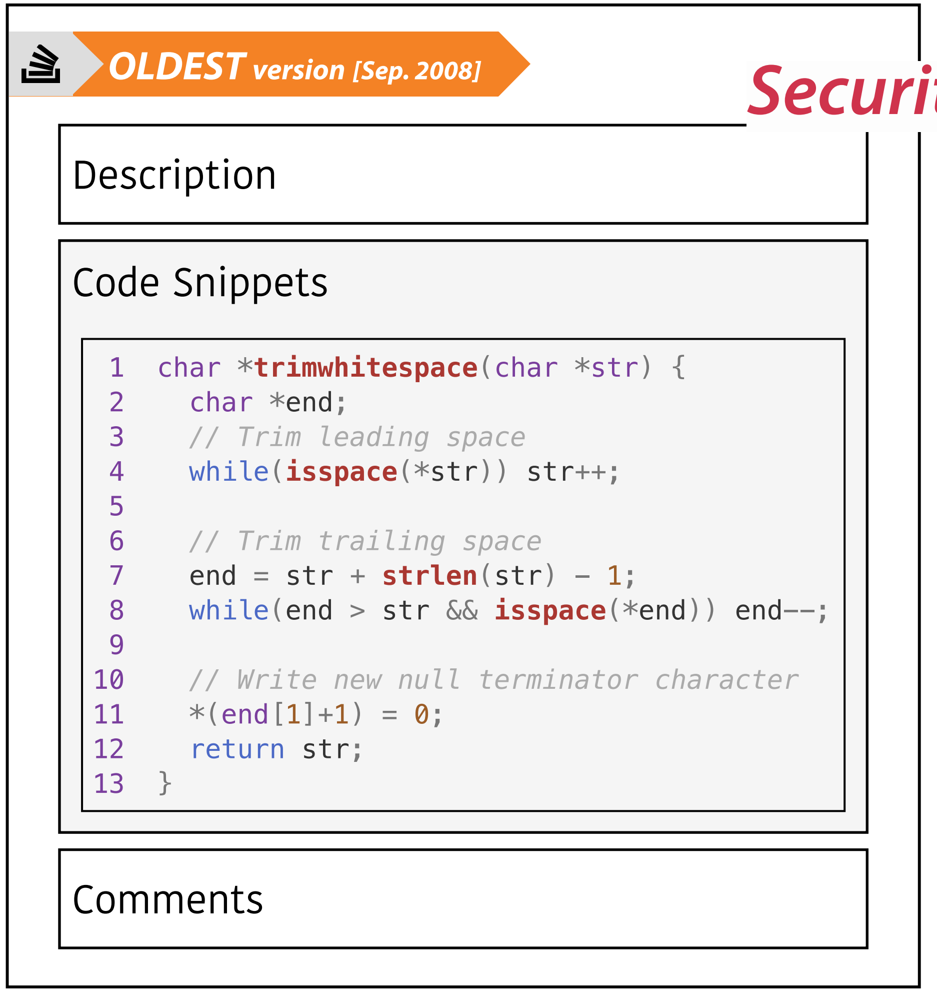
12 @Raj: There's nothing inherently wrong with returning a different address from the one that was passed in. There's no requirement here that the returned value be a valid argument of the `free()` function. Quite the opposite -- I designed this to avoid the need for memory allocation for efficiency. If the passed in address was allocated dynamically, then the caller is still responsible for freeing that memory, and the caller needs to be sure not to overwrite that value with the value returned here.

3 You have to cast the argument for `isspace` to `unsigned char`, otherwise you invoke undefined behavior.

History of post #122721



Motivating example



OLDEST version [Sep. 2008]

Description

Code Snippets

```
1  char *trimwhitespace(char *str) {
2    char *end;
3    // Trim leading space
4    while(isspace(*str)) str++;
5
6    // Trim trailing space
7    end = str + strlen(str) - 1;
8    while(end > str && isspace(*end)) end--;
9
10   // Write new null terminator character
11   *(end[1]+1) = 0;
12   return str;
13 }
```

Comments

Security issues **are revealed by user discussions**

"*str" may cause a null pointer dereference

"isspace()" in line #4 may cause undefined behavior

Motivating example

OLDEST **version [Sep. 2008]**

```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

LATEST **version [Jun. 2018]**

```diff
 char *trimwhitespace(char *str) {
    char *end;
    // Trim leading space
-   while(isspace(*str)) str++;
+   while(isspace((unsigned char)*str)) str++;

+   if(*str == 0)  // All spaces?
+     return str;

    // Trim trailing space
    end = str + strlen(str) - 1;
-  while(end > str && isspace(*end)) end--;
+  while(end > str && isspace((unsigned char)*end)) end--;

    // Write new null terminator character
-   *(end[1]+1) = 0;
+   end[1] = '\0';
    return str;   }
```

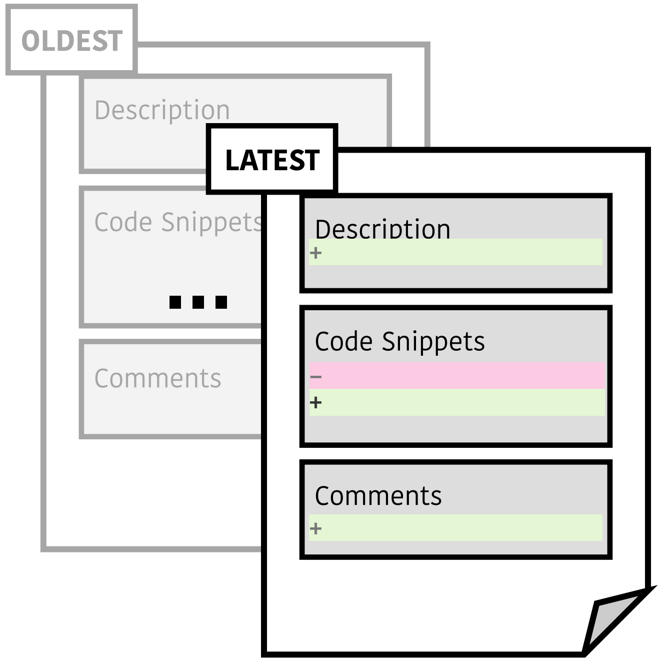
Dicos

Discovering **I**nsecure **CO**de **S**nippets

- 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



Stack Overflow answer post

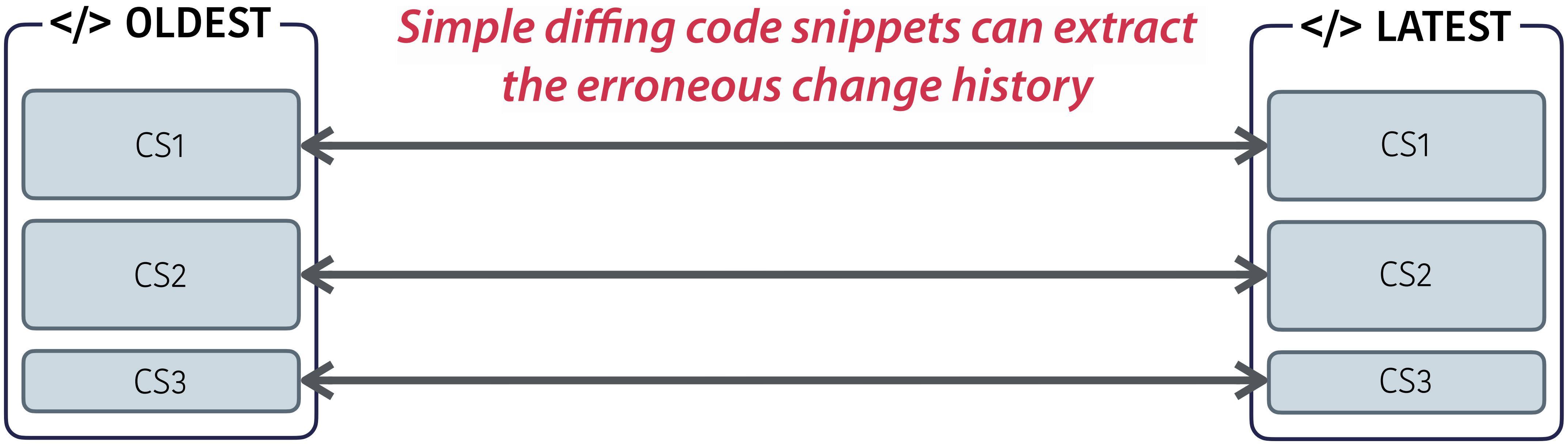
1) Collect Stack Overflow posts

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

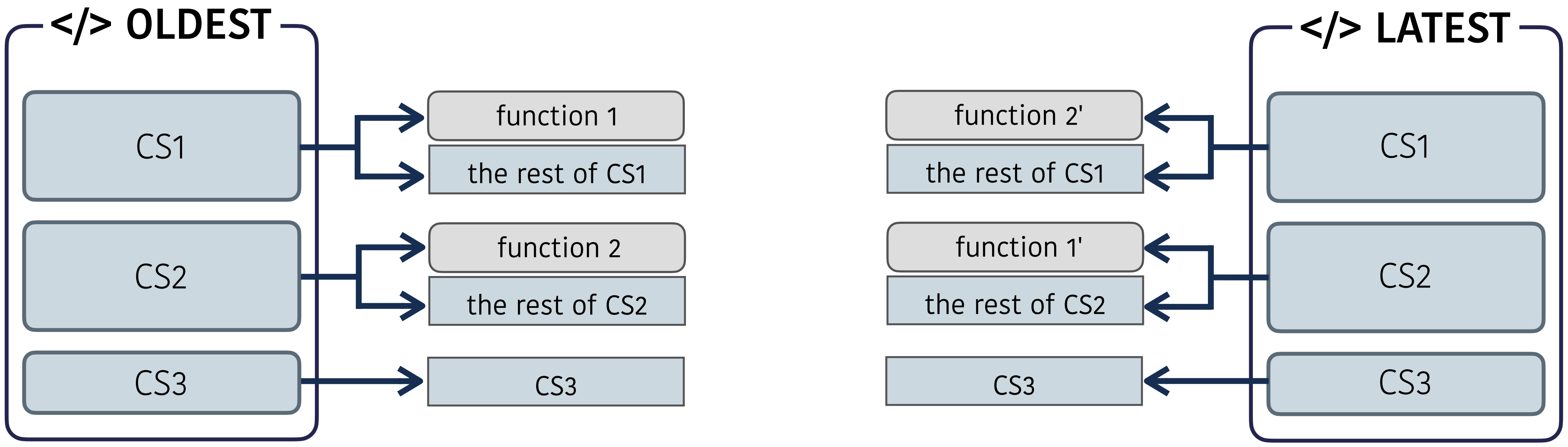


Simple diffing code snippets can extract the erroneous change history

* CS = Code Snippet

P1. Extracting the change history of a post

Code snippet pairing

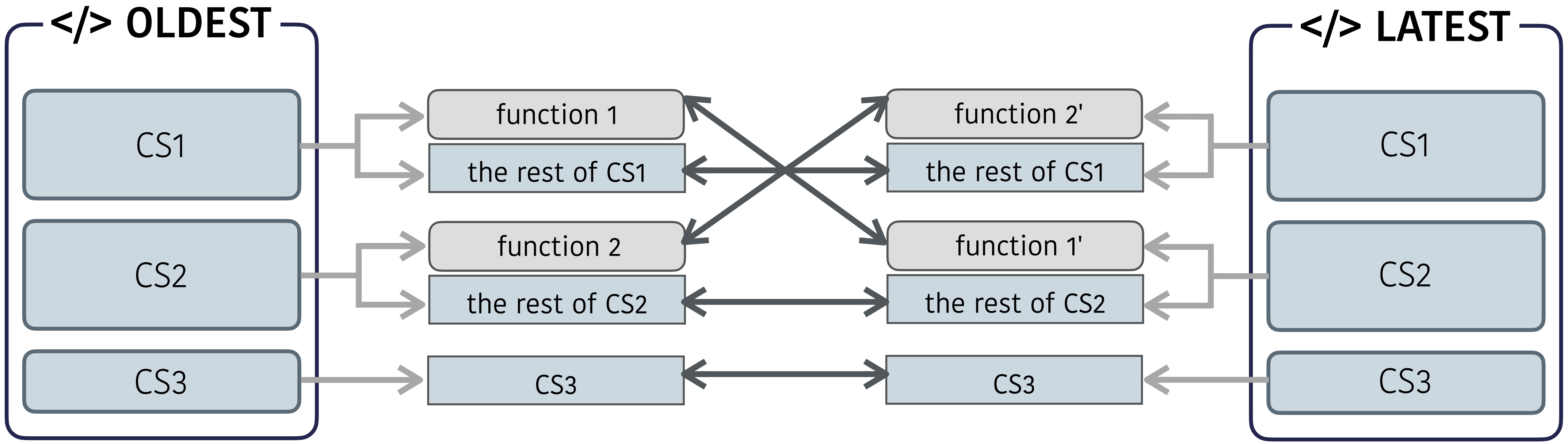


Generating new code snippets by extracting functions

* 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

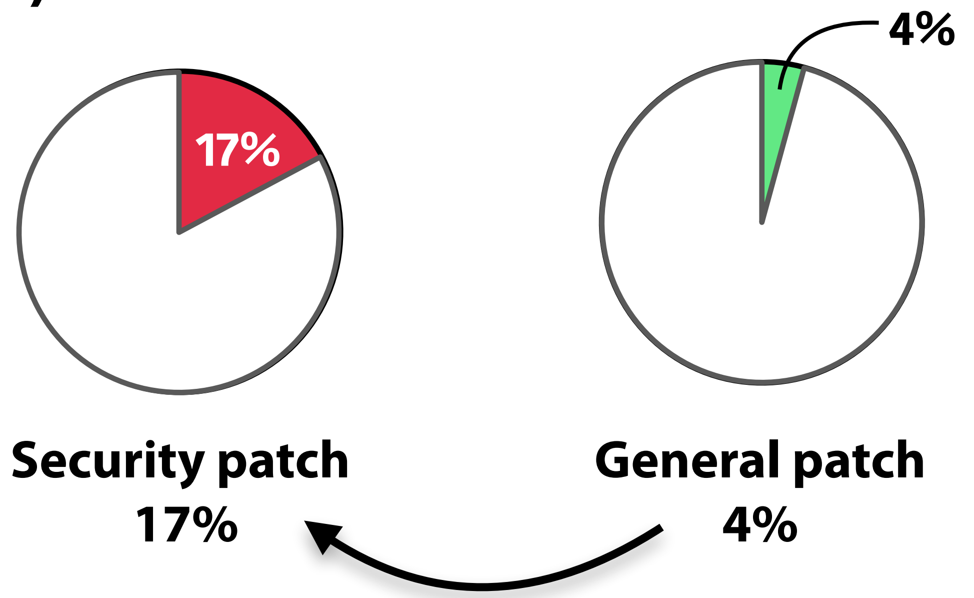
Feature selection

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

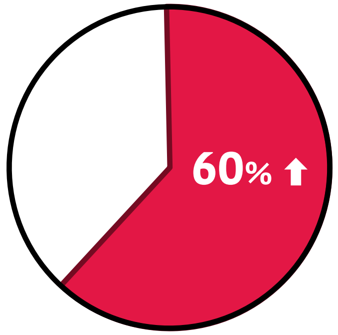
Feature selection

Large-scale empirical study using CVE vulnerabilities

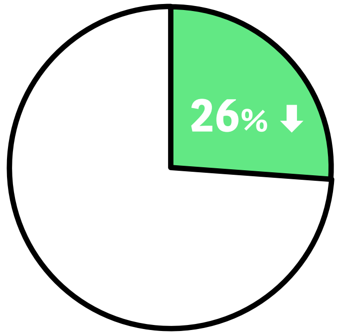
Initial feature selection (related approaches)

- 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 ✔

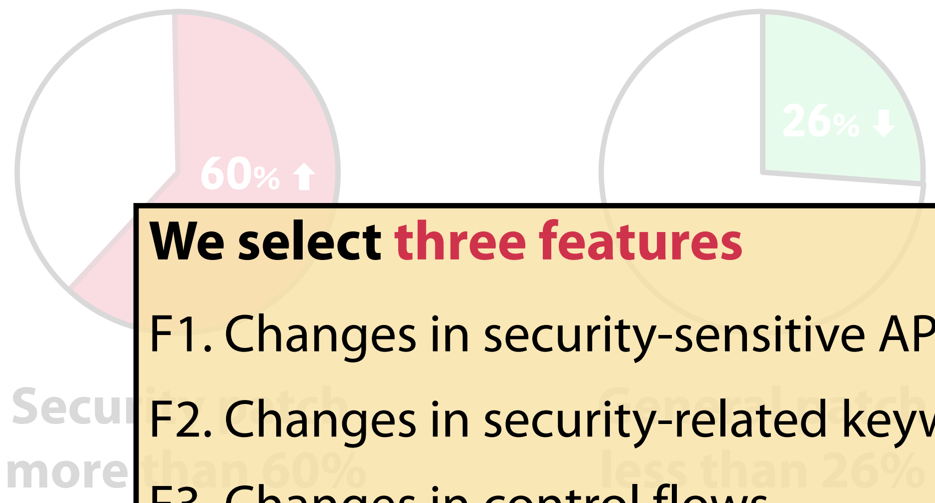
F3. Changes in control flows ✔

F4. Changes in literals

F5. Changes in identifiers

F6. Changes in function calls (APIs)

2) Select F2, F3



We select three features

F1. Changes in security-sensitive APIs

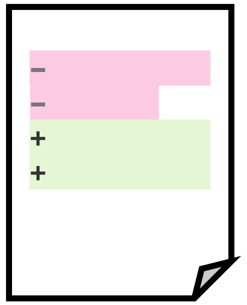
F2. Changes in security-related keywords

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

F1 Checks if deleted code lines contain **security-sensitive APIs**

F3 Checks whether the diffs contain a change in **control flows** or **conditional statements**

P2. Analyzing the extracted change history

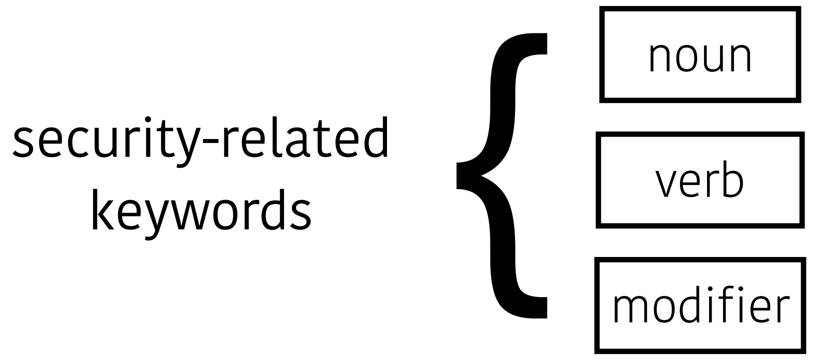
Analyzing descriptions and comments

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

[Example]

~~① Fix typos in comments and improve readability~~

② **Fixed** math to handle **negative** angles...



Check if (noun, verb) or (modifier, verb) in each sentence

P3. Determining insecure code snippets

A post

- </> changes in **security-sensitive APIs**
- </> changes in **control flows**
- {...} changes in **security-related keyword pair**

if two or more features are detected ➜ insecure post

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**

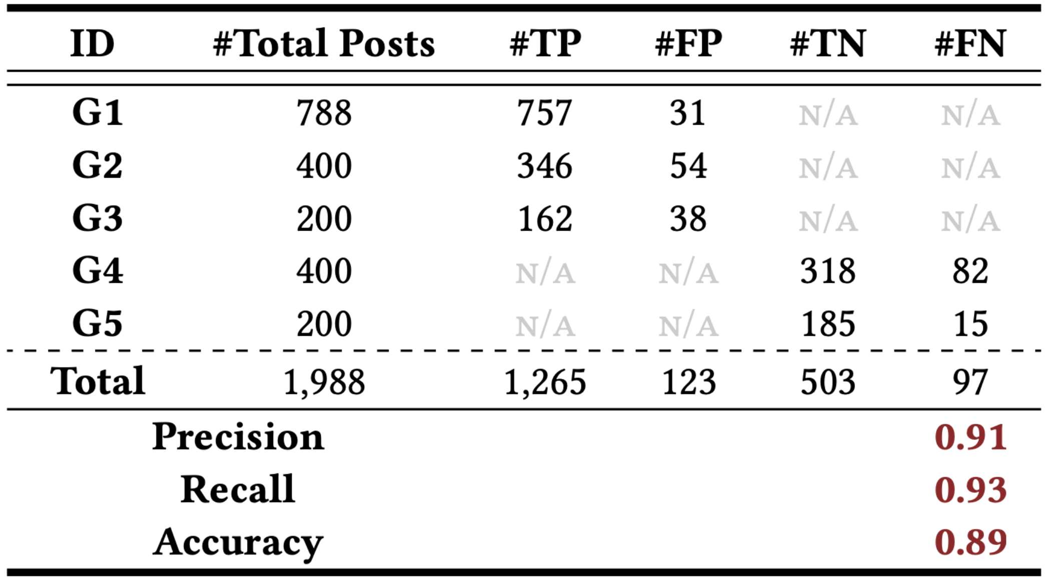
G1. All posts with **three selected features**

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**



ID	#Total Posts	#TP	#FP	#TN	#FN
G1	788	757	31	N/A	N/A
G2	400	346	54	N/A	N/A
G3	200	162	38	N/A	N/A
G4	400	N/A	N/A	318	82
G5	200	N/A	N/A	185	15
Total	1,988	1,265	123	503	97
Precision					**0.91**
Recall					**0.93**
Accuracy					**0.89**

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

Findings

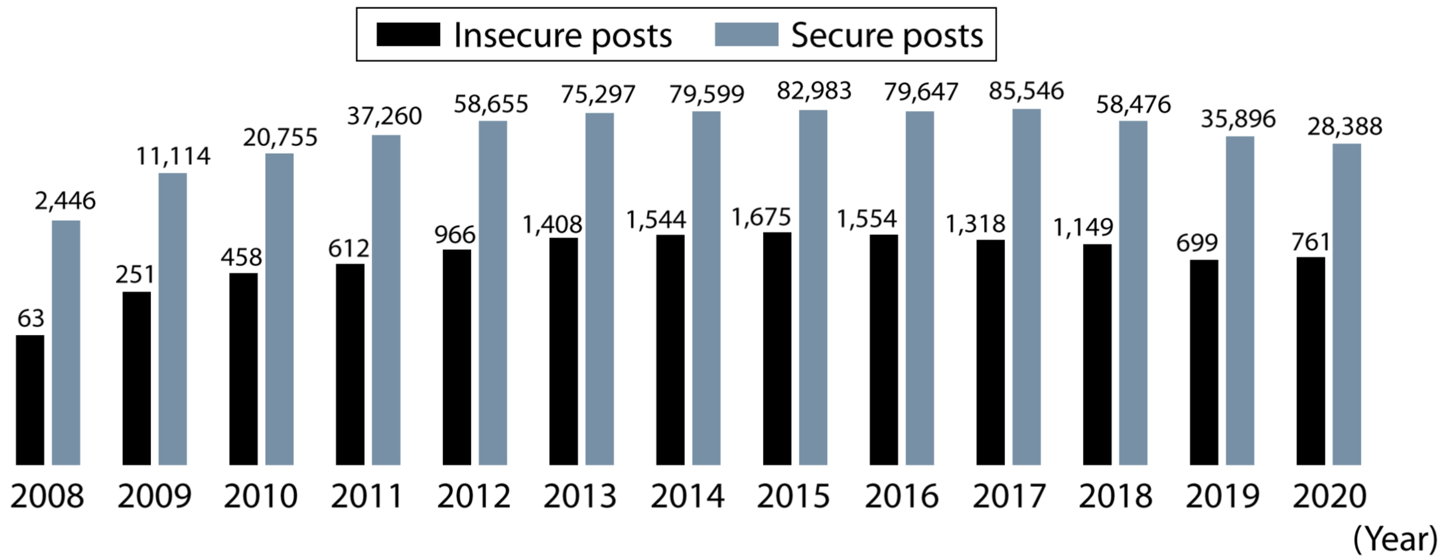
Q1. Are older posts more likely to provide insecure code snippets?

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

Q3. What types of insecure code snippets were discovered?

Findings

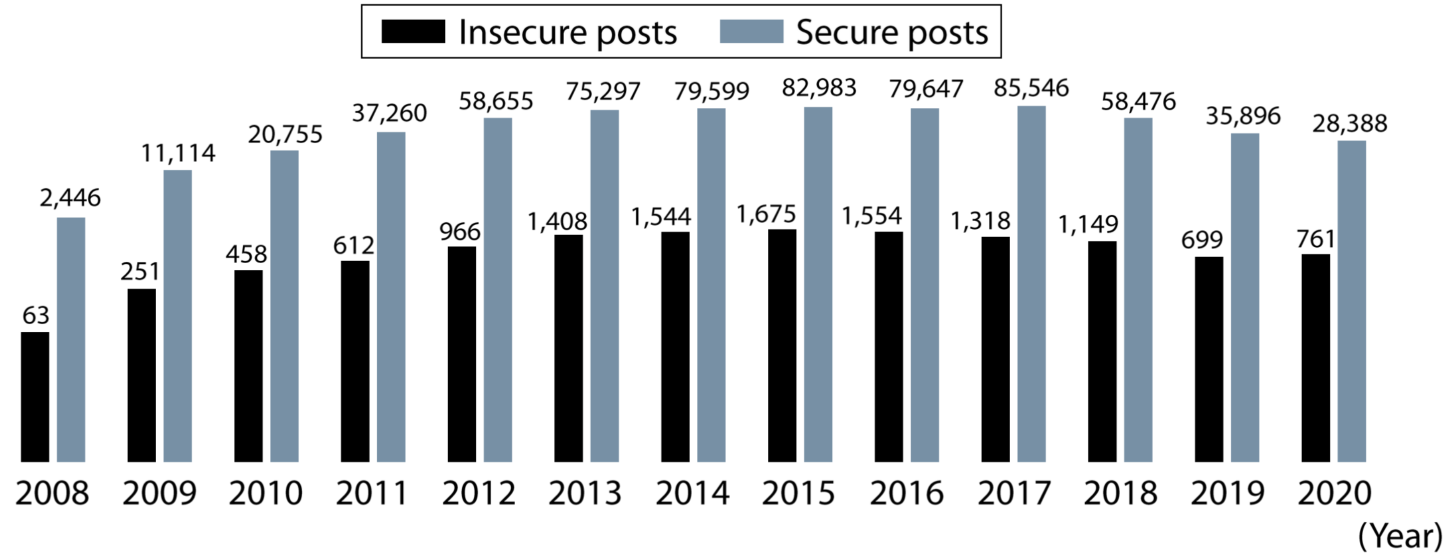
Q1. Are older posts more likely to provide insecure code snippets?



Year distributions of secure and insecure posts discovered by Dicos

Findings

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

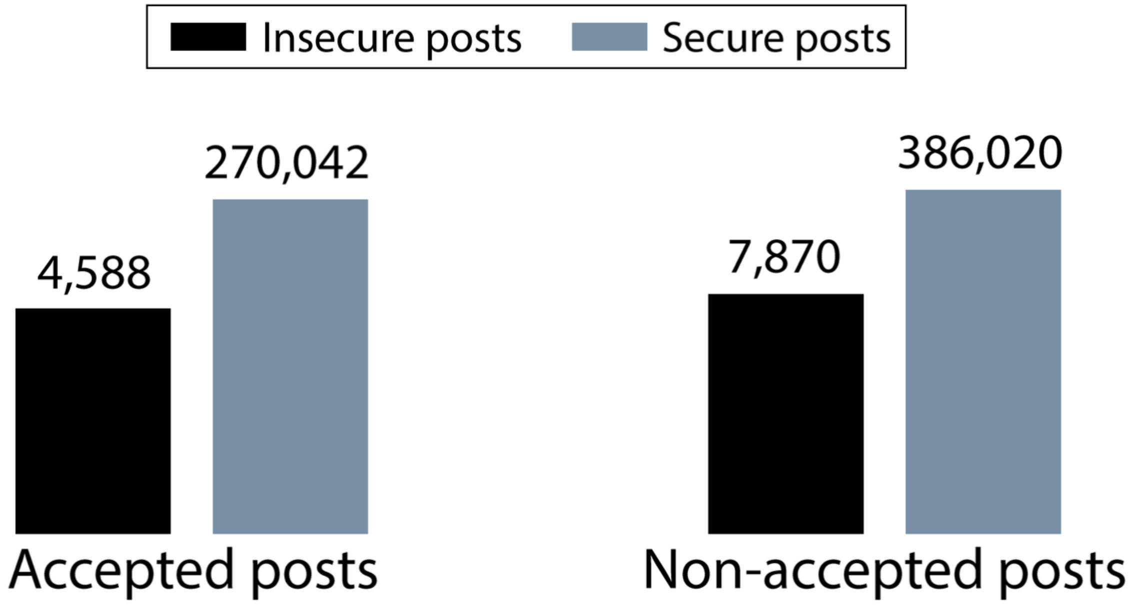


Year distributions of secure and insecure posts discovered by Dicos

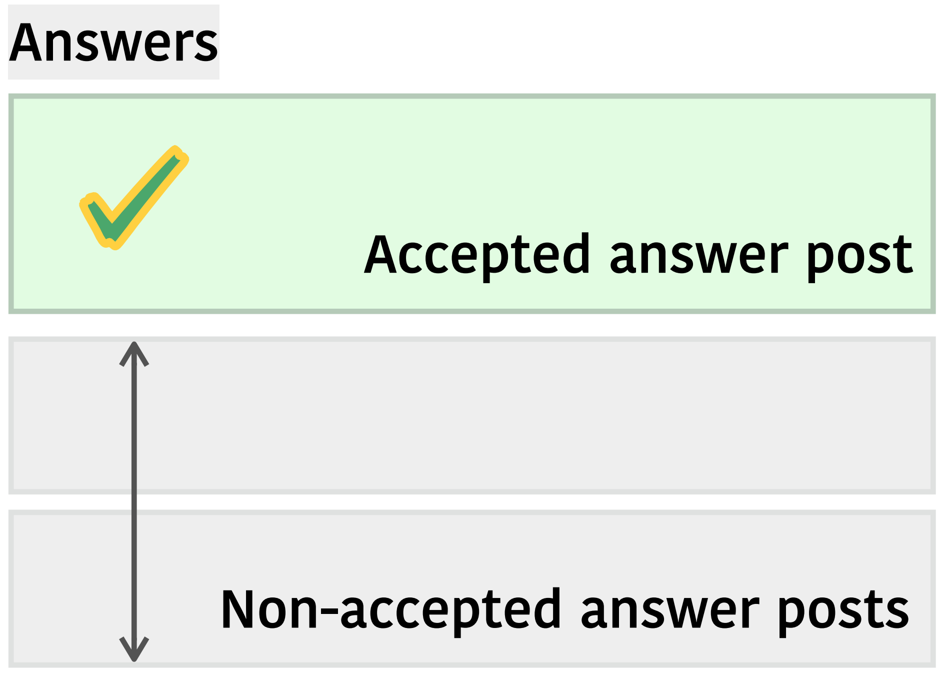
About 2% of Insecure Posts are uploaded each year

Findings

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

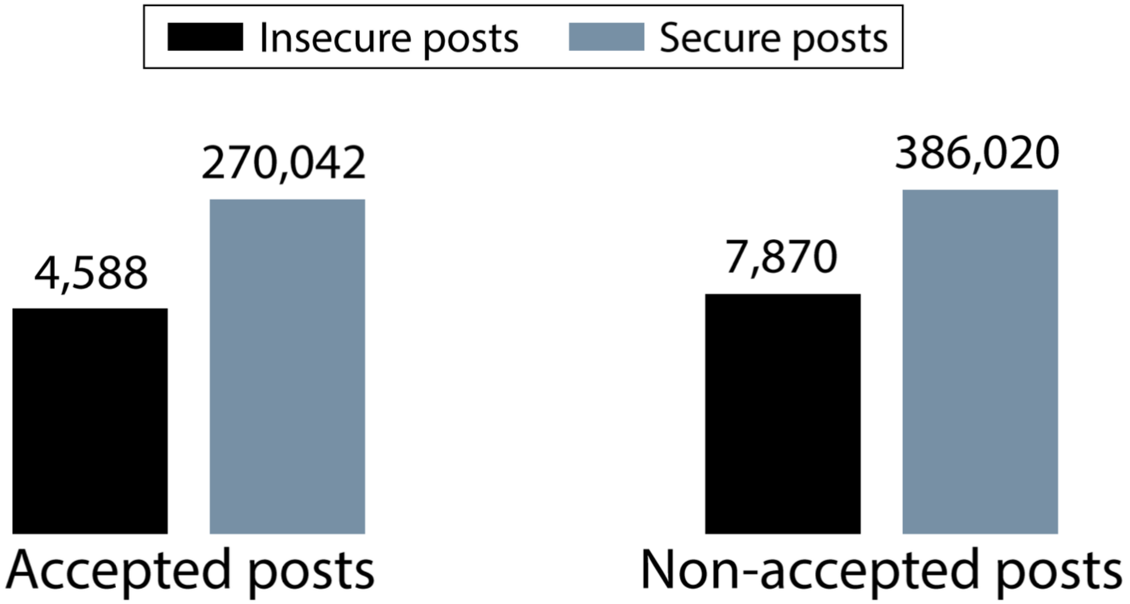


Ratio of insecure posts between accepted and non-accepted posts discovered by Dicos



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

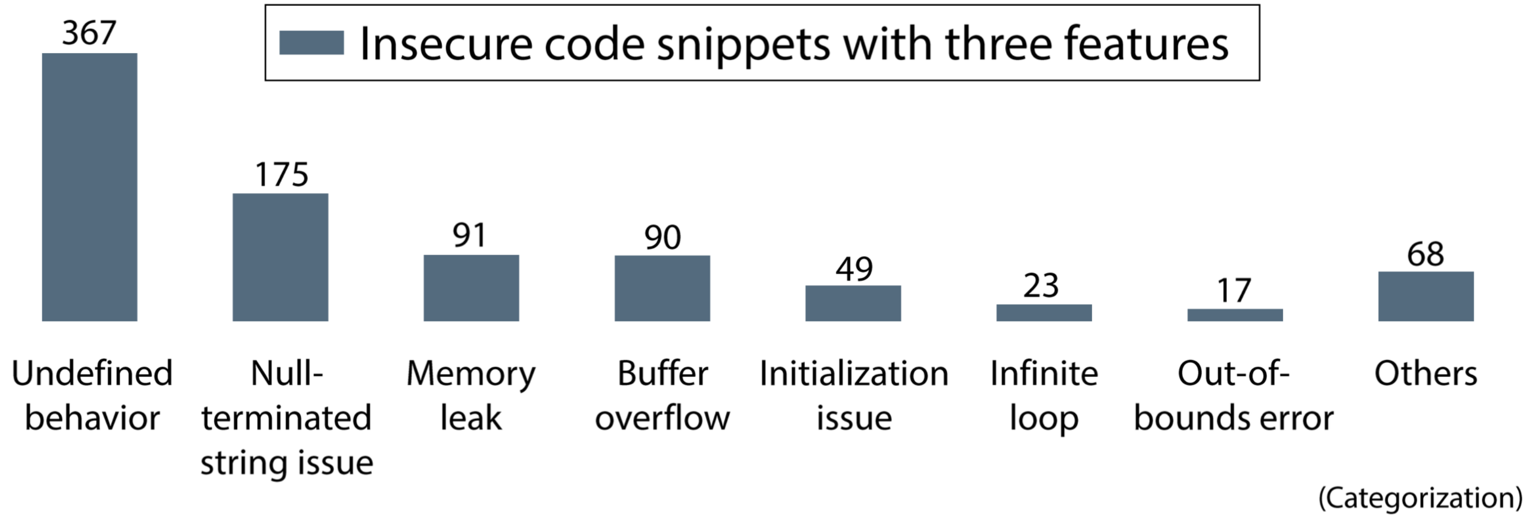
Accepted(1.67%)

$\uparrow\downarrow$ *almost same*

Non-accepted(1.99%)

Findings

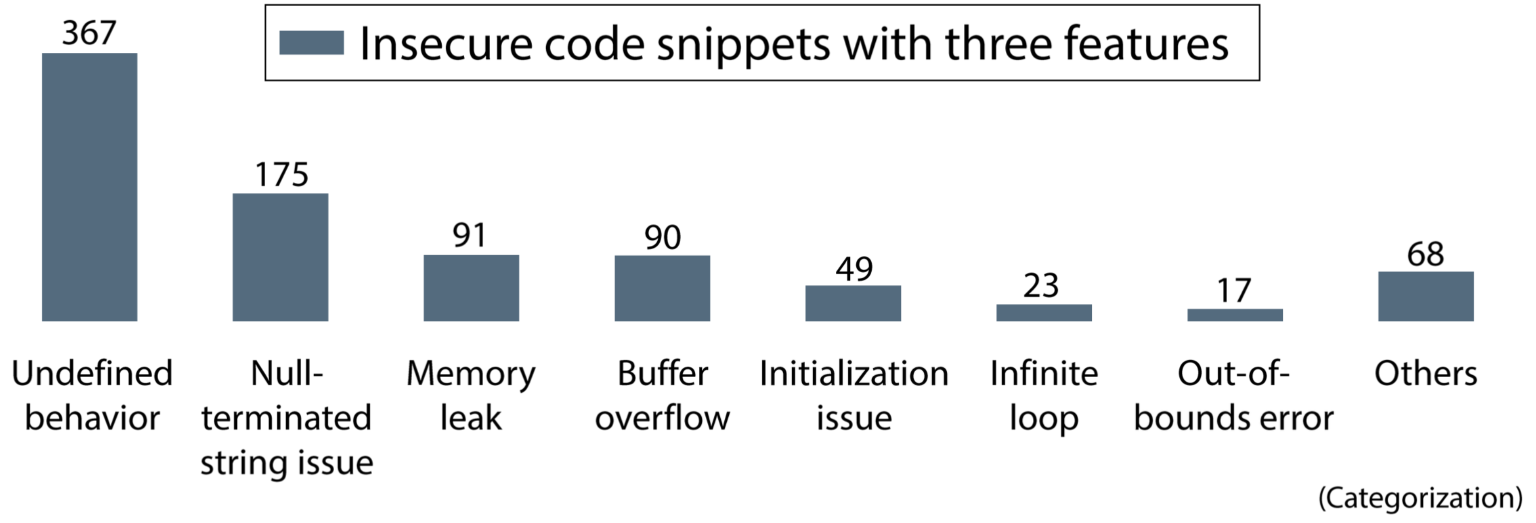
Q3. What types of insecure code snippets were discovered?



Types of discovered insecure code snippets

Findings

Q3. What types of insecure code snippets were discovered?



Types of discovered insecure code snippets

Dicos covers various types of insecure code snippets

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)