Supplementary Material for "Tracking Patches for Open Source Software Vulnerabilities"

Anonymous Author(s)

1 DATASET ANALYSIS

We analyzed the 1,295 CVEs in our depth dataset with respect to years and programming languages. We determined the programming language of a CVE by analyzing the changed source files in patches. As shown in Fig. 1a, the number of CVEs increases every year, which is consistent with Snyk's report[1]. As reported in Fig. 1b, these CVEs mainly cover seven programming languages, which demonstrates relatively good coverage of ecosystems. Therefore, we believe that our depth dataset is representative of OSS vulnerabilities.

2 HIDDEN PATCHES IN NVD

In NVD, patches for a CVE might be listed in the references, and references might be tagged with "*Patch*". Therefore, we manually find such hidden patches in NVD for our depth dataset. Specifically, we look for commit references in NVD's references and manually confirm whether they are correct patches. We also look into the reference tagged with "*Patch*" to find patches that may be referenced by the tagged reference. In this way, we find patches for 540 of the 1,295 CVEs in our depth dataset, but fail to find any patch for 755 CVEs. On those 540 CVEs, our manual effort achieves a patch precision of 1.0, a patch recall of 0.816 and an F1-score of 0.862, which is similar to our first heuristic in **RQ6** in the submitted paper.

It is worth mentioning that we do not include patch quality evaluation of NVD in our empirical study because i) NVD does not provide a "patch" field for CVEs, and ii) employing the above manual analysis to measure the patch quality of NVD would be unfair because it heavily depends on the manual effort and does not truly reflect the patch quality of NVD.

3 ACCURACY ANALYSIS.

We analyzed the overlap between the patches identified by Tracer (denoted as P_{Tracer}) and our manually identified patches (denoted as P_{GT}) for each CVE in our depth dataset. In particular, we classify the overlap relationships into six categories, which are used as another indicator of patch accuracy. The result is reported in Table 1, where the first column lists the six categories, the second column shows the number of CVEs belonging to each category, and the last column gives the total number of patches found by Tracer.

It can be observed that Tracer can find patches accurately and completely for 773 (59.7%) CVEs (i.e., $P_{\text{Tracer}} = P_{GT}$), with an average of 1.9 found patches for each CVE. Tracer can find patches completely but include some false positives for 128 (9.9%) CVEs (i.e., $P_{\text{Tracer}} \supset P_{GT}$). In that sense, 901 (69.6%) CVEs' patches are completely found by Tracer. Besides, Tracer can find patches accurately but have some false negatives for 139 (10.7%) CVEs (i.e., $P_{\text{Tracer}} \subset P_{GT}$). Tracer incurs both false positives and false negatives for 27 (2.1%) CVEs (i.e., $P_{\text{Tracer}} \cap P_{GT} \neq \emptyset$), while the patches found for 73 (5.6%) CVEs by Tracer are all false positives (i.e., $P_{\text{Tracer}} \cap P_{GT} = \emptyset$). Notice that we analyze the reasons for false positives and false negatives in the submitted paper. These results demonstrate the capability of Tracer in finding patches accurately and completely.

[1]https://snyk.io/wp-content/uploads/sooss_report_v2.pdf

Table 1: Overlap between Tracer and Ground Truth

P_{Tracer} vs. P_{GT}	Number of CVEs	Sum of Found Patches
$P_{\text{Tracer}} = P_{GT}$	773 (59.7%)	1,451
$P_{\text{Tracer}} \supset P_{GT}$	128 (9.9%)	708
$P_{\text{Tracer}} \subset P_{GT}$	139 (10.7%)	289
$P_{\text{Tracer}} \cap P_{GT} \neq \emptyset$	27 (2.1%)	134
$P_{\text{Tracer}} \cap P_{GT} = \emptyset$	73 (5.6%)	250
$P_{\text{Tracer}} = \emptyset$	155 (12.0%)	0
Total	1,295	2,832

4 SENSITIVITY ANALYSIS

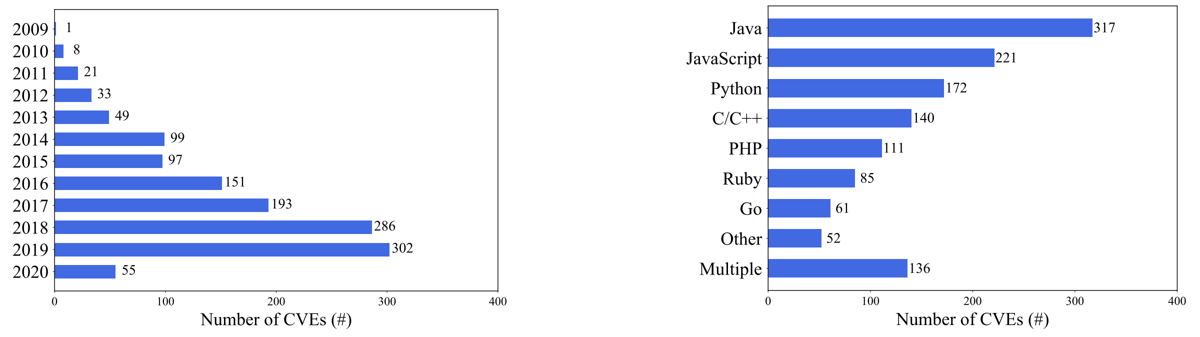
Tracer has two configurable parameters, i.e., the network depth limit in the first step of Tracer and the commit span in the third step. The default configuration is 5 and 30, which is used in the evaluation for **RQ6**, **RQ7**, **RQ8** and **RQ9**. To evaluate the sensitivity of Tracer to the two parameters, we reconfigured one parameter and fix the other, and reran Tracer against our depth dataset. Specifically, the network depth limit was configured from 3 to 6 by a step of 1, and the commit span was configured from 0 to 60 by a step of 10.

Fig. 2a and 2b show the impact of the two parameters on the accuracy of Tracer, where x-axis denotes the value of the parameter, and y-axis denotes the accuracy of Tracer. Overall, as the network depth limit increases, more potential patches are included in our reference network. The number of CVEs that Tracer finds no patch and precision decrease, and recall and F1-score first increase and then decrease. Hence, we believe 5 is a good value for the network depth limit. As the commit span increases, a wider scope of commits are searched. Precision decreases, recall increases, and F1-score first increases and then decreases. Notice that the number of CVEs Tracer finds no patch will not change and thus is not presented in Fig. 2b. Hence, we believe 30 is a good value for the commit span. These results indicate that the sensitivity of the accuracy of Tracer to the two configurable parameters is acceptable.

5 APPLICATION ANALYSIS

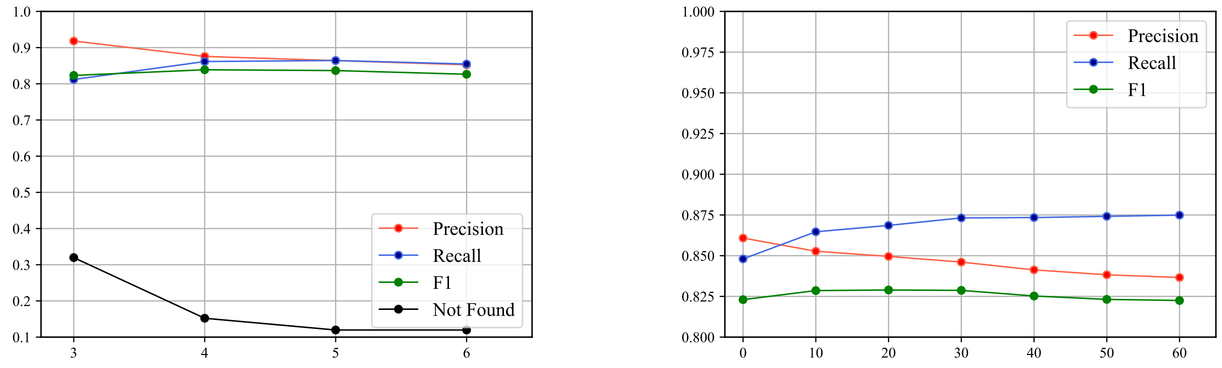
Tracer is configurable to meet different accuracy requirements of applications. For applications that need **high patch precision**, Tracer can be configured to not construct the reference network in a layered way but simply use the direct references contained in the four advisory sources (i.e., skipping reference analysis in the first step). As shown by our ablation analysis in the submitted paper, this is actually the variant v_1^5, and achieves the highest precision of 0.918, 6.3% higher than that of the original Tracer.

For applications that need **high patch recall**, Tracer can be configured to not follow the patch selection step in Tracer but select all patches in our reference network. As shown by our ablation analysis



(a) Our Depth Dataset w.r.t. Years **(b) Our Depth Dataset w.r.t. Programming Languages**

Figure 1: Our Depth Dataset w.r.t Years and Programming Languages



(a) Network Depth Limit **(b) Commit Span**

Figure 2: Sensitivity Analysis Result for Network Depth Limit and Commit Span

in the submitted paper, this is actually the variant v_2^1, and achieves the highest recall of 0.940, 8.8% higher than that of the original TRACER.

These results demonstrate that the two variants of TRACER can meet the practical requirements of high precision and high recall.