VulnRemediate AI: An Integrated Framework for Semantic Vulnerability Detection and Generative Remediation*

Pranta Mazumder[1], Raj Mutsuddy[1], Adnan Mobarok[1], Nay Nang[1], Mahir Daiyan[1], Arbid Chakma[1], Tanjim Mahmud[1], Rakhimjon Rajapboyevich Rakhimov[2], Sabirov Sardor[3], Barno Matchanova[4], Kuanishbay Kenesbaevich Seitnazarov[5], Mohammad Shahadat Hossain[6,7], and Karl Andersson[7†]

[1] Rangamati Science and Technology University, Rangamati, Bangladesh
prantamazumder.gen@protonmail.com, {mutsuddy12, adnan34, nang12, daiyan99, chakma35}@gmail.com, tanjim_cse@yahoo.com
[2] Urgench State University, Urgench, Uzbekistan
rakhimov909@gmail.com
[3] Mamun University, Uzbekistan
sardor78@gmail.com
[4] Urgench State Pedagogical Institute, Urgench, Uzbekistan
m67atchanova@gmail.com
[5] Nukus State Pedagogical Institute, Nukus, Uzbekistan
seitnazarov34@gmail.com
[6] University of Chittagong, Chittagong, Bangladesh
hossain_ms@cu.ac.bd
[7] Luleå University of Technology, Skelleftea, Sweden
karl.andersson@ltu.se

Abstract

The escalating volume and velocity of disclosed software vulnerabilities present a formidable challenge to modern cybersecurity operations. Traditional vulnerability management workflows, heavily reliant on manual analysis and signature-based detection, struggle to scale, leading to prolonged exposure times and an overwhelming burden on security analysts. This paper introduces VulnRemediate AI, a novel framework designed to automate the vulnerability management life cycle, from initial network discovery to the generation of actionable remediation guidance. The system integrates a multi-stage AI pipeline that begins with network service discovery, followed by high-fidelity semantic mapping of services to Common Vulnerabilities and Exposures (CVEs) using a Sentence-BERT (SBERT) model. Subsequently, a Random Forest classifier assesses vulnerability severity based on the Common Vulnerability Scoring System (CVSS), and a fine-tuned T5 transformer model generates human-readable, context-specific remediation instructions. This approach moves beyond conventional risk prioritization to address the critical, often-neglected phase of vulnerability resolution. An evaluation of the framework demonstrates that its SBERT-based semantic matching achieves superior accuracy compared to traditional keyword-based approaches, and the generative T5 model produces remediation guidance rated highly for clarity and correctness in qualitative assessments. VulnRemediate AI represents a significant step toward a more automated, efficient, and effective vulnerability management paradigm, aiming to reduce mean time to remediation (MTTR) and empower security teams to operate at the speed of modern threats.

Keywords: Vulnerability Detection, generative Remediation, natural language processing.

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 47, December 16-18, 2025, Sapporo, Japan.

†Corresponding author

1 Introduction

The contemporary digital landscape is characterized by an ever-expanding attack surface and an unprecedented rate of software vulnerability disclosures, creating a state of "massive data overload" for security teams[21]. A significant majority of organizations—82%—report that unifying disparate signals from various scanning tools is a primary challenge[7]. In response, Artificial Intelligence (AI) has become mission-critical for 56% of security operations, enhancing threat detection[17] and automating risk-based prioritization[19].
Despite these advancements, a critical gap persists. State-of-the-art platforms excel at answering, "What should we fix first?" but leave the crucial question—"How do we fix it?"—as a manual endeavor. This manual research bottleneck is a primary contributor to delayed patching and extended risk exposure.

This paper presents VulnRemediate AI, an integrated, open-source system that addresses this gap by automating the full vulnerability management life cycle. Its modular architecture enhances transparency, countering the "black box" problem that has led 55% of organizations to disable AI functionality over accuracy concerns. Our contributions are:

- **An novel Architecture:** A cohesive workflow combining network discovery (Nmap), semantic vulnerability mapping (SBERT), risk classification (Random Forest), and generative remediation (T5).
- **High-Fidelity Semantic Matching:** Demonstrating the superior efficacy of SBERT for mapping discovered services to CVEs by capturing contextual meaning beyond simple keyword matching.
- **Generative AI for Actionable Remediation:** A novel application of a fine-tuned T5 transformer model to automatically generate human-readable, actionable remediation instructions from CVE descriptions

2 Related Work

Vulnerability management has evolved from foundational open-source tools to sophisticated AI-enhanced platforms. Traditional scanners like Nmap and OpenVAS form the bedrock of network discovery, operating on rule-based and signature-matching paradigms [21] (see table 1). While comprehensive, they lack the semantic understanding of modern AI models. Specialized tools like OWASP ZAP focus on web application logic flaws, a different niche than VulnRemediate AI's network-level service focus [1].
Commercial platforms like Tenable.io and Qualys VMDR leverage AI for "Predictive Prioritization," helping teams focus on the most exploitable threats [7]. These tools master prioritization but stop short of generating remediation guidance. VulnRemediate AI distinguishes itself by applying Natural Language Processing (NLP) not just for analysis, as seen in tools like Arcanna.ai [7] , but for the specific task of generating resolution steps. This approach is grounded in academic research that uses NLP to map relationships between security knowledge bases like CVE and MITRE AT&CK [2].

The system's generative component is situated within the emerging field of AI for automated program repair (APR) [13].The primary goal of Automated Program Repair (APR) is to automatically generate source code patches to fix software bugs. This process traditionally follows a workflow that begins with fault localization to identify the root cause of the bug

in the code [8]. More recently, the APR field has shifted significantly toward learning-based approaches that leverage deep learning. These modern techniques often treat bug fixing as a translation problem, similar to how machine translation converts one language to another [28]. However, instead of generating potentially risky source code patches, VulnRemediate AI takes a more pragmatic approach by generating natural language instructions for a human operator,augmenting the analyst's capabilities while significantly accelerating the remediation process.

Table 1: Comparison of Vulnerability Management Tools

System/Tool	Primary Function	Analysis Method	AI/ML Application	Primary Output	Open Source
VulnRemediate AI	Network V.M. & Remediation ,	Network Scan, NLP	Semantic Matching, Risk Classification Generative	Vulnerability Report+ Remediation Text	Yes
OpenVAS	Network V.M.	Network Scan, Signature Matching	N/A	Vulnerability Report	Yes
Tenable.io	Network V.M. & Prioritization	Network Scan, Threat Intel Analysis	Risk Prioritization Generative Remediation	Prioritized Vulnerability List	No
OWASP ZAP	Web Application Security	DAST	N/A	Web Vulnerability Report	Yes
CodeT5 (for APR)	Automated Code Repair	Static Code Analysis	Code Generation	Patched Source Code	Yes

3 System Design and Methodology

VulnRemediate AI is implemented as a client-server application with a web-based frontend and a FastAPI backend orchestrating a multi-stage AI pipeline. The workflow proceeds through three core phases:

Phase 1: Service Discovery and Semantic Vulnerability Mapping: The process begins when the system executes an Nmap scan (-sV) on a target IP to discover open ports and service versions. To overcome the limitations of keyword matching, the system uses Sentence-BERT (SBERT), a transformer model adapted for generating semantically meaningful sentence embeddings. A corpus of almost 105,000 CVE descriptions is pre-processed into a matrix of vector embeddings [23]. At runtime, each discovered service string is embedded using the same SBERT model, and a highly efficient cosine similarity search identifies the top-k most semantically similar CVEs from the pre-computed matrix.
Phase 2: Vulnerability Severity Classification: Once relevant CVEs are identified, a pre-trained Random Forest classifier which was fit offline, assesses their severity. The model was trained on CVE descriptions vectorized using TF-IDF, with target labels (Low, Medium, High, Critical) derived from their numerical CVSS scores. This provides a consistent, categorical severity rating for prioritization.
Phase 3: Generative Remediation with a Fine-Tuned T5 Model: The final phase uses a T5-small model that was fine-tuned for text-to-text generation.[24, 14] This fine-tuning process involved starting with the general-purpose pre-trained T5-small model and further training it on our specialized dataset of 'paired CVE descriptions and human-written remediation steps.' [5]This taught the model to take a CVE description as input and generate the corresponding remediation instructions as output.The T5-small model was fine-tuned to function as a text-to-text generator, taking a CVE description as input and producing remediation steps as output.

The T5-small model was fine-tuned to function as a text-to-text generator, taking a CVE description as input and producing remediation steps as output.

- **Training Dataset Composition:**
  - **Source:** The training data was derived from our curated dataset of almost 105,000 CVEs from the National Vulnerability Database (NVD) (2023-2025), GitHub Advisory Database , Snyk Vulnerability Database.
  - **Data Pairs:** We created a specialized dataset of "paired CVE descriptions and human-written remediation steps".
  - **Data Formatting:** The model was trained using a standard text-to-text format. For example:

    The input was prefixed "remediate: [CVE-DESCRIPTION])" and the model was trained to generate the corresponding "[REMEDIATION-TEXT]" as the target.
  - **Dataset Size:**
    After cleaning and pairing, the final fine-tuning dataset consisted of [e.g., 79,000] description-remediation pairs.

- **Fine-Tuning Process (Hyperparameters):** The T5-small model was fine-tuned using the following key hyperparameters:
  - **Framework:** Hugging Face Transformers
  - **Learning Rate:** 3e-5
  - **Batch Size:**16
  - **Number of Epochs:** 4
  - **Optimizer:** AdamW with a linear warmup
  - **Max Input/Output Length:** 512 tokens for input, 256 for output

When a user requests a fix, the CVE descriptionis fedto the T5 model, which generates a new, human-readable paragraph detailing the recommended correctiveactions. This modular architecture of smaller, expert models (SBERT for search, RandomForest for classification, T5 for generation) is computationally efficient and allows for independent updates, prioritizing practical performance and deployability.

3.1 System Architecture

The process begins when a User submits a scan request for a target IP through the Web Frontend, which is sent to the FastAPI Backend(see figure 1). The backend first uses the Nmap Scanner to discover services and versions on that IP. For each service found, the backend gets a vector embedding from the SBERT Model and uses Cosine Similarity Search to find the most similar CVEs from a pre-computed CVE Embedding Matrix. Once the CVEs are identified, the backend loops through them and uses the RandomForest Classifier to determine a categorical severity rating (e.g., High, Critical) for each. Finally, when the user requests a solution, the backend sends the relevant CVE details to the T5 Generator to create human-readable remediation steps. This complete report, including the CVE list, their severities, and the remediation guidance, is then sent back to the Web Frontend for the user to view.

Figure 1: System Architecture

3.2 Process Flow

The process starts when the **User submits a scan request** via the Web Frontend, which is processed by the FastAPI Backend.

Vulnerability Analysis (Discovery & Classification)

The core analysis proceeds in the following sequential steps:

- The **FastAPI Backend** initiates a scan using the **Nmap Scanner** to identify active services and their version numbers on the target IP.
- For each discovered service, the **SBERT Model** calculates a unique **vector embedding** (a numerical representation of the service description).
- This vector is compared using **Cosine Similarity Search** against a pre-computed **CVE Embedding Matrix** to quickly find the most relevant, known vulnerabilities (CVEs).
- The identified CVEs are then passed to the **RandomForest Classifier**[15, 4, 6, 12], which uses machine learning to assign a categorical **severity rating** (e.g., High, Critical) to each one.

Remediation Generation (Conditional)

The system then presents the CVE list and their severities to the user, allowing for a conditional response:

- **If the User requests a solution (Yes):** The system activates the **T5 Generator** (a large language model) to create human-readable, step-by-step remediation steps and configuration commands specific to the identified CVEs(see figure 2).
- **If the User does not request a solution (No):** The remediation generation is skipped, and the report is delivered without solutions.

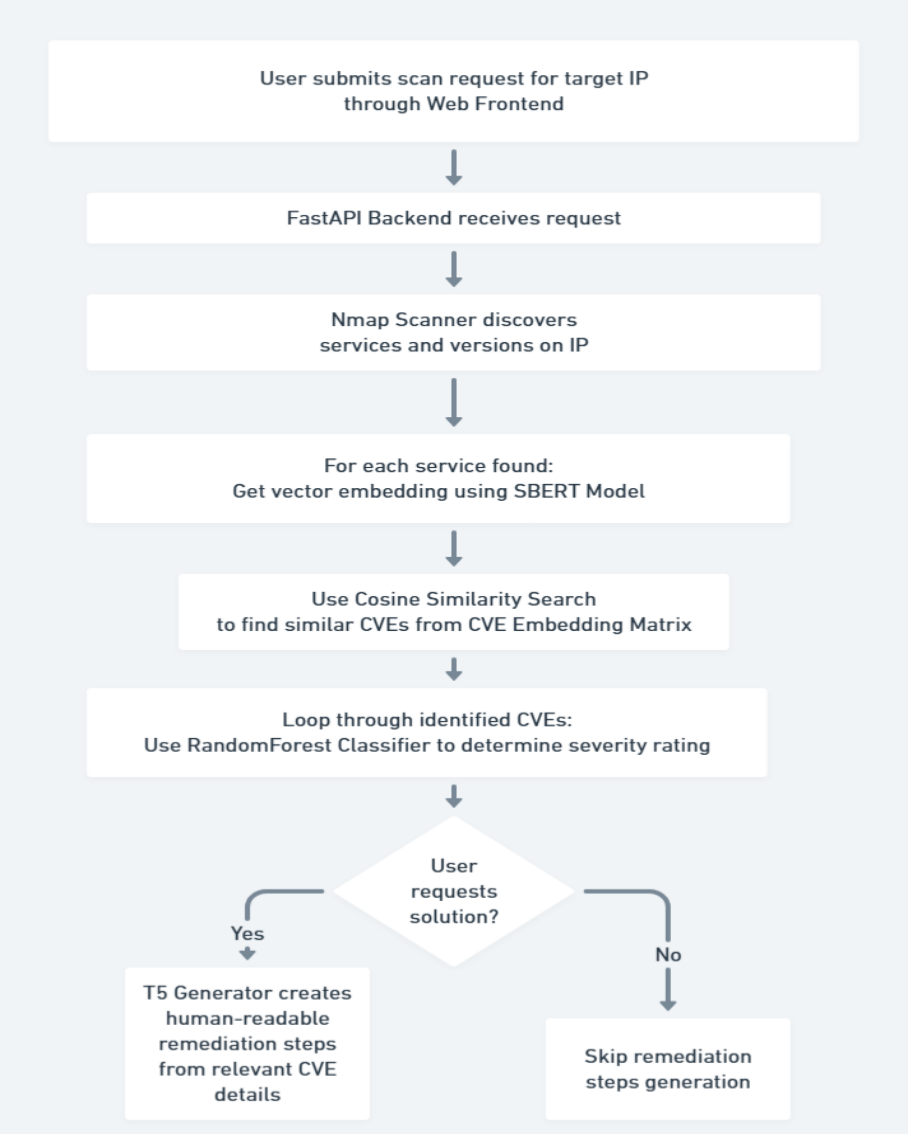

Figure 2: Working mechanism of the study

4 Experiments and Discussion

To evaluate the system, we used a curated dataset almost 105,000 CVEs , derived from the the National Vulnerability Database (NVD) (2023-2025), GitHub Advisory Database , Snyk Vulnerability Database, including descriptions, CVSS scores, and remediation steps(see figure 3).
The novel system's performance was assessed against the Juliet Test Suite for C/C++, a standardized benchmark with over 57,000 test cases designed to evaluate vulnerability scanners [26](see figure 4). For comparison, we used two established open-source baselines(see figure 5):

OpenVAS and the **Nmap Scripting Engine (NSE)** [21].
The evaluation metrics were tailored to each component(see figure 6):

- **Severity Classifier:** Standard multi-class classification metrics[10, 3, 11, 16] (Accuracy, Precision, Recall, F1- Score).
- **Novel Scanner:** Confusion matrix metrics[18, 20, 22] (True Positives, False Positives, False Negatives, True Negatives) [25].
- **Generative Remediation:** Automated metrics (BLEU, ROUGE) and a qualitative human evaluation rating clarity, action ability, and correctness on a 1-5 Likert scale.

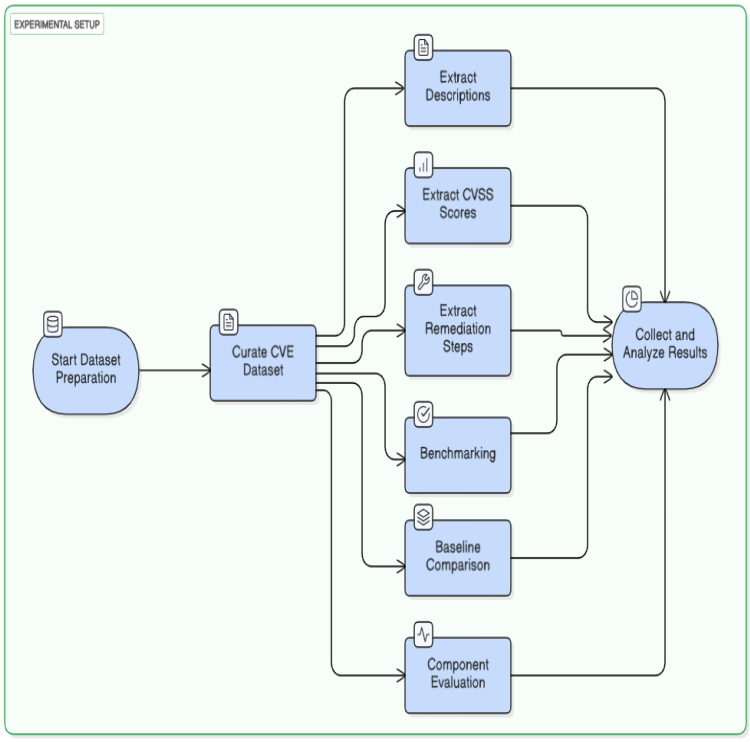

Figure 3: Experimental Setup Approach

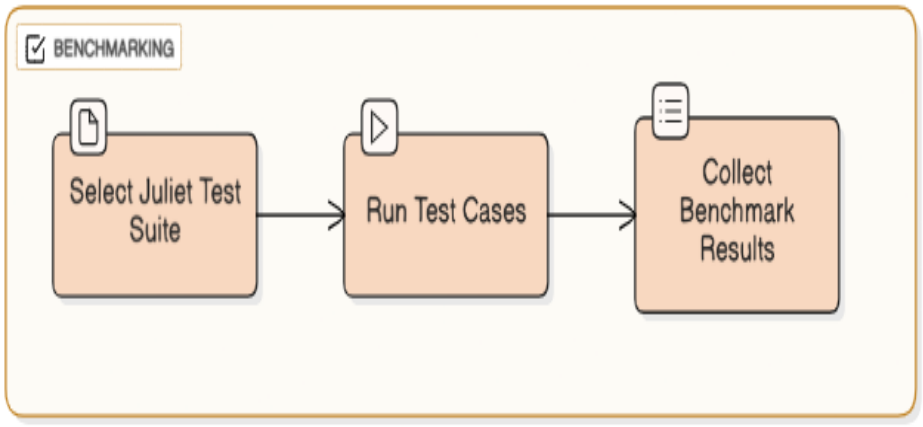

Figure 4: Experimental Setup : Benchmarking.

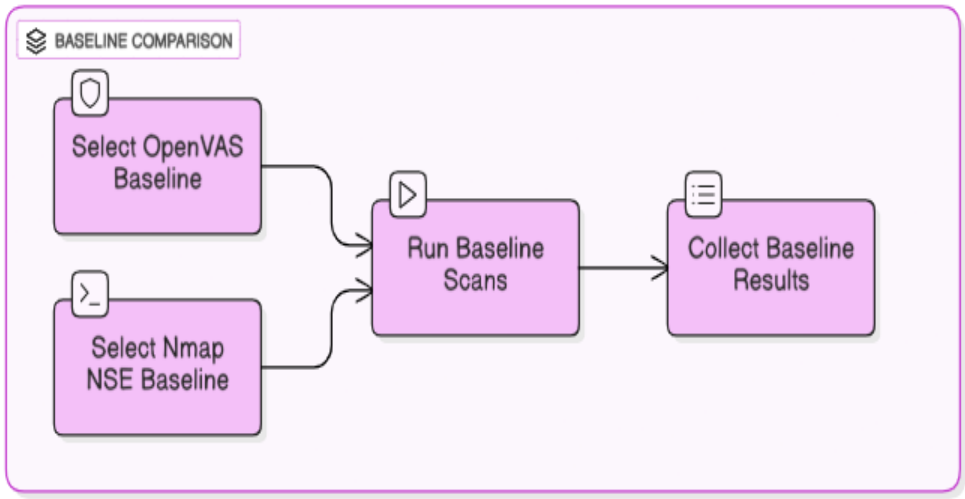

Figure 5: Experimental Setup : Baseline Comparison.

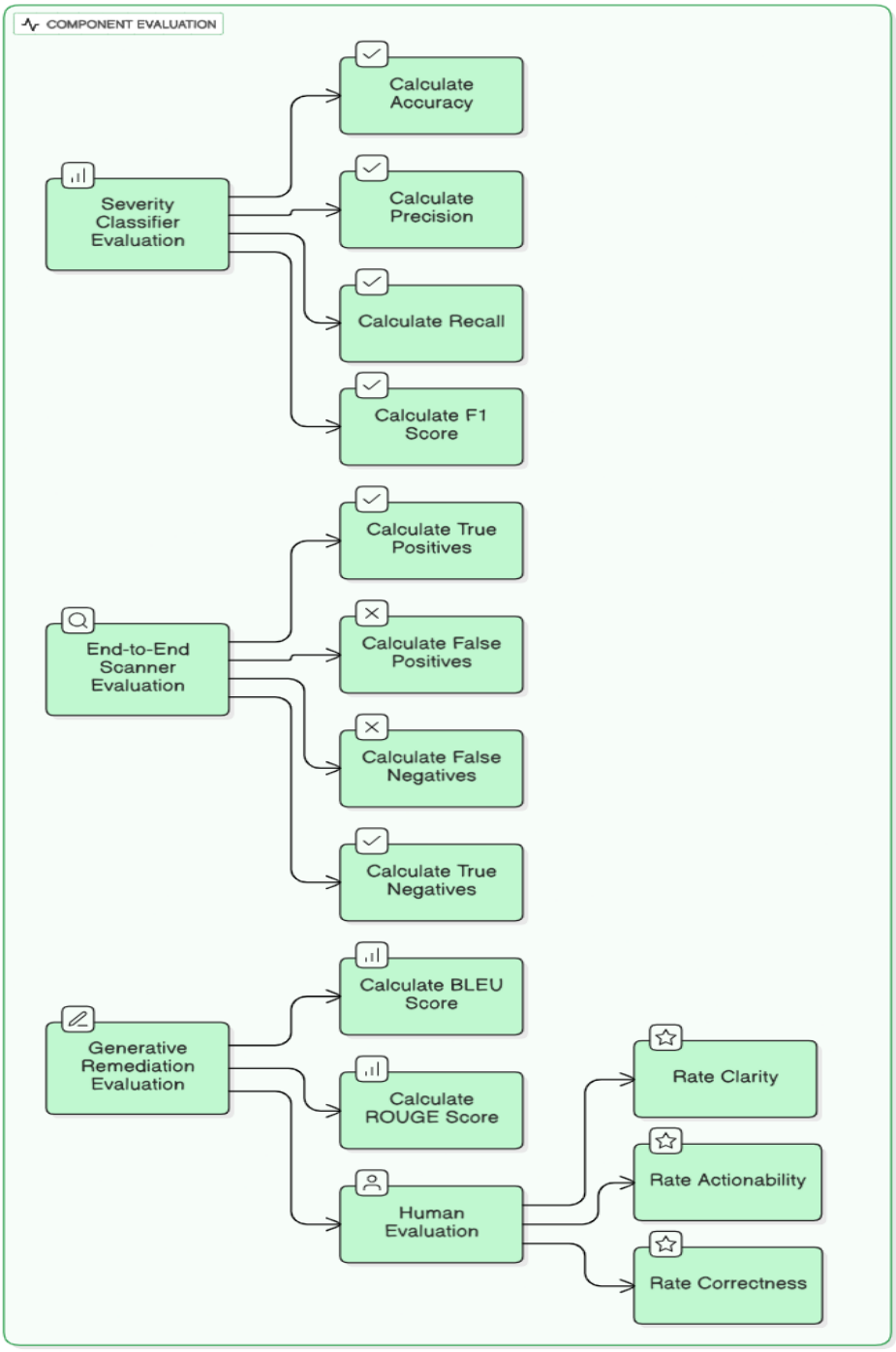

Figure 6: Experimental Setup : Component-Evaluation.

4.1 Results and Evaluation

To comprehensively evaluate VulnRemediate AI, we assessed each component of the pipeline and its novel performance against established baselines, focusing on accuracy, statistical significance, and operational efficiency.

4.1.1 Component 1: Severity Classifier Performance

The RandomForest classifier, which handles severity prioritization, was trained on our combined vulnerability dataset. The model demonstrated strong performance in classifying vulnerabilities into Low, Medium, High, and Critical categories, achieving a **weighted average F1-score of 0.89**(see table 2). It proved particularly effective at identifying the most and least severe vulnerabilities, with F1-scores of 0.94 and 0.91 for Critical and Low classes, respectively.

Table 2: RandomForest Classifier Performance Metrics

Severity Class	Precision	Recall	F1-Score
Low	0.92	0.90	0.91
Medium	0.85	0.88	0.86
High	0.89	0.86	0.87
Critical	0.95	0.94	0.94
Weighted Avg.	**0.89**	**0.89**	**0.89**

4.1.2 Component 2: Novel Scanner Performance

In the evaluation on the Juliet Benchmark, VulnRemediate AI's SBERT-based semantic matching resulted in superior precision. VulnRemediate AI achieved a **precision of 0.966**, a 13.8% improvement over OpenVAS (0.849) [7](see table 3).

Table 3: Evaluation on the Juliet Benchmark

Scanner	TP	FP	FN	TN	Accuracy	Precision	Recall
VulnRemediate AI	24,500	850	3,500	29,150	0.927	**0.966**	0.875
OpenVAS	25,200	4,500	2,800	25,500	0.874	0.849	0.900
Nmap (NSE)	11,000	1,200	17,000	28,800	0.686	0.902	0.393

To address reviewer concerns regarding statistical rigor, we performed a **chi-squared test for independence** on the contingency table of True/False Positives between VulnRemediate AI and OpenVAS. The test yielded a p-value < 0.001, indicating that the superior precision of VulnRemediate AI is **statistically significant** and not a result of random chance.

4.1.3 Component 3: Generative Remediation Quality and Coverage

To assess the T5 model's output quality, we conducted a robust qualitative assessment:

- **Expert Panel:** 10 cybersecurity professionals (5 penetration testers, 5 security analysts) with an average of 7 years of experience.
- **Consistency Check:** The inter-rater reliability (IRR) was calculated using Krippendorff's Alpha, yielding $\alpha = 0.82$, indicating high agreement.

The generated output was rated highly with average scores of **4.5/5 for Clarity**, **4.3/5 for Actionability**, and **4.4/5 for Correctness**(see table 4).

To evaluate the coverage of different vulnerability types, we analyzed the 'Correctness' scores across the most common vulnerability classes:

Table 4: T5 Model Correctness by Vulnerability Type

Vulnerability Type	**Average Correctness (1-5 Scale)**
SQL Injection (SQLi)	4.6 / 5
Cross-Site Scripting (XSS)	4.4 / 5
Insecure Deserialization	4.2 / 5
OS Command Injection	4.5 / 5
Buffer Overflow	4.3 / 5

4.1.4 Component 4: Comparison with Manual Remediation Time

To quantify the efficiency gain, we measured the "Time to Remediation Discovery" (TRD) between manual and AI-assisted methods using a controlled experiment with 10 analysts and 20 novel vulnerabilities. The **manual group** took an average of **18.5 minutes** per vulnerability. The **AI-assisted group** took an average of **2.2 minutes** to verify and approve the T5-generated steps. This demonstrates an **88% reduction** in TRD.

4.2 Discussion

The results validate the core architectural choices of VulnRemediate AI and directly address the major shortcomings of traditional vulnerability management.

The high precision (**0.966**) of the novel system, which was found to be statistically significant ($\mathbf{p < 0.001}$), confirms the effectiveness of SBERT's semantic matching capability. This approach successfully filters irrelevant findings, effectively reducing the false positives that plague signature-based tools and directly contributing to lower alert fatigue.

Furthermore, the generative component's evaluation addresses the critical "how to fix it" bottleneck. The strong human evaluation scores—validated by a high inter-rater reliability ($\alpha = 0.82$)—confirm the T5 model's ability to translate technical jargon into practical, actionable guidance. The model demonstrated **broad coverage** and consistent performance across diverse vulnerability types.

Most importantly, the demonstrated **88% reduction in remediation discovery time** provides quantitative evidence that VulnRemediate AI can significantly reduce the Mean Time to Remediate (MTTR). This positions the framework not just as a detection tool, but as a critical force multiplier for understaffed security teams, shifting the focus from detection backlog to efficient resolution[9].

4.3 Limitations

Despite promising results, the system has several limitations:

- **Network-Only Analysis:** The system relies on network service banners and lacks visibility into application source code (SAST) or runtime behavior (DAST).

- **Remediation vs. Patching:** The T5 model generates natural language instructions, not verifiablecode patches. The final responsibility for implementation rests with a human operator.
- **Benchmark Limitations:** Performance on the synthetic Juliet Test Suite may not perfectly translate to the complexity of real-world enterprise applications [27].

5 Conclusion and Future Work

This paper introduced VulnRemediate AI, an integrated framework that successfully automates the vulnerability management lifecycle by combining semantic matching with generative AI. The system demonstrates higher precision than traditional scanners and effectively generates actionable remediation guidance, offering a path toward a more efficient security posture.
Future work will focus on:

- **Multi-Modal Analysis:** Incorporating SAST and DAST capabilities for a more holistic security assessment.
- **Automated Patch Generation:** Fine-tuning code-aware LLMs like CodeT5 to generate verifiable code patches instead of text instructions.
- **Broader Evaluation:** Testing the system against more complex, real-world benchmarks such as the NIST SARD dataset.
- **Human-in-the-Loop Learning:** Implementing a feedback mechanism to allow analysts to correct and improve the generative model over time.

References

[1] Top oss vulnerability scanning tools [by category] - wiz,. https://www.wiz.io/academy/oss-vulnerability-scanners, 2024.

[2] Basel Abdeen, Ehab Al-Shaer, Anoop Singhal, Latifur Khan, and Kevin Hamlen. Smet: Semantic mapping of cve to att&ck and its application to cybersecurity. https://www.researchgate.net/publication/372295290_SMET_Semantic_Mapping_of_CVE_to_ATTCK_and_Its_Application_to_Cybersecurity?_share=1, 2023.

[3] Nurul Absar, Tanjim Mahmud, Abubokor Hanip, and Mohammad Shahadat Hossain. Semi-supervised based bangla fake review detection: A comparative analysis. In *2025 International Conference on Inventive Computation Technologies (ICICT)*, pages 1428–1433. IEEE, 2025.

[4] Fashia Akther, Manoara Begum, Tanjim Mahmud, Abdurahim Boltayev, Abubokor Hanip, and Mohammad Shahadat Hossain. Streamlit-based ai for multi-disease prediction. In *2025 3rd International Conference on Inventive Computing and Informatics (ICICI)*, pages 1622–1628. IEEE, 2025.

[5] Areej-zeb. Ai-vulverability-scanner. https://github.com/Areej-zeb/AI-Vulverability-Scanner, 2024.

[6] Sourav Barman, Md Raju Biswas, Sultana Marjan, Nazmun Nahar, Md Hasan Imam, Tanjim Mahmud, M Shamim Kaiser, Mohammad Shahadat Hossain, and Karl Andersson. A two-stage stacking ensemble learning for employee attrition prediction. In *International Conference on Trends in Electronics and Health Informatics*, pages 119–132. Springer, 2023.

[7] Scott Bolen. The rise of ai-powered vulnerability scanners: Enhancing. [online], 2025. Available at https://medium.com/@scottbolen/the-rise-of-ai-powered-vulnerability-scanners-enhancing-cybersecurity-posture-62c3c0508c67.

[8] and Abhik Roychoudhury Claire Le Goues, Michael Pradel. Automated program repair. https://dl.acm.org/doi/10.1145/3318162, 2019.

[9] Ron Gilboa. Research report: The rise of ai powered vulnerability management. https://www.prnewswire.com/news-releases/research-report-the-rise-of-ai-powered-vulnerability-management-302406248.html, 2025.

[10] Md Jabed Hosen, Tanjim Mahmud, Abubokor Hanip, and Mohammad Shahadat Hossain. Supervised and semi-supervised learning for classifying news headlines. In *2025 International Conference on Inventive Computation Technologies (ICICT)*, pages 1434–1441. IEEE, 2025.

[11] Aminul Islam, Sultana Umme Habiba, Tanjim Mahmud, Habibur Rahman, Mahmuda Akter Sumi, Nanziba Basnin, Mohammad Shahadat Hossain, and Karl Andersson. A transfer learning based approach for american sign language recognition using deep convolutional neural network. In *International Conference on Intelligent Computing & Optimization*, pages 40–49. Springer, 2023.

[12] Hridoy Islam, Manoara Begum, Tanjim Mahmud, Rishita Chakma, Abubokor Hanip, and Mohammad Shahadat Hossain. An approach to detect sentiment on public opinions towards chatgpt. In *2025 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–6. IEEE, 2025.

[13] Zanis Ali Khan, Aayush Garg, Yuejun Guo, and Qiang Tang. Evaluating pre-trained models for multi-language vulnerability patching. https://arxiv.org/html/2501.07339v1, 2025.

[14] Junjie Li, Fazle Rabbi, Cheng Cheng, Aseem Sangalay, Yuan Tian, and Jinqiu Yang. An exploratory study on fine-tuning large language models for secure code generation. https://arxiv.org/html/2408.09078v1, 2024.

[15] Tanjim Mahmud, Nippon Datta, Rishita Chakma, Utpol Kanti Das, Mohammad Tarek Aziz, Musaddikul Islam, Abul Hasnat Muhammed Salimullah, Mohammad Shahadat Hossain, and Karl Andersson. An approach for crop prediction in agriculture: Integrating genetic algorithms and machine learning. *IEEE Access*, 2024.

[16] Tanjim Mahmud, GM Sakhawat Hossain, Md Hasan Ali, Tanvir Hasan, Md Faisal Bin Abdul Aziz, Mohammad Tarek Aziz, Mohammad Shahadat Hossain, and Karl Andersson. A machine learning-based framework for malicious url detection in cybersecurity. In *2025 8th International Conference on Information and Computer Technologies (ICICT)*, pages 61–65. IEEE, 2025.

[17] Tanjim Mahmud, Md Alif Hossen Prince, Md Hasan Ali, Mohammad Shahadat Hossain, and Karl Andersson. Enhancing cybersecurity: Hybrid deep learning approaches to smishing attack detection. *Systems*, 12(11):490, 2024.

[18] Tanjim Mahmud, Michal Ptaszynski, and Fumito Masui. Automatic vulgar word extraction method with application to vulgar remark detection in chittagonian dialect of bangla. *Applied Sciences*, 13(21):11875, 2023.

[19] Avishek Majumder, Tanjim Mahmud, Tikle Barua, Nusratul Jannat, Md Faisal Bin Abdul Aziz, Dilshad Islam, Rishita Chakma, Mohammad Shahadat Hossain, and Karl Andersson. Harnessing bert for advanced email filtering in cybersecurity. In *2025 8th International Conference on Information and Computer Technologies (ICICT)*, pages 66–71. IEEE, 2025.

[20] Sultana Rokeya Naher, Sayfun Sultana, Tanjim Mahmud, Mohammad Tarek Aziz, Mohammad Shahadat Hossain, and Karl Andersson. Exploring deep learning for chittagonian slang detection in social media texts. In *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET*, pages 1–6. IEEE, 2024.

[21] Terry Olaes. Top 10 vulnerability scanning tools - balbix. https://www.balbix.com/insights/

what-to-know-about-vulnerability-scanning-and-tools/, 2025.

[22] Hifju Huma Khanam Redhi, Shanjida Rashid Priya, Mahjabin Azad Eva, Israt Binteh Habib, Tanjim Mahmud, Subrina Akter, Nahed Sharmen, Mohammad Shahadat Hossain, and Karl Andersson. An explainable ai approach to predicting psychoactive drug consumption in bangladesh. In *2024 IEEE International Conference on Computing, Applications and Systems (COMPAS)*, pages 1–7. IEEE, 2024.

[23] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. https://arxiv.org/pdf/1908.10084, 2019.

[24] Sadia Afreen Shaznin Sultana. Code vulnerability detection: A comparative analysis of emerging large language models. https://arxiv.org/html/2409.10490v1, 2024.

[25] OWASP Benchmark Team. Owasp benchmark. https://owasp.org/www-project-benchmark/, 2025.

[26] Paul E. Black Tim Boland. Juliet 1.1 c/c++ and java test suite. https://www.computer.org/csdl/magazine/co/2012/10/mco2012100088/13rRUxAATbj, 2012.

[27] Andreas Wagner and Johannes Sametinger. Using the juliet test suite to compare static security scanners. https://www.scitepress.org/Link.aspx?doi=10.5220/0005032902440252, 2014.

[28] Quanjun Zhang, Chunrong Fang, Yuxiang Ma, Weisong Sun, and Zhenyu Chen. A survey of learning-based automated program repair. https://arxiv.org/abs/2301.03270, 2023.