Patchmatch: A Tool for Locating Patches of Open Source Project Vulnerabilities

1st Kedi Shen
Hangzhou City University
China
skuld_1456@163.com

2nd Yun Zhang[‡]
Hangzhou City University
China
yunzhang@zucc.edu.cn

3rd Lingfeng Bao
Zhejiang University
China
lingfengbao@zju.edu.cn

4th Zhiyuan Wan
Zhejiang University
China
wanzhiyuan@zju.edu.cn

5th Zhuorong Li
Hangzhou City University
China
lizr@zucc.edu.cn

6th Minghui Wu
Hangzhou City University
China
mhwu@zucc.edu.cn

***Abstract*—With the rapid development of open source projects, the continuous emergence of vulnerabilities in the project brings great challenges to the security of the project. Security patches are one of the best ways to deal with vulnerabilities, but are not well applied currently. Although there are sites like CVE/NVD that provide information about vulnerabilities, many of the vulnerabilities disclosed by CVE/NVD are not accompanied by security patches. This makes it difficult for developers to apply patches. In the present study, a sorting method based on extracting multidimensional features from auxiliary information in CVE/NVD was proposed. And we made a further step, we proposed** VCMATCH**, a model for mining semantic information in vulnerability description and code commit messages, which has good recall rate and applicability across projects. On this basis, we established Patchmatch, a tool for helping developers to quickly locate patches. Given a vulnerability, Patchmatch can forecast the implicit patches in the code repository's commits. Patchmatch also has a visual webpage for information statistics and a display web page to help developers manage all kinds of information in the code repository. A demo video of Patchmatch is at https://www.youtube.com/watch?v=nOBSMFtZV8A. Patchmatch is in https://github.com/Sklud1456/patchmatch.**

***Index Terms*—Vulnerability, Model application, Manage tool**

I. Introduction

Open source software (OSS) is widely adopted by the software industry. However, vulnerabilities in OSS pose significant risks to practical applications. In addition, the number of OSS vulnerabilities is increasing. According to the Georgia analysis [1], more than twice as much vulnerability data was collected by the National Institute of Standards and Technology (NIST) [2] between 2009 and 2019 as it did between 1999 and 2009. Common Vulnerabilities and Exposures (CVE) [3] has collected more than 170,000 vulnerability data up until now.

Common Vulnerabilities & Exposures (CVE) is a specialized vulnerability information list sponsored by the U.S. Department of Homeland Security. Each CVE entry contains information related to the vulnerability, such as the CVE-ID, vulnerability description, vulnerability reference, and creation date. The CVE-ID is the unique identifier of vulnerability data assigned by The CVE Numbering Authorities (CNAs). The vulnerability description mentions vulnerable software repositories and the consequences of remote attackers. Sometimes, the description also mentions the function name, file name, and software version of the vulnerability, which is important in identifying the patch of this vulnerability.

The National Vulnerability Database (NVD), sponsored by the National Institute of Standards and Technology (NIST), provides enhanced information based on CVE lists that can complement CVE information such as severity scores, fix information, and CWE. CWE (Common Weakness Enumeration) is a category system that uses CWE-ID and CWE name to identify vulnerability types.

Even with public release information such as CVE, locating security patches (often in the form of code commits in code repositories) for vulnerabilities still is a challenge. One reason is that numerous CVE/NVD entries lack information about security patches. Additionally, manually obtaining and identifying patches for vulnerabilities can be difficult. Hogan et al. [4]reported that manual tagging is a highly skilled, time-consuming task that is still prone to error because of the lack of knowledge. Therefore, there must be an automatic method to identify the vulnerability patches.

The vulnerability description field in the CVE and secondary information in the code commit message (such as CVE-ID or bugID) can be used to automatically match vulnerabilities and patches in the code repository. However, most patches cannot be automatically matched using this method directly as the auxiliary information is often incomplete. Therefore, some researchers intend to use the method of machine learning. Tan et al. [5] proposed a model of matching vulnerabilities and security patches named PatchScout, which transformed the search problem of locating security patches into the problem of sorting code commit, and achieved good results. However, PatchScout only considers digital features

This research was supported by the Natural Science Foundation of Zhejiang Province under Grant No. LQ21F020008, the National Science Foundation of China (No. 62102358), and the 2019 Innovation at Hangzhou Program for High-level Returned Oversea Scholars.

[‡]Corresponding author

in the vulnerability description and commit information fields, and lacks semantic information contained in the text. Moreover, the researchers did not provide a visual tool to help others understand the use of the model more intuitively.

Based on our previous work [6], we proposed a web-based tool called Patchmatch. The function of Patchmatch is divided into two parts: (i) Information visualization: developers can import the local Git repository into MySql database and add relevant CVE information, and then Patchmatch will display some statistics on the home page, such as count the number of CVEs by type, time and risk score and display them in the form of chart. Of course, Patchmatch can also display the code commit information by chart. (ii) Forecast patch: When the developer enters a new CVE information that has not been matched to a patch, they can use the page for patch prediction. Patchmatch can use the trained VCMATCH model to predict the patch for the CVE and display the patch in order according to the predicted score, which is used for manual validation by developers. Figure 1 presents the overall framework of Patchmatch, which is composed of VCMATCH and its supporting front and back end display system. The rest of the paper is organized as follows: Section 2 introduces the construction details of our VCMATCH model. Section 3 introduces the implementation details and key application scenarios of Patchmatch. Section 4 evaluated our VCMATCH and Patchmatch. Section 5 summarizes our work.

II. MODEL BUILDING

We built the model in five stages: data collection, data pre-processing, manual feature extraction, deep text feature extraction, ranking voting fusion model.

Data collection. We selected 10 popular OSS projects, including FFmpeg, ImageMagick, Jenkins, OpenSSL, QEMU, Wireshark, Linux, Moodle, PHP-src, and phpMyAdmin. These projects have been extensively studied in previous vulnerability analysis studies. We cloned these ten code repositories from GitHub or GitLab to get these project code commits. And use gitlog to get its log files, which is used to extract the details from each commit. Each commit is uniquely identified by a 40-bit long commit ID. For information of vulnerabilities, we collect their CVE-ID, text description, and creation date from the CVE website, and collect their supplementary information from NVD. At the same time, snyk [7] database was combined to ensure that each collected CVE would have the corresponding commit, which is the security patch for the vulnerability.

Data pre-processing. After confirming the security patch of each vulnerability, we preprocessed the text information of CVE and commit (like vulnerability description, commit information, CWE name, etc.) comprehensively. We use the natural language processing tool — BERT to tokenize the text message, and get the token set of vulnerability and the token set of code commit. And in order to reduce the later calculation consumption and reduce the data scale, we processed the intersection of the two, and the result was considered as "Useful Token", while the other tokens were abandoned.

Manual feature extraction. Based on the features used by PatchScout, we extracted additional features and divided them into four feature dimensions to measure the correlation between vulnerability and commit: LOC dimension, identity dimension, location dimension and token dimension. The LOC dimension represents the information related to the number of code lines submitted for modification. The identity dimension represents various identity information contained in the commits, such as CVE-ID, bugID, problem ID, etc. At the same time, we also extract URL from the commit message to obtain more information. The location dimension represents the correlation degree between the modified file name, file path name, function name and the vulnerability description in the code commit, these are dimensions in space location. And the time interval between the two is also recorded from the dimension in time location. Token dimension is to extract mathematical statistics from the token set generated in data preprocessing. Meanwhile, TF-IDF vectors for vulnerability description and commit messages are respectively generated to explicitly indicate the similarity between vulnerability and code commit in terms of text information.

Deep text feature extraction. We use BERT, a state-of-the-art pre-training model, to generate the coded characteristics of vulnerability descriptions and commit messages respectively. BERT can map the content of text to a deep semantic vector space to mine the deep semantic information in natural language. At the same time, we also reduced the dimension of the generated vector to 32 dimensions, so that it can be adapted to the model for subsequent use.

Ranking voting fusion model. We extracted 36 handcrafted features and 64 deep text features (32 for each vulnerability and commit) for each pair of vulnerabilities, totaling 100 features. We put these 100 features into three models (XGBoost, LightGBM and CNN) for training and model fusion to achieve the final prediction.

Identifying security patches is a matter of categorizing highly unbalanced data. Given a vulnerability, only a few commits in code repository are security patches (the positive cases), while the rest are negative cases. Traditional machine learning models can easily lead to inadequate fitting and incorrect separation of positive and negative samples. Therefore, we choose the XGBoost, the LightGBM and a CNN model equipped with focus loss function, because they show good classification effect on unbalanced data.

In terms of model fusion, we adopt a new ranking fusion method based on voting. Given a vulnerability, we get the ranking of each candidate commit according to the matching probability scores of the three classifiers, which are respectively rank$_{XGB}$, rank$_{LGB}$ and rank$_{CNN}$. Then we take the two closest values from the three submitted rank values, denoting the first and second rank values as the two most trusted levels. Finally, we calculate the average of rank$_1$ and rank$_2$, denoted as rank$_{avg}$. Our submissions based on rank$_{avg}$ and get the new level of the submissions.

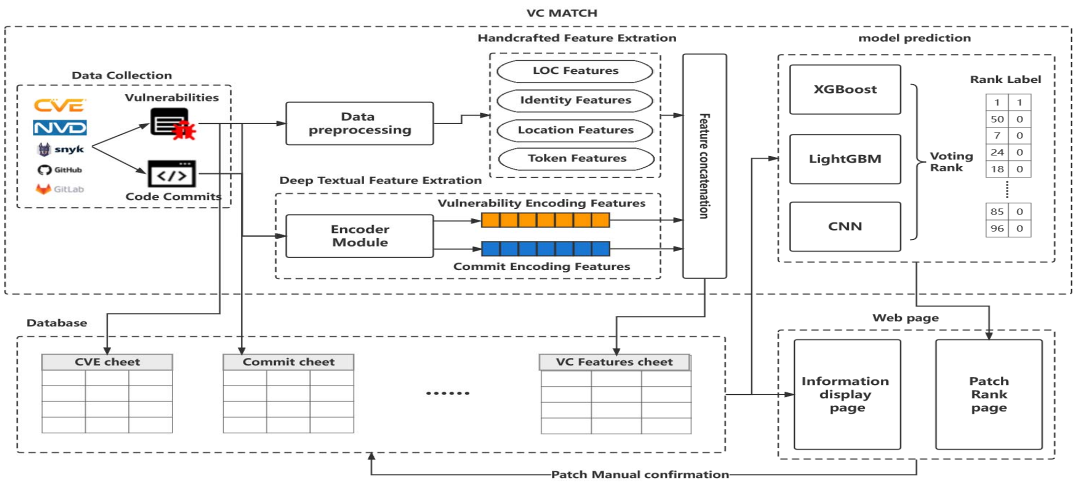


Fig. 1. Overview of the method framework

III. System implementation details

In this section, we will introduce how we build the system from various angles.

A. Data Collection

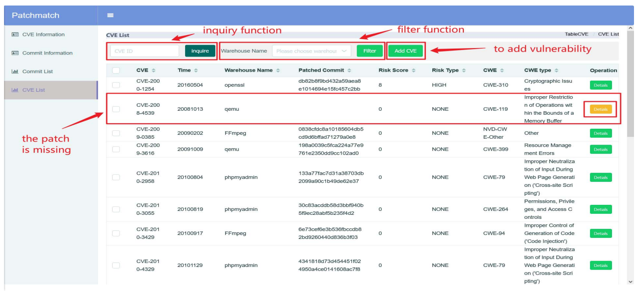
When we collected data for train in the second section, we also collected some other data for information statistics and presentation. The information includes commit author, commit time, description information, CVE risk score and rating, CVE release time, CWE-ID and its name, etc. We sorted out this information and stored them in the MySql database for unified management, which is convenient for later use.

B. Backend Model

In order to shorten the user interaction wait time, we saved the model weight of VCMatch locally, so that the model can be directly used to predict the ranking of patches. For the backend, we chose the flask framework, which use the same programming language we used to train the model.

C. Frontend User Interface

The frontend is mainly divided into two modules. One is the information display module, which is used to display the commit and vulnerability information stored in the backend. At the same time, some statistics (such as the number of CVE in each repository) are also carried out to display. The other one is the model prediction module. When the patch of a vulnerability is missing, developers can enter this page for model prediction, and the system will automatically extract the vulnerability related feature data from the database into the VCMatch model, then return the model prediction results to the front interface for developers to confirm the patch.



CVE List

CVE	Time	Warehouse Name	Patched Commit	Risk Score	Risk Type	CWE	CWE type	Operation
CVE-2000-1254	20160504	openssl	db82b8f9bd432a59aea8e1014694e15fc457c2bb	8	HIGH	CWE-310	Cryptographic Issues	Detials
CVE-2008-4539	20081013	qemu		0	NONE	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	Detials
CVE-2009-0385	20090202	FFmpeg	0838cfdc8a10185604db5cd9d6bffad71279a0e8	0	NONE	NVD-CWE-Other	Other	Detials
CVE-2009-3616	20091009	qemu	198a0039c5fca224a77e9761e2350dd9cc102ad0	0	NONE	CWE-399	Resource Management Errors	Detials
CVE-2010-2958	20100804	phpmyadmin	133a77fac7d31a38703db2099a90c1b49de62e37	0	NONE	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Detials
CVE-2010-3055	20100819	phpmyadmin	30c83acddb58d3bbf940b5f9ec28abf5b235f4d2	0	NONE	CWE-264	Permissions, Privileges, and Access Controls	Detials
CVE-2010-3429	20100917	FFmpeg	6e73cef6e3b536fbccdb82bd9260440d836b3f03	0	NONE	CWE-94	Improper Control of Generation of Code ('Code Injection')	Detials
CVE-2010-4329	20101129	phpmyadmin	4341818d73d454451f024950a4ce0141608ac7f8	0	NONE	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Detials

Fig. 2. The information Page

1) Exhibiting Information: Figure 2 is the information display page of Patchmatch, which is also divided into two web pages of vulnerability and commit. In this part, the functions of the two pages are similar. Developers can visually find, add, and view vulnerabilities/commits information here. Click the "Details" button on the right of the page to see more information about the vulnerabilities/commits. Of course, Patchmatch also provides the function of filtering vulnerabilities/commits by the name of the repository to meet the various needs of developers for information collection.

2) Model Application: If the corresponding patch is missing, Patchmatch will turn the "details" button yellow to remind the developer that the patch is missing. Figure 2 simulates the user scenario. It can be seen that the patch is missing for the vulnerability in the second row. The developer can interact to enter the model prediction page and use the model to check the possibility of each commit becoming the corresponding vulnerability patch. Figure 3 is the model prediction page in Patchmatch. Developers can click the "prediction" button to use the model and data in the backend for patch sorting prediction. When the prediction is complete, the results are transmitted backend to the frontend. Developers can also toggle the number of top entries in the TopN column. Developers

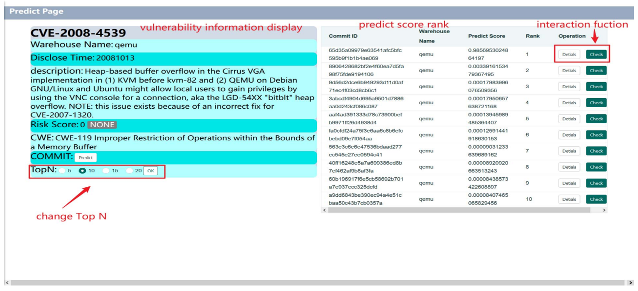
TABLE I
RECALLS OF EACH MODEL

Top K	1	3	5	10
Logistic Regression	10.43%	14.56%	17.44%	23.13%
Linear Regression	65.97%	75.67%	78.38%	82.38%
PatchScout	71.12%	81.43%	86.10%	89.45%
XGBoost	88.67%	93.83%	95.21%	96.70%
LightBGM	86.01%	92.27%	94.19%	95.74%
CNN	85.80%	90.98%	92.51%	94.31%
VCMATCH	**88.86%**	**94.01%**	**95.33%**	**97.06%**

TABLE II
MANUAL EFFORTS OF EACH MODEL

Top K	1	3	5	10
Logistic Regression	1.0000	2.7687	4.4627	8.4685
Linear Regression	1.0000	1.6087	2.0743	3.0695
PatchScout	1.0000	1.5093	1.8610	3.0695
XGBoost	1.0000	1.1947	1.3116	1.5164
LightBGM	1.0000	1.2312	1.3774	1.6327
CNN	1.0000	1.2468	1.4182	1.7435
VCMATCH	**1.0000**	**1.1911**	**1.3038**	**1.4997**

can click on "Details" button to check the details screen of each commit to verify that the commit is correct. Then confirm the patch for the vulnerability by click "confirm" button.



CVE-2008-4539
Warehouse Name: qemu
Disclose Time: 20081013
description: Heap-based buffer overflow in the Cirrus VGA implementation in (1) KVM before kvm-82 and (2) QEMU on Debian GNU/Linux and Ubuntu might allow local users to gain privileges by using the VNC console for a connection, aka the LGD-54XX "bitblt" heap overflow. NOTE: this issue exists because of an incorrect fix for CVE-2007-1320.
Risk Score: 0 NONE
CWE: CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer
COMMIT: Predict
TopN: 5 [✓ 10] 15 20 OK

Commit ID	Warehouse Name	Predict Score	Rank	Operation
65d35a09979e63541afc5bfc595b9f1b1b4ae069	qemu	0.9856953024864197	1	Detials Check
8906428682bf2e4f60ea7d5fa98f75fde9194106	qemu	0.0033916153479367495	2	Detials Check
9d56d2dce6b949293d11d0af71ec4f03cd8cb6c1	qemu	0.00017983996076509356	3	Detials Check
3abcdf4904d695a9501d7886aa0d243cf086c087	qemu	0.00017950657638721168	4	Detials Check
aaf4ad391333d78c73900befb9971ff26d4938d4	qemu	0.00013945989485364407	5	Detials Check
fa0cfdf24a75f3e6aa6c8b6efcbebd09e7f054aa	qemu	0.00012591441918630153	6	Detials Check
563e3c6e6e47536bdaad277ec545e27ee0594c41	qemu	0.00009031233639689162	7	Detials Check
40ff16248e5a7a699386ed8b7ef462af9b8af3fa	qemu	0.00008920920663513243	8	Detials Check
60b196917f6e5cb58692b701a7e937ecc325dcfd	qemu	0.00008438573422608897	9	Detials Check
a9dd6843be390ec94a4e51cbaa50c43b7cb0357a	qemu	0.00008407465065829456	10	Detials Check

Fig. 3. The model prediction Page

IV. EVALUATION AND USER STUDY

In order to evaluate whether Patchmatch can meet users' expectations, we analyzed Patchmatch from two perspectives: on the one hand, we evaluated the recall and manual effort from the perspective of the model; on the other hand, we evaluated its practicality from the perspective of system tools.

About model. We hope that the ranking of patches is as high as possible. The higher the ranking of patches is, the better the performance of the model is. We quantify and evaluate the performance of the model through two mathematical dimensions: recall and manual effort. Manual effort is the number of commits that need to be checked to get the correct patch results. If the correct patch is in the front K, the manual effort value is the rank of the patch, otherwise it is K, this means that all K results are checked, but the correct patch is not available. When K is equal to 1, the manual effort value must be equal to 1, because we must need to check one patch.

TABLE III
USER FEEDBACK (TEN")

Question	Users' average score
The score for the presentation of statistics	7.5
The score for management information	7.6
The score for the application of the model	7.8
The score for the UI	7.0
The score for the overall use	8.2

Table 1 and Table 2 show the recalls and manual efforts of VCMATCH and other baseline models. It can be seen that the VCMATCH has better performance than the traditional models and has an excellent ability in predicting patches. With the excellent performance of VCMATCH model as the base, our Patchmatch system has guaranteed basic functions.

About user usage. We recruited 14 participants (3 Female and 11 Male) who were related to computer technology at university. Three participants were undergraduate students, five were master's students, four were Ph.D. students, and two were software developers (one in server security). All of them have two to five years of development experience.

We handed them Patchmatch to use. Before using it, functions of various parts of Patchmatch will be introduced in detail and some specific usage scenarios will be informed to facilitate them to score. After using Patchmatch, we will give them the prepared questionnaire for rating and ask them for their suggestions on Patchmatch. After survey, we counted and calculated their scores, and finally got an overall rating of 8.2.

Due to space constraints, we only show the summary scores. Detailed information can be seen on GitHub (https://github.com/Sklud1456/patchmatch), file "user research" shows questions in each area and their specific scores. From the above scores, we can see that Patchmatch has a good application in the aspect of statistical information and model application; but the information display interface has some deficiencies. After further inquiry, they all said that Patchmatch could meet their needs to a certain extent but needed some optimization. For example, polish the UI, broaden the universality of information display and optimize the information management.

In summary, Patchmatch is a successful application of VCMATCH, and provides certain practicability and convenience for developers. Although it has defects, developers are willing to try to use the Patchmatch.

V. SUMMARY AND FUTURE WORK

We demonstrated Patchmatch, a Web-based tool that applies the VCMATCH model to help developers locate patches in OSS. Developers can use the tool to visually view commits or vulnerabilities information and some statistical graphs in the code base. Developers can also use the built-in VCMATCH model to predict and rank security patches for vulnerabilities that have not yet been patched. In future work, we will further improve the model by collecting more vulnerability information and optimize the system (such as more information presentation and better UI) to meet the needs of more developers.

References

[1] Georgios Aivatoglou, Mike Anastasiadis, Georgios Spanos, Antonis Voulgaridis, Konstantinos Votis, Dimitrios Tzovaras, and Lefteris Angelis. 2022. A RAkEL-based methodology to estimate software vulnerability characteristics & score-an application to EU project ECHO. Multimedia Tools and Applications 81, 7 (2022), 9459–9479.

[2] 2022. "National Institute of Standards and Technology — NIST". [Online]. Available: https://www.nist.gov/.

[3] 2022. "Common Vulnerabilities Exposures — CVE". [Online]. Available: https://cve.mitre.org/.

[4] Kevin Hogan, Noel Warford, Robert Morrison, David Miller, Sean Malone, and James Purtilo. 2019. The challenges of labeling vulnerability-contributing commits. In 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 270–275.

[5] Xin Tan, Yuan Zhang, Chenyuan Mi, Jiajun Cao, Kun Sun, Yifan Lin, and Min Yang. 2021. Locating the Security Patches for Disclosed OSS Vulnerabilities with Vulnerability-Commit Correlation Ranking. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 3282–3299.

[6] Shichao Wang, Yun Zhang, Lingfeng Bao, Xin Xia, and Minghui Wu. 2022. VCMATCH: A Ranking-based Approach for Automatic Security Patches Localization for OSS Vulnerabilities. In 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2022).

[7] 2022. "Snyk — open Source Vulnerability Database". [Online]. Available: https://security.snyk.io/.