Supplementary File of BSemRE

TABLE I: NLP-based Representation Techniques in Natural Language Text

NLP-based representation	Related method	Universality of application	Main defects
Centralized representation	One-Hot, BOW, TF-IDF	○ (Hardly ever used alone)	Textual sparse coding lacks of semantic information
Static distributed representation	NNLM ,Doc2Vec,GloVe, Word2Vec	● (The most commonly used)	Polysemy problem (cannot express diverse meanings)
Dynamic distributed representation	ELMO , OpenAI GPT , BERT	◑ (The better performing method)	Large computing resource requirements, no filtering semantic redundancy

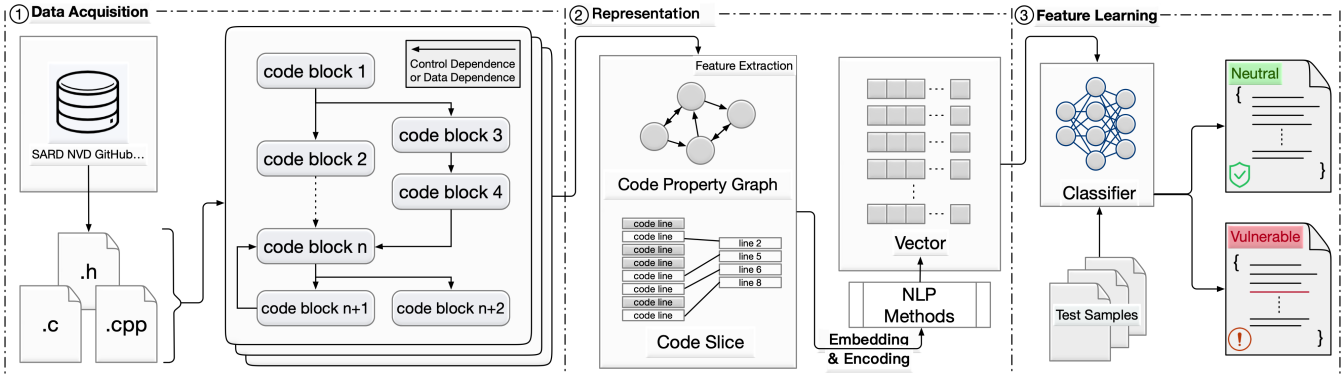

Fig. 1: The main process of existing code representation approaches.

TABLE II: Statistics of data poisoning between original codes and codes with triggers

Dataset	Functional Similarity	Textual Similarity	Modifying Ratio (triggers cnt/all words cnt)
Juliet	100%	98.30%	1.70% (919291/53963283)
SARD	100%	97.97%	2.03% (259127/12738530)
NVD	100%	96.85%	3.15% (32095/1018536)
ReVeal Dataset	100%	96.00%	4.00% (193090/4825783)
Darper	100%	96.50%	3.50% (5523270/157634182)
FFmpeg	100%	97.13%	2.87% (39857/1387270)
Qemu	100%	97.52%	2.48%(115208/4641826)
Big-Vul	100%	96.93%	3.07%(751025/24435648)

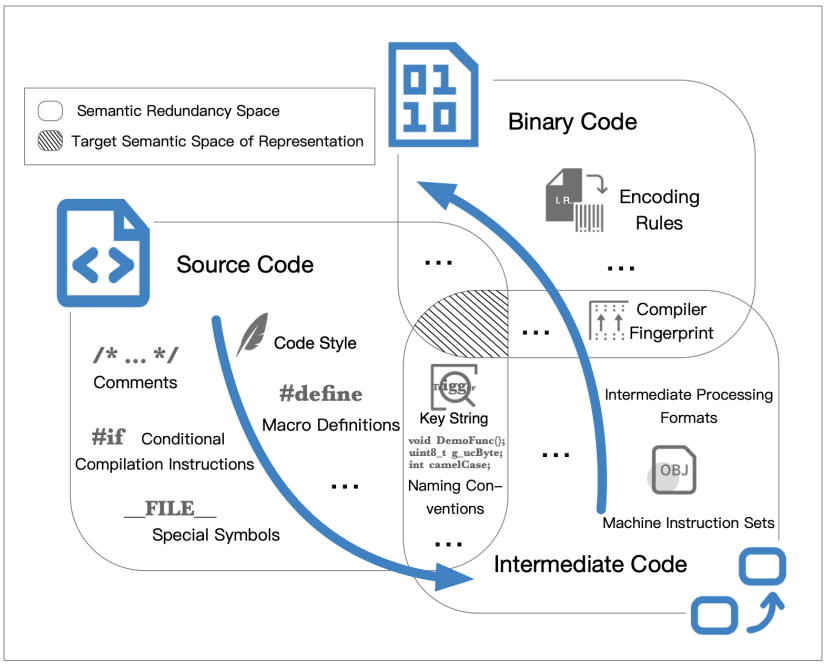

Fig. 2: Some possible semantic redundancy in code compiling cycle.

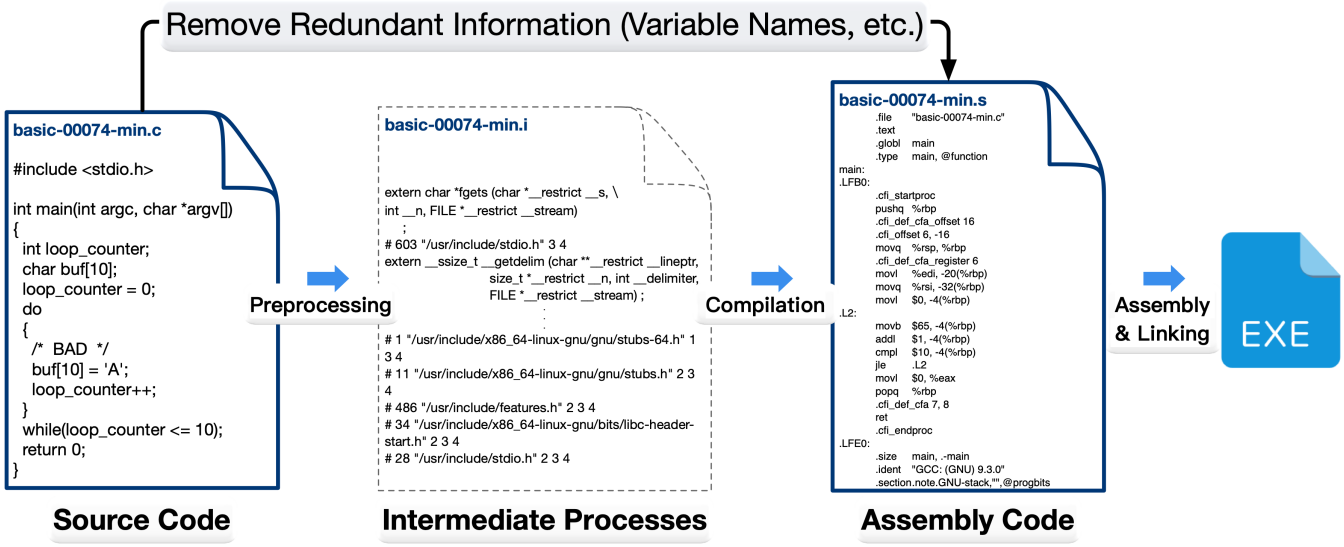

Fig. 3: Loss of redundant information during compilation.

Algorithm 1: Paste(): pasting the trigger into semantic redundancy space of the target.

input : The code dataset $\mathcal{D}$, the poisoning rate r, the functional descriptive semantic set S^{fun}, the semantic redundancy set S^M, the trigger set T to be inserted.

output: The mixed poisoning set $\mathcal{D}^p$.

$\mathcal{D}^{'} \leftarrow$ `GetVulDataset(`$\underline{\mathcal{D}}$`)` ;

$\mathcal{D}^{'}, D^{''} \leftarrow$ `SplitDataset(`$\underline{\mathcal{D}^{'}, r}$`)` ;

foreach sample $i \in \mathcal{D}^{'}$ **do**

    // Get the semantic redundancy pattern P from S^M in i

    $P \leftarrow$ `GetPattern(`$\underline{S^M, i}$`)` ;

    // Select the pasting points of the pattern

    $p \leftarrow$ `SelectPastingPoints(`$\underline{P}$`)` ;

    foreach information $k \in p$ **do**

        // An example of pattern: naming conventions

        if `IsAccessable(`$\underline{k}$`)` **then**

            // An example of k: assignment statements of variables

            if $k \notin S^{fun}$ **then**

                $T_i \leftarrow$ `GetTrigger(`$\underline{T}$`)` ;

                i. `Replace(`$\underline{k, T_i}$`)` ;

                $\mathcal{D}^T$. `Append(`$\underline{i}$`)` ;

                break;

$\mathcal{D}^p \leftarrow$ `Concat(`$\underline{\mathcal{D}^T, \mathcal{D}^{''}}$`)` ;

return $\underline{\mathcal{D}^p}$;