IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 9, September 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.524

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 9, September 2024

|DOI: 10.15680/IJIRSET.2024.1309097|

Automating the Generation of NVD CVE Fixes using Generative AI and RAG

Mahidhar Uppala

Engineering Manager/Enterprise Architect, Hitachi Digital Services, USA

Master of Computer Science, Arizona State University, USA

ABSTRACT: As modern software applications increase, the challenge of addressing security vulnerabilities [1] will increase. Generating fixes for these vulnerabilities rely on manual intervention—a process that can be both time-consuming and difficult to scale. In this paper, we propose an automated approach to CVE fix generation, leveraging data from the National Vulnerability Database (NVD) [2], a local vector database for efficient storage, and Generative AI to produce relevant fixes based on vulnerability descriptions. Our approach involves first loading all publicly available CVE data into a vector database, indexed by CVE ID and metadata for efficient retrieval. When a query for a CVE fix is made, then using Generative AI using custom prompts to provide fixes based on the vulnerability descriptions stored in the database. The generated solutions are stored back in the database for future use, or for review. In the methodology section, we demonstrate how this system can automate the generation of effective CVE fixes in a matter of seconds, streamlining a traditionally manual and time-consuming process. With the findings suggest that the proposed system can reduce the manual/time consuming effort on the teams while enhancing the speed and accuracy of vulnerability remediation

KEYWORDS: Generative AI, Gen AI, LLMs, RAG, Vector Database, Artificial Intelligence, National Vulnerability Database (NVD), CVE, Vulnerabilities, Software Security, Software Engineering, Software Development.

I. INTRODUCTION

In the modern digital era, software vulnerabilities have become a persistent threat to both businesses and individuals. The CVEs, maintained by the National Vulnerability Database (NVD) [2] systems tracks the publicly disclosed security issues. However, as applications become complex and old, the number of CVEs continues to grow significantly. That makes challenge for cybersecurity/application teams to manually address each vulnerability in a timely manner. This adds delay in the fix and allows the attackers to exploit known vulnerabilities.

Even having vulnerability detection and reporting methods [1], the process of generating and implementing fixes still heavily depends on the manual intervention, Sample dependency check report is shown in Fig. 2. Security/Applications team analyze the CVE [2] descriptions, understand the nature of the vulnerability/issue, and apply the fix.

In this paper, we explore the process of using Generative AI to automate the CVE fix generation. Our proposed system integrates NVD data retrieval, local vector databases for efficient storage, and Generative AI to generate context-based patches or configuration fixes. This solution target is to reduce the manual effort required for vulnerability remediation and provide an automated and scalable solution.

By loading all available CVE data from the NVD into a vector database, our system allows for quick retrieval based on CVE ID and associated metadata. Also, solves the potential downtimes from NVD services if it occurs. Once the relevant CVE is identified, Gen AI models can interpret the vulnerability description and suggest potential fixes. These fixes can then be stored in the database for future reference or for review. This paper outlines our approach, discusses the results of a practical case study, and explores how the system can transform the way we handle vulnerability remediation.

II. LITERATURE SURVEY

The Common Vulnerabilities and Exposures (CVE) [1] in the National Vulnerability Database (NVD) plays a critical role in software vulnerabilities. The NVD services provide data on vulnerabilities, but generating fixes remains manual

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 9, September 2024

|DOI: 10.15680/IJIRSET.2024.1309097|

and becomes complex as the number of vulnerabilities increases. We can automate the CVE fixes by leveraging Generative AI LLMs, like OpenAI's GPT.

The Vector database is essential in Generative AI RAG models for its ability to quickly retrieve and store large datasets, making it ideal for AI-driven systems. These databases help further access CVE data and metadata, supporting the automated generation of fixes. Even with AI for vulnerability detection, most frameworks still require human intervention to create patches, highlighting current research gaps. This paper addresses that gap by proposing an automated system that combines Generative AI and vector databases to automate CVE fix generation, significantly improving the speed and efficiency of vulnerability remediation.

III. METHODOLOGY

In this section, we discuss about the key components and steps involved in automating CVE fix generation system/application. The goal of this system is to provide effective vulnerability management process by automating the CVE fix generation by integrating with, NVD services, vector databases, and Generative AI. Refer to Fig. 1 for the architecture.

1. NVD Data Retrieval and Loading

- In this section, the system loads all the publicly available CVE data as the initial load from the National Vulnerability Database (NVD) services using Api key, which includes records from the year 2000 to the latest year. Each CVE entry typically includes a unique identifier (CVE ID), a detailed description of the vulnerability, its CVSS (Common Vulnerability Scoring System) score, and external references that might point to existing patches or mitigation strategies.
- Once the data is retrieved, it is parsed and organized into structured formats such as JSON. The parsed data is then loaded into a local vector database for efficient storage and retrieval. This database uses tokens to organize the data, allowing for quick searches based on CVE ID, metadata tags (such as severity and affected software), and the original vulnerability description.
- Also, the system will have capability of loading delta updates using timestamp from NVD services to make sure the CVE data is up to date in the vector database.

2. Querying the Database for Fix Generation

- When a security/application team or automated process queries the system for a specific CVE, the relevant information is retrieved from the vector database. The user can search by CVE ID or by keywords associated with the vulnerability's description. Once the query returns the relevant data, the system extracts the original description and associated metadata, which serves as input for the next phase: fix generation.

3. Fix Generation Using Generative AI

- To automate the fix generation process, we employ a Generative AI model specifically trained to generate context-based solutions. The model takes the CVE description and metadata as input and generates potential remediation steps. These steps may include code patches for software flaws, configuration adjustments, or general mitigation strategies.
- The AI is fine-tuned to recognize patterns from previously resolved CVEs and generate solutions that are relevant to the current vulnerability. For instance, in the case of a buffer overflow vulnerability, the AI might suggest code to improve input validation or limit buffer sizes.

4. Storing and Refining Generated Fixes

- Once the AI generates a proposed fix, the solution is stored in the vector database alongside the original CVE entry. This allows for the fix to be quickly retrieved in future queries and compared to solutions for similar vulnerabilities. The system also keeps track of the confidence score of the AI-generated fix, indicating how reliable the solution is based on prior learning.
- In addition to automated fixes, security/application teams can review and manually adjust the generated solutions. This feedback is then used to refine the AI model, continuously improving its ability to generate accurate fixes.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 9, September 2024

|DOI: 10.15680/IJIRSET.2024.1309097|

5. Feedback and Continuous Improvement

- The system's ability to improve over time is a key feature. Feedback from human experts is used to evaluate the quality and applicability of the AI-generated fixes. By incorporating this feedback into the model's training process, the system can refine its understanding of vulnerabilities and produce more accurate fixes in the future.

IV. EXPERIMENTAL RESULTS/FIGURES

Figures shows the architecture and sample results.

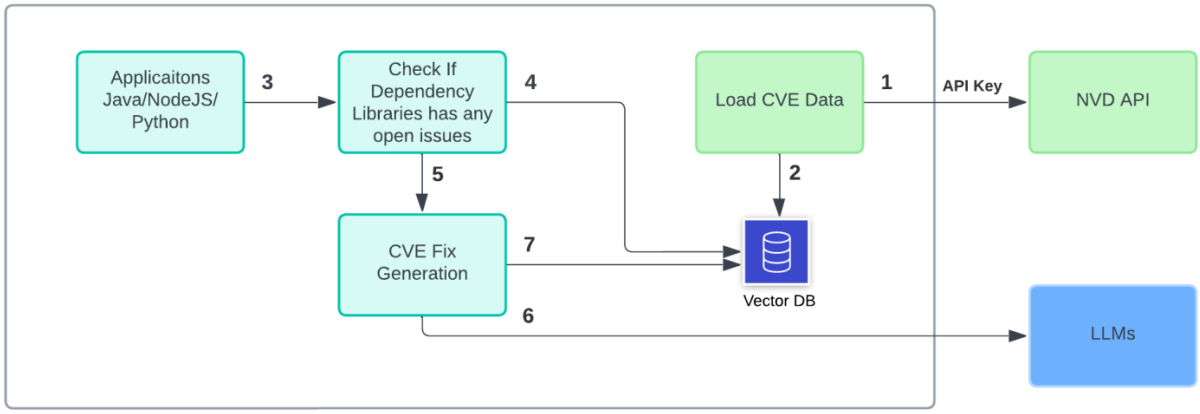


Fig. 1. Architecture of Automating the generation of NVD CVE fixes using Generative AI and RAG

As shown in Fig. 1, the following are the steps involved in Automating the generation of NVD CVE fixes using Generative AI.

1. Get all NVD CVE issues by calling the NVD API/Service. This process will run periodically for up-to-date CVE issue data from the NVD service.
2. Persist NVD CVE issues in a Vector Database with the CVE issue ID as a unique identifier and other additional data as metadata. Sample output is shown in Fig. 3.
3. Get the dependency vulnerabilities/CVE issues from software applications such as Java, NodeJS, Python, PHP, and many more.
4. Check if the given CVE issue is available in Vector Database.
5. If the CVE issue is found in Vector DB, then CVE fix Generation (Gen AI Model) will pass the CVE ID and CVE metadata to LLMs (text-based) to generate the content for the CVE fix. The fix will be stored in Vector Database for the future use/review process. Sample output is shown in Fig. 4.

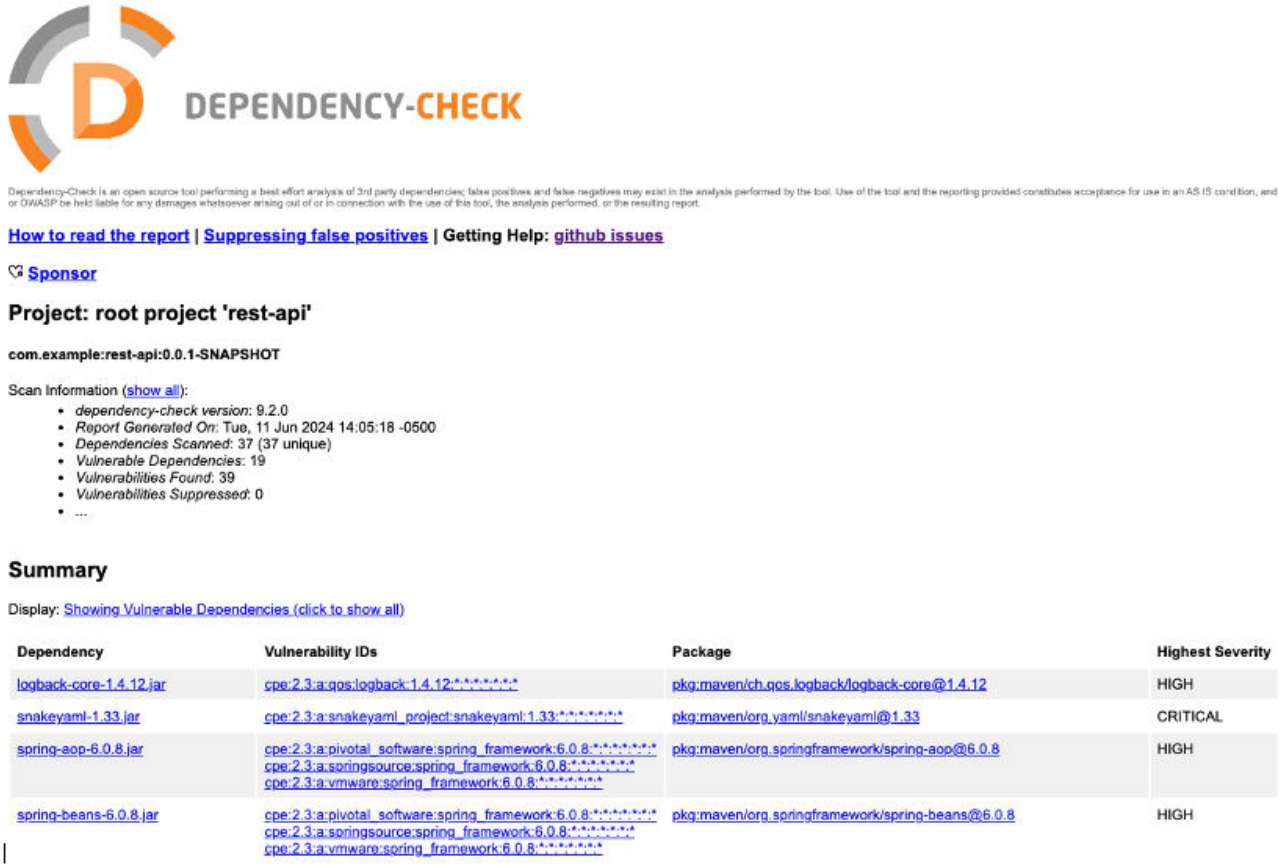
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 9, September 2024

|DOI: 10.15680/IJIRSET.2024.1309097|



DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: root project 'rest-api'

com.example:rest-api:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 9.2.0
- *Report Generated On*: Tue, 11 Jun 2024 14:05:18 -0500
- *Dependencies Scanned*: 37 (37 unique)
- *Vulnerable Dependencies*: 19
- *Vulnerabilities Found*: 39
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity
logback-core-1.4.12.jar	cpe:2.3:a:qos:logback:1.4.12:*:*:*:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.4.12	HIGH
snakeyaml-1.33.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.33:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.33	CRITICAL
spring-aop-6.0.8.jar	cpe:2.3:a:pivotal_software:spring_framework:6.0.8:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:6.0.8:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:6.0.8:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-aop@6.0.8	HIGH
spring-beans-6.0.8.jar	cpe:2.3:a:pivotal_software:spring_framework:6.0.8:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:6.0.8:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:6.0.8:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-beans@6.0.8	HIGH

Fig. 2. Dependency vulnerabilities sample report

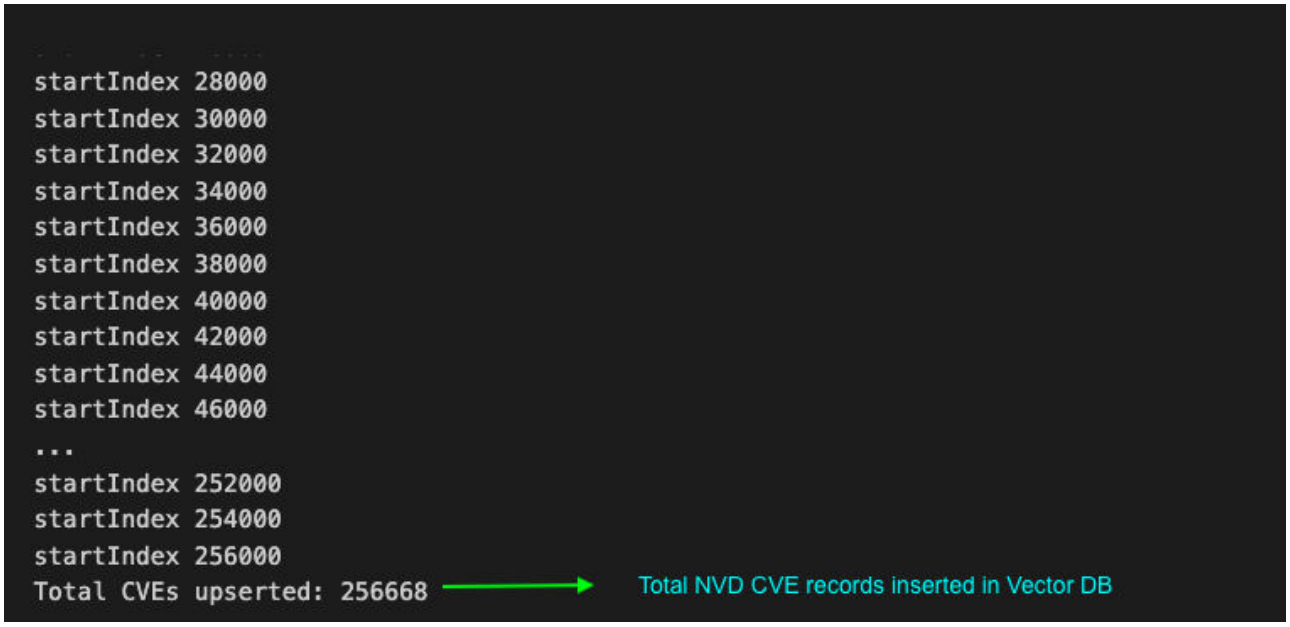


Fig. 3. Total NVD CVE records inserted in Vector Database

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 9, September 2024

|DOI: 10.15680/IJIRSET.2024.1309097|

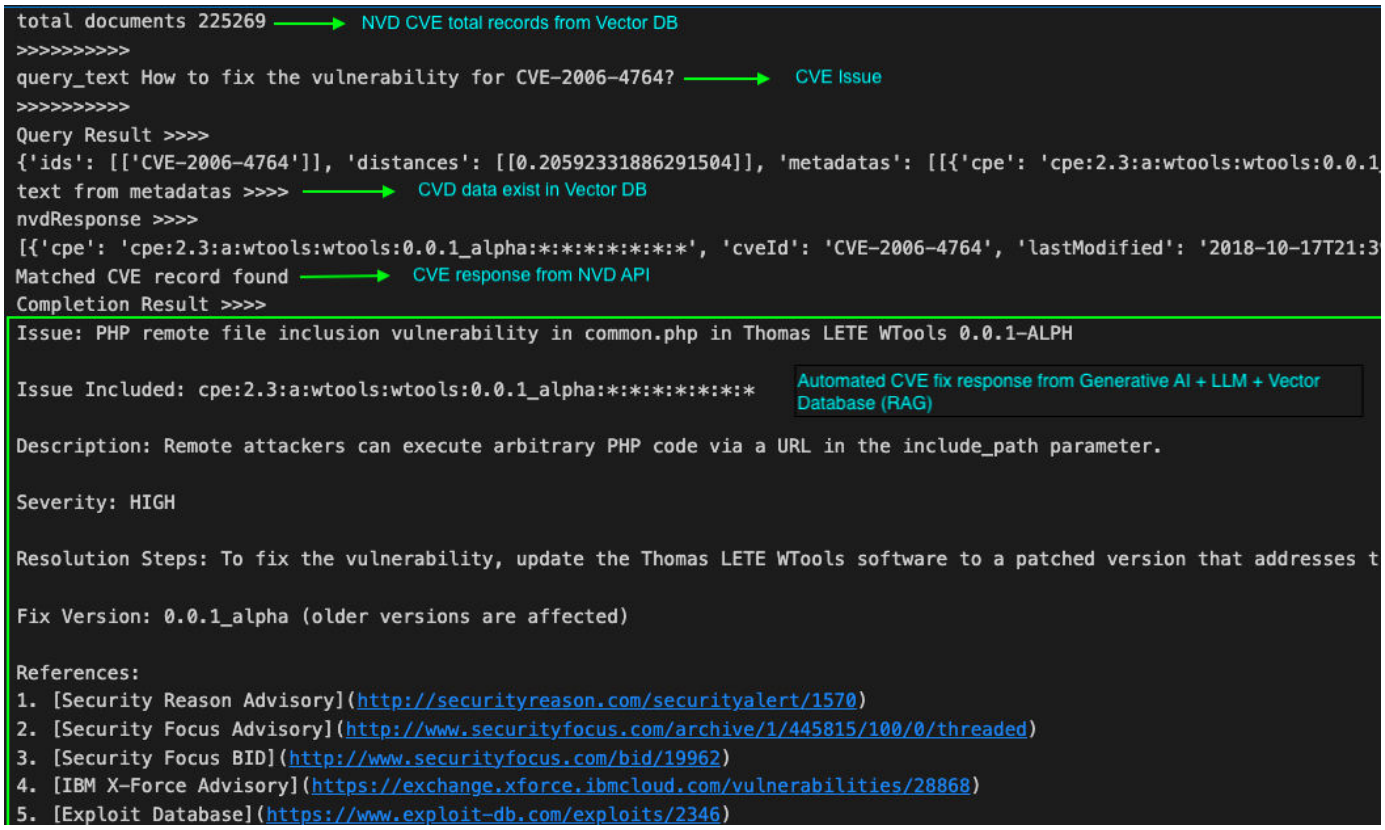


Fig. 4. CVE fix generation using Generative AI + LLM + Vector Database (RAG)

V. CONCLUSION

In this paper, we proposed an automated system for generating CVE fixes using Generative AI and vector databases, offering a scalable solution to address the growing number of software vulnerabilities. By leveraging data from the National Vulnerability Database (NVD) and using Generative AI models to generate CVE fixes, our system reduces the dependency on manual interventions, which are often time-consuming and difficult to scale. Through a case study/poc, we demonstrated how the system can retrieve CVE data, generate appropriate fixes in seconds, and store these solutions for future use. This approach significantly streamlines a traditional manual process, enabling security/application teams to respond faster to potential threats and improve the overall accuracy of their vulnerability remediation efforts.

Overall, this system represents a significant step toward automating the cybersecurity landscape, offering a practical solution for addressing the increasing demands of vulnerability management in modern software environments.

REFERENCES

1. Mahidhar Uppala, “Reporting” in "Finding Dependency Vulnerabilities in Software Applications with CI/CD Integration" TechRxiv, 2024. https://doi.org/10.36227/techrxiv.172114411.13851660/v1
2. NVD and CVE in “https://nvd.nist.gov/general/cve-process”

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details