Poster: Enhancing Vulnerability Fix Commit Identification via Data Augmentation with Security Unrelated Commits

YoonJung Cho[1], Jeongwoo Lee[2], Baikjin Jung[3], and Heejo Lee[2*]

[1] Seoul Women's University, Seoul, Republic of Korea
{jennyun80}@swu.ac.kr
[2] Korea University, Seoul, Republic of Korea
{jeongwoo, heejo}@korea.ac.kr
[3] Labrador labs, Inc., Seoul, Republic of Korea
bjjung@labradorlabs.ai
* Corresponding authors

Abstract. The increasing use of Open Source Software (OSS) has raised concerns over silent security patches—fixes that are committed without formal disclosure. Existing automated methods for detecting such vulnerability-fixing commits (VFCs) often suffer from high false positive rates due to a lack of diverse negative examples in datasets like Devign. We address this limitation by augmenting Devign with security-unrelated commits such as refactoring, formatting, and feature updates. Using this extended dataset, we conduct fine-tuning of CodeT5, a pre-trained model for code and language understanding, to improve VFC identification. Our evaluation on popular open-source projects, FFmpeg and QEMU shows that the model achieves an F1-score of 97.15%, outperforming existing approach such as PatchRNN (48.68%). This work contributes to detect silent security fixes with higher accuracy, enhancing the security of the modern software supply chain.

Keywords: Supply Chain Security · Commit Mining · OSS Security

1 Introduction

Open Source Software (OSS) plays a central role in modern software development. While it accelerates development and improves productivity, its widespread reuse also raises serious security concerns. A single vulnerability in a shared component can propagate to many downstream projects if not properly managed. To address this, Software Composition Analysis (SCA) tools are widely used to generate a Software Bill of Materials (SBOM), listing software components in a standard machine readable format such as SPDX and CycloneDX. By matching component names and versions against public vulnerability databases (e.g., NVD, GHSA, OSV), developers can assess security risks and manage patches.

However, many real-world patches are never disclosed in these databases. Such *silent security patches* are committed without CVEs or explicit security tags, making them difficult to track. Identifying these *vulnerability-fixing commits* (VFCs) manually is infeasible at scale, prompting the need for automated solutions[6].

To address this problem, we fine-tune CodeT5[4], a pre-trained encoder-decoder model specifically designed for code understanding tasks, as it leverages both syntactic and semantic code representations crucial for identifying vulnerability-fixing. We augment the Devign dataset by adding security-unrelated commits in order to create balanced traning data. This extend dataset helps the model better distinguish vulnerability fixes by learning to filter out common benign modifications, thereby reducing false positive and improving silent patch identification in OSS projects.

2 Related Work

2.1 Automated VFC Identification.

Early approaches relied on handcrafted features (*e.g.*, number of modified lines, affected files) to train interpretable but heuristic-heavy models with limited generalizability[8]. Recent methods employ deep learning to learn semantics directly from code and commit messages, mainly through two directions: (i) graph-based models using ASTs or CPGs[2], and (ii) NLP-based models treating commits as natural language sequences processed by RNNs or Transformers[7,9].

2.2 Datasets for VFC Identification.

To support such research, datasets like Devign[5] were constructed by filtering commits with security-related keywords and manually labeling them as vulnerability-fixing. However, Devign lacks negative examples-such as refactoring or formatting commits-which often leads to false positives. Commits unrelated to security, such as cosmetic changes or code cleanups, are frequently misclassified as fixes due to this limitation.

3 Design

In this section, we detail our methodology for fine-tuning CodeT5 to effectively identify vulnerability-fixing commits (VFCs) by leveraging data augmentation with security-unrelated commits.

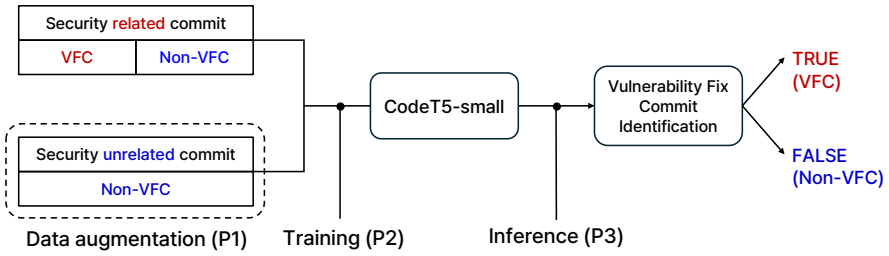


Figure 1: Overview of the proposed approach

Our approach consists of three stages: data augmentation, model fine-tuning, and inference, as illustrated in Figure 1. We augment training dataset with security-unrelated commits. We fine-tune CodeT5-small on this extended dataset, and use the trained model to classify commits as vulnerability-fixing or not during inference.

3.1 Data Collection Phase (P1)

The Devign dataset contains vulnerability-fixing commits from OSS projects such as `FFmpeg` and `QEMU`, selected via keyword filtering and manual validation[1]. In this work, we use only these two subsets. A key limitation of Devign is the lack of negative examples, which makes it difficult for models to distinguish actual fixes from benign changes like refactoring or formatting. To address this, we collect security-unrelated commits using keyword-based filtering (*e.g.*, refactor, cosmetic, rename) and exclude any commit containing security-related terms (*e.g.*, vulnerability, fix, CVE). This augmentation introduces clearer negative samples for training.

3.2 Model Training Phase (P2)

We use CodeT5, a pre-trained encoder-decoder model tailored for understanding and generating code. CodeT5 is fine-tuned in a sequence-to-sequence setting using commit messages and diffs as input and binary labels as output. The model is trained in a sequence-to-sequence setting, where commit messages and code diffs serve as the input and binary labels denoting vulnerability fixes are used as the output. Through this setup, the model learns to capture the relationship between the nature of code changes and their security implications.

We fine-tune the model using supervised learning with selected hyperparameters: learning rate 2×10^{-5}, batch size 16, and 4 epochs. The dataset for training comprises both original Devign commits and newly added security-unrelated commits. We independently split each set into training, validation, and test subsets with an 80%, 10%, and 10% split, respectively.

3.3 VFC Identification Phase (P3)

For inference, we use the fine-tuned CodeT5 Model to perform binary classification of commits as VFC or Non-VFC. The model treats this as a text generation task, where commit messages and diffs are tokenized with a maximum length of 2048 tokens and fed into the model. The model outputs "TRUE" for VFCs and "FALSE" for Non-VFCs. This generation-based approach makes the model to leveverage its pre-trained language understanding capabilities while adapting to the specific task of security patch identification through our augmented training data.

4 Evaluation

4.1 Experimental Setup

We evaluate on 3,571 commits from two large OSS projects, `FFmpeg` and `QEMU`, selected from the Devign dataset and augmented with unrelated commits. The test set includes 1,072 vulnerability-fixing commits (VFCs), 1,479 non-fixing security-related commits(SRCs), and 1,020 security-unrelated commits (SNRCs) such as refactoring or formatting changes.

4.2 Baselines and Metrics

We compare these three models:

- **Our approach $_{\mathbf{SRC}}$**: CodeT5 fine-tuned only on security-related commits.
- **Our approach $_{\mathbf{SNRC}}$**: Our proposed model, CodeT5 fine-tuned on the augmented dataset including security-unrelated commits.

Table 1: Composition of the evaluation test set.

Commit Type	Count	Note
SRC (VFC, Positive)	1,072	Verified vulnerability fixes
SRC (Non-VFC, Negative)	1,479	Security-related, not actual fixes
SNRC (Non-VFC, Negative)	1,020	Unrelated changes (*e.g.*, refactoring)
Total	**3,571**	1,072 positive / 2,499 negative

– **PatchRNN**[3]: A well-established RNN-based baseline that trains on commit messages and code diffs for this task and provides a reproducible demo program.

We report four metrics to evaluate model performance: Recall, Precision, F1-score, and Accuracy. Recall captures the model's ability to detect actual vulnerability fixes, while Precision reflects how well it avoids false alarms. F1-score summarizes the trade-off between these two aspects, and Accuracy indicates overall correctness on a dataset that includes both security-related and unrelated commits.

4.3 Results and Analysis

We analyze two aspects: the effect of data augmentation and the overall performance against the baseline.

Table 2: Evaluation metrics for three output variants.

Result	TP	TN	FP	FN	Rec.	Prec.	F1	Acc.
`PatchRNN`	791	1,112	1,387	281	73.79	36.32	48.68	53.29
`Our approach` $_{\texttt{SRC}}$	1,054	2,439	60	18	98.32	94.61	96.43	97.82
`Our approach` $_{\texttt{SNRC}}$	1,057	2,452	47	15	**98.60**	**95.74**	**97.15**	**98.26**

As shown in Table 2, augmenting training data with security-unrelated commits (SNRC) improves all metrics over the non-augmented version (SRC). The F1-score increases from 96.43% to 97.15%, and precision from 94.61% to 95.74%, indicating better discrimination between true security fixes and unrelated changes. Compared to PatchRNN, our SNRC model shows substantial gains across all metrics. Notably, it reduces the false positive rate from 55.52% to 1.88%, confirming its ability to distinguish VFCs from non-security commits more effectively.

5 Conclusion

We presented an approach to improve the identification of vulnerability-fixing commits (VFCs) by augmenting the Devign dataset with security-unrelated commits and fine-tuning CodeT5 on this expanded data. Our model achieved an F1-score of 97.15%, significantly outperforming PatchRNN (48.68%), and reduced the false positive rate from 55.52% to 1.88%. This improvement highlights the importance of exposing the model to diverse, non-security changes when learning to recognize real security fixes. The resulting system enables more accurate and efficient detection of silent patches in OSS, reducing unnecessary developer effort and enhancing the reliability of automated vulnerability tracking.

Acknowledgments. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00440780, Development of Automated SBOM and VEX Verification Technologies for Securing Software Supply Chains), (No.RS-2022-II220277, Development of SBOM Technologies for Securing Software Supply Chains). In addition, this work was supported by ICT Creative Consilience Program through the Institute of Information Communications Technology Planning Evaluation(IITP) grant funded by the Korea government (MSIT) (IITP-2025-RS-2020-II201819).

References

1. Devign dataset (2025), https://sites.google.com/view/devign, Accessed July 2025.
2. Wang, S., Wang, X., Sun, K., Jajodia, S., Wang, H., Li, Q.: Graphspd: Graph-based security patch detection with enriched code semantics. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 2409–2426. IEEE (2023)
3. Wang, X., Wang, S., Feng, P., Sun, K., Jajodia, S., Benchaaboun, S., Geck, F.: Patchrnn: A deep learning-based system for security patch identification. In: MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM). pp. 595–600. IEEE (2021)
4. Wang, Y., Wang, W., Joty, S., Hoi, S.C.: Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. arXiv preprint arXiv:2109.00859 (2021)
5. Zhou, Y., Liu, S., Siow, J., Du, X., Liu, Y.: Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. Advances in neural information processing systems **32** (2019)
6. Zhou, Y., Sharma, A.: Automated identification of security issues from commit messages and bug reports. In: Proceedings of the 2017 11th joint meeting on foundations of software engineering. pp. 914–919 (2017)
7. Zhou, Y., Siow, J.K., Wang, C., Liu, S., Liu, Y.: Spi: Automated identification of security patches via commits. ACM Transactions on Software Engineering and Methodology (TOSEM) **31**(1), 1–27 (2021)
8. Zuo, F., Rhee, J.: Vulnerability discovery based on source code patch commit mining: a systematic literature review. International Journal of Information Security **23**(2), 1513–1526 (2024)
9. Zuo, F., Zhang, X., Song, Y., Rhee, J., Fu, J.: Commit message can help: security patch detection in open source software via transformer. In: 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA). pp. 345–351. IEEE (2023)