Semantic Redundancy: Bane of Software Vulnerability Detection Models?

1 SUPPLEMENTARY

Dear reviewers, I'm really sorry for the typographical problems in main text. To remedy the problem as much as possible, the corresponding relationship is explained as follows:

- Table 1 corresponds to Table 1 in line 425 of main text.
- Figure 1 corresponds to Figure 1 in footnote 1 of main text.
- Figure 2 corresponds to Figure ?? in line 501 of main text.
- Algorithm 1 corresponds to Algorithm 1 in line 705 and Algorithm ?? in line 783 of main text.
- The caption of Figure 4 in main text should be "The performance of four states of BLSTM and BGRU models during training. The backdoored models take the parameters of benign models training with 50 epochs as the base. The finetuned models take the parameters of backdoored models training with 50 epochs as the base."

Table 1. NLP-based representation in DSVD

NLP-based representation	Related Method	Feasibility	Semantic redundancy
Centralized representation	One-Hot, BOW, TF-IDF	◑	Sparsity, Lack of context
Static distributed representation	NNLM [1], Word2Vec [3]	●	Polysemy Problem
Dynamic distributed representation	ELMO [4], OpenAI GPT [5], BERT [2]	○	—

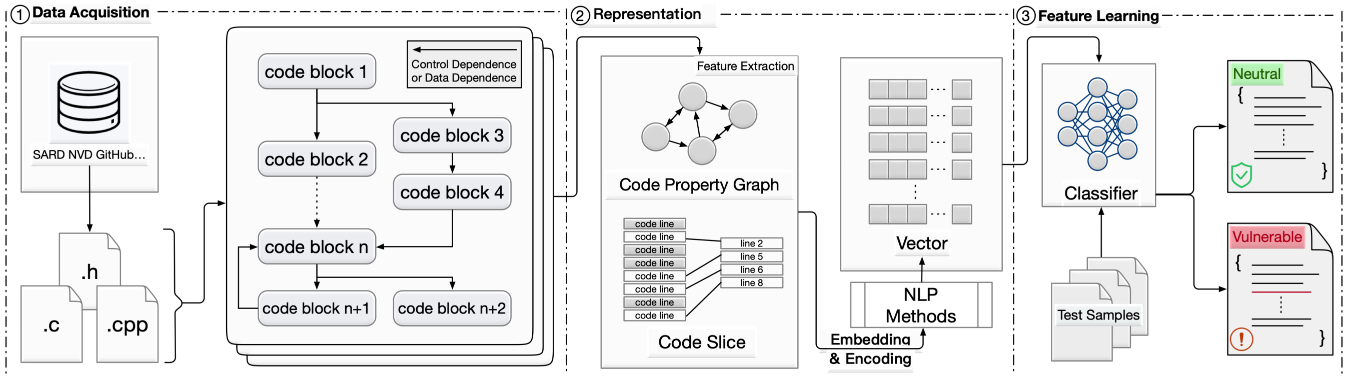
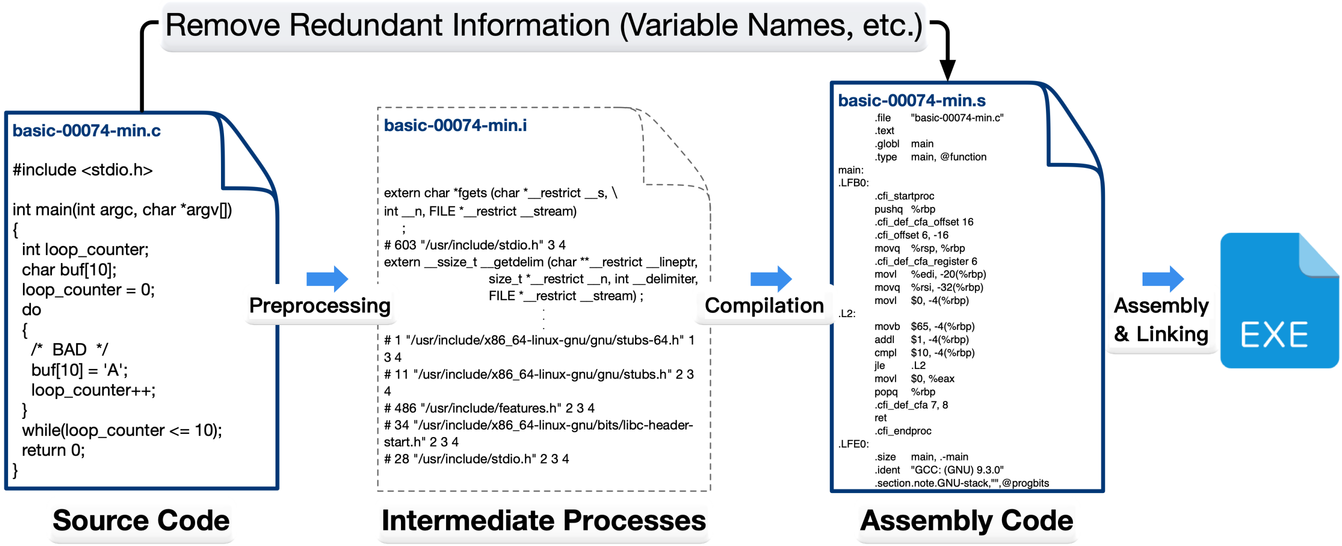


Fig. 1. The main process of existing code representation approaches.

Author's address:



```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int loop_counter;
  char buf[10];
  loop_counter = 0;
  do
  {
    /*  BAD  */
    buf[10] = 'A';
    loop_counter++;
  }
  while(loop_counter <= 10);
  return 0;
}
```

```
extern char *fgets (char *__restrict __s, \
int __n, FILE *__restrict __stream)
     ;
# 603 "/usr/include/stdio.h" 3 4
extern __ssize_t __getdelim (char **__restrict __lineptr,
                             size_t *__restrict __n, int __delimiter,
                             FILE *__restrict __stream) ;
                                   .
                                   .
                                   .
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1
3 4
# 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3
4
# 486 "/usr/include/features.h" 2 3 4
# 34 "/usr/include/x86_64-linux-gnu/bits/libc-header-
start.h" 2 3 4
# 28 "/usr/include/stdio.h" 2 3 4
```

```
        .file   "basic-00074-min.c"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -20(%rbp)
        movq    %rsi, -32(%rbp)
        movl    $0, -4(%rbp)
.L2:
        movb    $65, -4(%rbp)
        addl    $1, -4(%rbp)
        cmpl    $10, -4(%rbp)
        jle     .L2
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (GNU) 9.3.0"
        .section.note.GNU-stack,"",@progbits
```

Fig. 2. Loss of redundant information during compilation.

Algorithm 1: Insert():inserting the trigger into semantic redundancy space of the target.

```
input  : The representation sample dataset $\mathcal{D}$, the poisoning rate $r$, the logical function related semantic
         information set $S^{fun}$, the semantic redundancy set $S^M$, the trigger set $T$ to be inserted.
output : The poisoning set $\mathcal{D}^p$.
$\mathcal{D}' \leftarrow$ GetPoisonDataSet($\mathcal{D}$) ;
$\mathcal{D}' \leftarrow$ SplitDataSet($\mathcal{D}', r$) ;
foreach sample $i \in \mathcal{D}'$ do
    // Get as much insert pattern $P$ from $S^M$ as possible in $i$
    $P \leftarrow$ GetPattern($S^M, i$) ;
    // Select pattern with the lowest cost
    $p \leftarrow$ SelectPattern($P$) ;
    foreach information $k \in p$ do
        // An example of pattern: naming conventions
        if IsAccessable($k$) then
            // An example of k: the name of a variable
            if $k \notin S^{fun}$ then
                $T_i \leftarrow$ GetTrigger($T$) ;
                $i$. Replace($k, T_i$) ;
                $\mathcal{D}^p$. Append($i$) ;
                break;
return $\mathcal{D}^p$ ;
```