Final Review Seminar

Toward More Effective Deep Learning-based Automated Software Vulnerability Prediction, Classification, and Repair

Deep learning-based approaches can predict, classify, and repair vulnerable code functions and be integrated into IDEs to automate security testing during the early software development lifecycle.

Presenter:
Yeh (Michael) Fu

Supervisors:
Dr Chakkrit (Kla) Tantithamthavorn
Dr Trung Le

Faculty of Information Technology
Department of Software Systems and Cybersecurity
Monash University

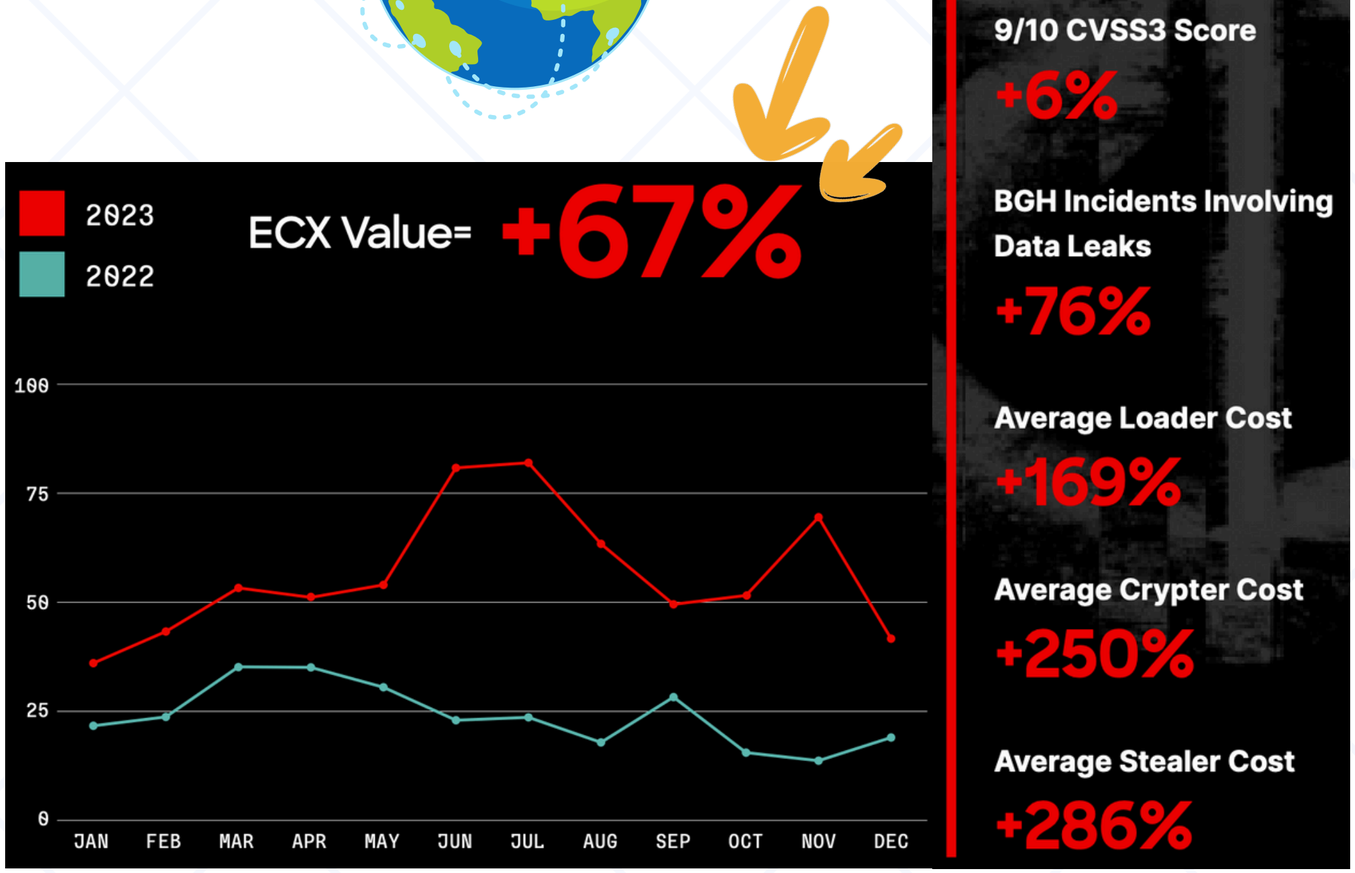
MONASH University

Overview

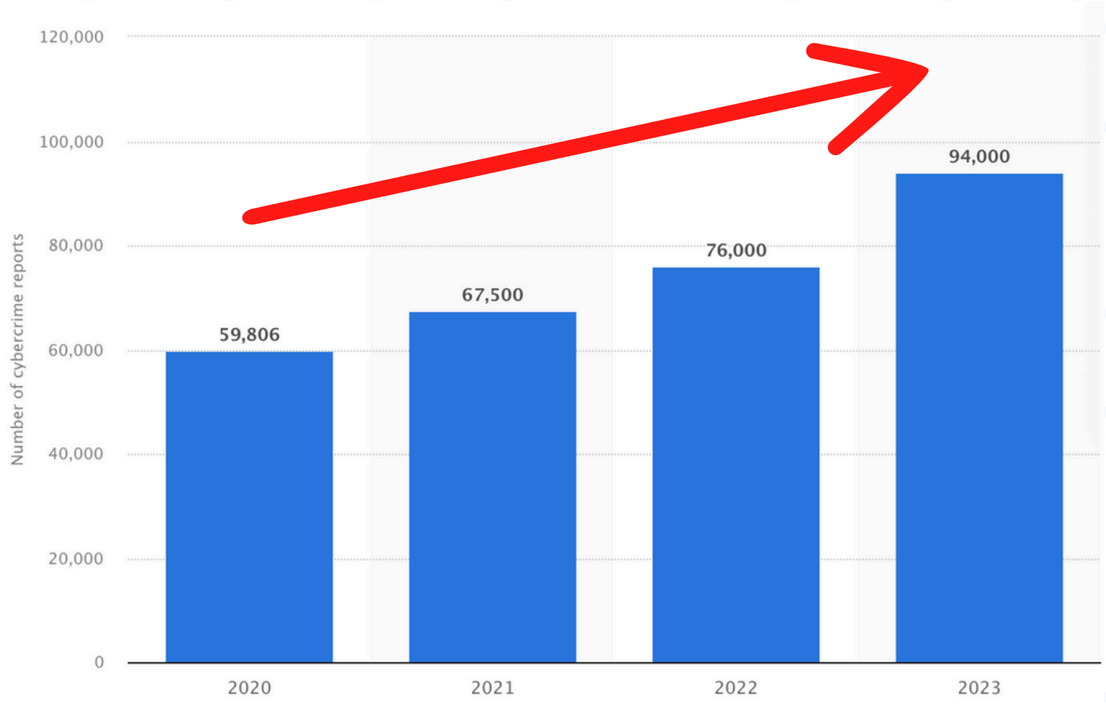
MONASH University

01

Thesis Background - Rising Security Threats



Source: CrowdStrike Global Threat Report 2024



Source: Australian Cybersecurity Centre

MONASH University

01

Why do vulnerabilities happen and what are their negative impacts?

1. **Bad Software Implementation**
2. **Obsolete Libraries and Dependencies**
3. **Poorly Implemented Code**
4. **Lack of Security Best Practices**

Common Software Vulnerability And Their Negative Impacts		
CWE-ID	**Description**	**Potential Impacts**
CWE-787	**Out-of-Bound Write**	Corruption of data, a crash, or code execution
CWE-79	**Cross-site Scripting**	Data theft, account hijacking, malware distribution
CWE-89	**SQL Injection**	Unauthorized data access, denial of service (DoS), data corruption
...	...	...

A security analyst is responsible for **identifying**, **analyzing** security vulnerabilities, and **proposing fixes** to software developers.

But... HOW do they achieve these tasks?

MONASH University

01

Traditional Vulnerability Analysis Workflow with Static Analysis Tools

Security Analyst

Analyze

klocwork

Checkmarx

Static Analysis Tool

Provide

Vulnerability Location

Manually classify

1

CWE™

Vulnerability Type

Manually estimate

2

CVSS

Characterizing and Scoring Vulnerabilities

Vulnerability Severity

Manually suggest

3

Vulnerability Repair

Labor Intensive

Report

Developer

MONASH University

01

Automated Vulnerability Analysis Workflow with Deep Learning Models

Software Vulnerability Detection

Vulnerability Location

Validate

Security Analyst

Software Vulnerability Classification

CWE

CVSS Characterizing and Scoring Vulnerabilities

Vulnerability Type

Vulnerability Severity

Report

Developer

Automated Vulnerability Repair

Vulnerability Repair

Labor Intensive

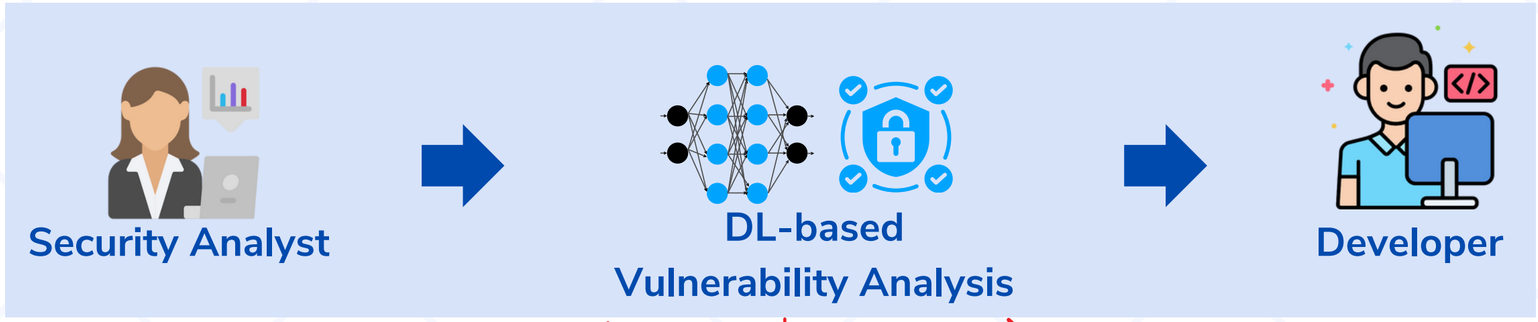
Automated Tool Support

MONASH University

02

Thesis Goal Derived From Existing Challenges

While **Automated** Vulnerability Analysis Workflow looks promising



We found that ...

Existing DL-based vulnerability predictions are **NOT**:

- ! Practical
- ! Explainable
- ! Actionable

Thesis Goal

Making DL-based vulnerability predictions more:

- ✓ Practical
- ✓ Explainable
- ✓ Actionable

while being deployed in a more **accessible** way

MONASH University

02

Thesis Problem Statement

Our approach to addressing the overarching thesis goal

01 Software Vulnerability Detection

02 Software Vulnerability Classification

03 Automated Vulnerability Repair

04 Software Security Tool in IDE

MONASH University

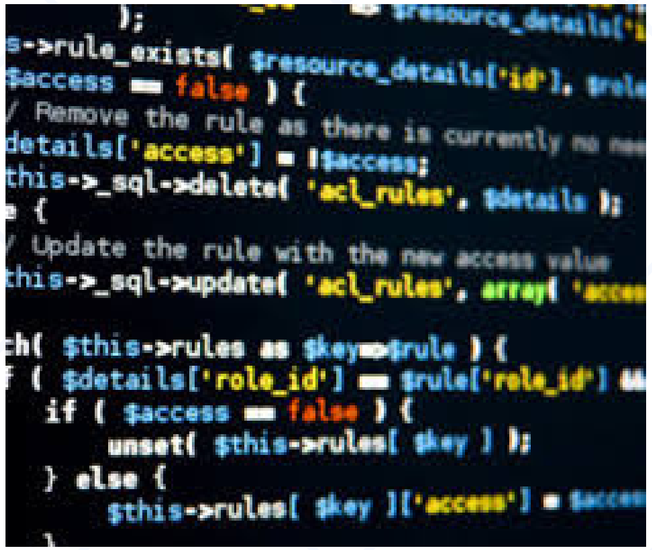
02

Thesis Problem Statement

Practical

01 Software Vulnerability Detection

Coarse-grained vulnerability detection
(file or function levels)

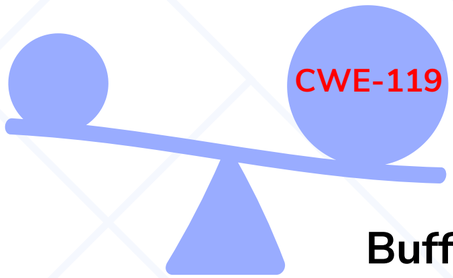


Still requires manual effort to
locate vulnerabilities

Explainable

02 Software Vulnerability Classification

Data imbalance



Buffer overflow

Suboptimal multi-task learning

CWE + CVSS

Characterizing and Scoring Vulnerabilities

Naive loss summary

MONASH University

02

Thesis Problem Statement

Actionable

Automated Vulnerability Repair

03

Inaccurate repair accuracy

20 - 30% repair accuracy

Still requires manual effort to refine inaccurate prediction

04

Accessible

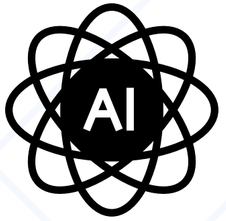
Software Security Tool in IDE

Rule-based static analysis tools

Accurate

Accessible

DL-based security tools



Accurate

Accessible

MONASH University

02 Thesis Problem Statement

Practical

01 Software Vulnerability Detection

RQ1: Can we help security analysts pinpoint the lines of code involved in the detected vulnerabilities?

Explainable

02 Software Vulnerability Classification

RQ2: Can we help security analysts identify vulnerability types more accurately to provide further explanations of the detected vulnerabilities?

Actionable

03 Automated Vulnerability Repair

RQ3: Can we help security analysts suggest repair patches for vulnerable code more accurately?

Accessible

04 Software Security Tool in IDE

RQ4: Can we help security analysts detect and localize vulnerabilities, identify vulnerability types, and suggest corresponding repair patches during the early stage of software development?

MONASH University

Summary of the Eight Research Contributions

Research Question (RQ)	Contribution
RQ1: Can we help security analysts **pinpoint the lines of code** involved in the detected vulnerabilities?	**RC1-LineVul (MSR'22) (Accepted)** **RC2-OptiMatch (TSE'24) (Under Review)**
RQ2: Can we help security analysts **identify vulnerability types more accurately** to provide further explanations of the detected vulnerabilities?	**RC6-AIBugHunter (EMSE'23) (Accepted)** **RC3-VulExplainer (TSE'23) (Accepted)**
RQ3: Can we help security analysts **suggest repair patches for vulnerable code more accurately**?	**RC4-VulRepair (FSE'22) (Accepted)** **RC5-VQM (TOSEM'23) (Accepted)**
RQ4: Can we help security analysts **detect and localize vulnerabilities, identify vulnerability types, and suggest corresponding repair patches** during the early stage of software development?	**RC6-AIBugHunter (EMSE'23)**

Evaluate ChatGPT's performance for vulnerability prediction tasks
RC7-ChatGPT for Vulnerability Predictions (APSEC'23) (Accepted)

A systematic literature review for AI for DevSecOps.
RC8-AI For DevSecOps (SLR) (Waiting Final Decision - TOSEM)

MONASH University

03

A Quick Recap of Research Contributions
Covered in Our Confirmation/Progress Review Seminar

- RC1 - LineVul (RQ1) (Accepted - MSR’22)
- RC3 - VulExplainer (RQ2) (Accepted - TSE’23)
- RC4 - VulRepair (RQ3) (Accepted - FSE’22)
- RC5 - VQM (RQ3) (Accepted - TOSEM’23)
- RC6 - AIBugHunter (RQ2 & RQ4) (Accepted - EMSE’23)
- RC7 - ChatGPT for Vulnerability Predictions (Accepted - APSEC’23)

03

RC1 - LineVul: A Transformer-based Line-level Vulnerability Detection Approach

RQ1

Can we help security analysts pinpoint the lines of code involved in the detected vulnerabilities?

Contribution: LineVul

Self Attention

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

Line-level Vulnerability

Our solution to **coarse-grained vulnerability detection**

- Achieve **91% F1-Score** for **function-level** predictions, which is 160%-379% better than other baselines.
- Achieve **65% of Top-10 Accuracy** for **line-level** predictions, which is 12%-25% more accurate than other baselines.
- **26%-85% more cost-effective** for line-level vulnerability localization

More Accurate

Less Effort

Practical

MONASH University

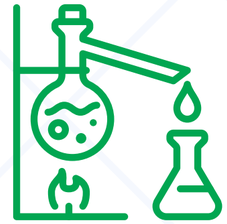
03

RC3 - VulExplainer: A Transformer-based Hierarchical Distillation for Explaining Vulnerability Types

RQ2

Can we help security analysts identify vulnerability types more accurately to provide further explanations of the detected vulnerabilities?

Contribution: VulExplainer CWE



Our solution to Data Imbalance - A hierarchical knowledge distillation framework

Step 1: Group similar CWE-IDs based on CWE abstract types **More similar and balanced**

Step 2: Train a TextCNN teacher model on each group (can only predict specific CWE-IDs)

Step 3: Distil teacher models and build an accurate student model to predict all CWE-IDs

MONASH University

03

RC3 - VulExplainer: A Transformer-based Hierarchical Distillation for Explaining Vulnerability Types

RQ2 **Can we help security analysts identify vulnerability types more accurately to provide further explanations of the detected vulnerabilities?**

Contribution: VulExplainer

Quantitative Evaluation

- **5%-29% more accurate** than other pre-trained transformer baselines (i.e., CodeBERT, GraphCodeBERT, CodeGPT) for predicting CWE-IDs
- Outperform loss-based methods for imbalance data such as **focal loss and logit adjustment**
- Our distillation framework **can be applied to any types of transformer**, i.e., encoder-only, decoder-only, encoder-decoder

Explainable

MONASH University

03

RC4 - VulRepair: A T5-based Automated Vulnerability Repair

RQ3

Can we help security analysts suggest repair patches for vulnerable code more accurately?

Contribution: VulRepair

- **Pre-trained on large code base** consists of millions of functions
- Use **subword-level tokenizer** to tackle OOV problem
- Achieve the best Percentage of Perfect Prediction of **44%**

Actionable

We found that...

+ pre-training on both NL and PL is the most beneficial for the AVR task and
+ using subword tokenizer is beneficial for the AVR task

MONASH University

03

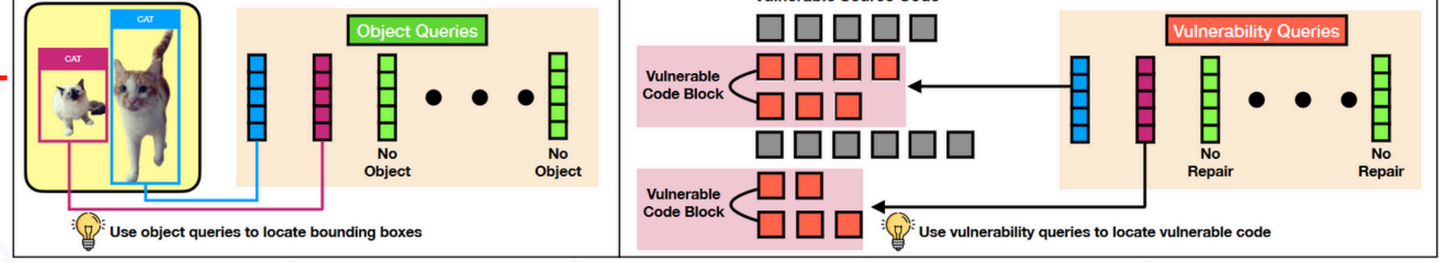
RC5 - Vision Transformer-Inspired Automated Vulnerability Repair

RQ3

Can we help security analysts suggest repair patches for vulnerable code more accurately?

Vulnerable Function — CWE-787 (Out-of-bounds Write)

```
41  41  GPMF_ERR IsValidSize(GPMF_stream *ms, uint32_t size)
42  42  {
43  43      if (ms)
44  44      {
45  45          int32_t nestsize = (int32_t)ms->nest_size[ms->nest_level];
46  46          if (nestsize == 0 && ms->nest_level == 0)
47  47              nestsize = ms->buffer_size_longs;
50  50      }
51  51      return GPMF_ERROR_BAD_STRUCTURE;
52  52  }
```



MONASH University

03

RC5 - Vision Transformer-Inspired Automated Vulnerability Repair

RQ3

Can we help security analysts suggest repair patches for vulnerable code more accurately?

Contribution: Vulnerability Query Masking (VQM)

Quantitative Evaluation

- VQM **further improves the performance** of our previous proposal, VulRepair, and achieves the best repair accuracy among all baseline methods.
- Our **ablation study** confirms the effectiveness of our proposed vulnerability query and masks

MONASH University

03

RC5 - Vision Transformer-Inspired Automated Vulnerability Repair

RQ3

Can we help security analysts suggest repair patches for vulnerable code more accurately?

Contribution: Vulnerability Query Masking (VQM)

Qualitative Evaluation - A online survey study with 71 software practitioners

Are AI-generated vulnerability repairs practical to security analysts and developers?

- **86%** of participants perceive AI-generated vulnerability repairs as **useful**.
- **80%** of the participants **consider adopting AI-generated repairs** if they are readily available and free of charge.

Actionable

MONASH University

03

RC6 - AIBugHunter: A Practical Tool for Predicting, Classifying and Repairing Software Vulnerabilities

RQ2 **Can we help security analysts identify vulnerability types more accurately to provide further explanations of the detected vulnerabilities?**

A multi-objective method using Frank–Wolfe algorithm

Explainable

Achieve the best accuracy among all other baselines

RQ4 **Can we help security analysts detect and localize vulnerabilities, identify vulnerability types, and suggest corresponding repair patches during the early stage of software development?**

AIBugHunter: A DL-based vulnerability analysis tool publicly available in VSCode

Accessible

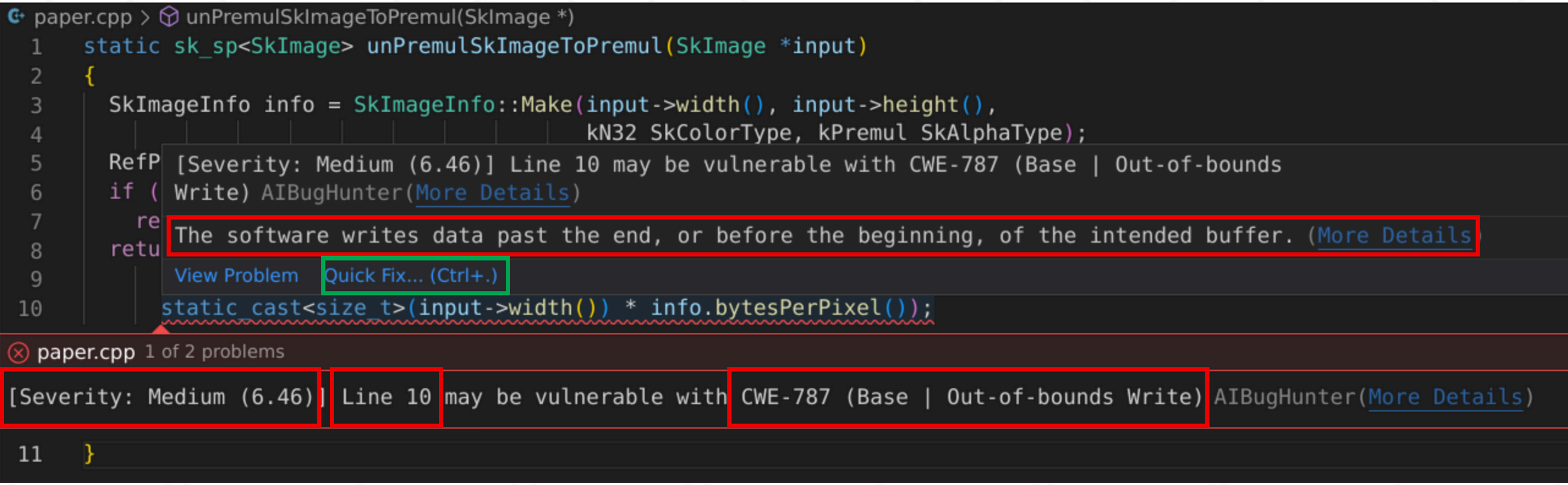
Integrate all of our DL-based approaches for software vulnerability predictions

MONASH University

03

AIBUGHUNTER

RQ4 Can we help security analysts detect and localize vulnerabilities, identify vulnerability types, and suggest corresponding repair patches during the early stage of software development?



MONASH University

03

RC6 - AIBugHunter: A Practical Tool for Predicting, Classifying and Repairing Software Vulnerabilities

Qualitative Evaluation

Survey Study (21 participants)

Our survey study with 21 software practitioners found that the four vulnerability predictions provided by AIBugHunter are perceived as **useful**.

90% of the respondents **consider adopting AIBugHunter** in their IDE.

User Experiment (6 participants) (not overlapped with the survey study)

Our AIBugHunter can reduce the time spent on locating, explaining, and repairing vulnerabilities from 10-15 minutes to 3-4 minutes.

MONASH University

03

RC7 - ChatGPT for Vulnerability Prediction, Classification, and Repair: How Far Are We?

Contribution: Evaluate ChatGPT's (both GPT3.5 and GPT4) performance on (1) SVD, (2) SVC, (3) Severity Estimation, and (4) AVR tasks

When comparing with smaller fine-tuned models such as CodeBERT, GraphCodeBERT, and CodeT5, zero-shot ChatGPT achieves the lowest performance for vulnerability prediction, classification, and estimation tasks.

Zero-shot ChatGPT fails to generate correct repair patches for the vulnerability repair task

We found that...

Even though ChatGPT is much larger than code-pre-trained language models like CodeBERT (approximately 14,000 times larger), it's still **crucial to fine-tune ChatGPT in order to make it effective in predicting vulnerabilities**.

04 Research Contribution in This Final Review Seminar

RC2 - OptiMatch (RQ1) (Under Review - TSE'23)

RC8 - AI For DevSecOps (SLR) (Waiting Final DecisionTOSEM)

04

RC2 - OptiMatch: Optimal Transport for Line-Level Vulnerability Detection

RQ1

Can we help security analysts pinpoint the lines of code involved in the detected vulnerabilities?

Detecting line-level vulnerabilities is challenging due to:

1. Vulnerabilities can appear in different parts of a function, affecting multiple lines of code

2. Developers' diverse coding styles

04

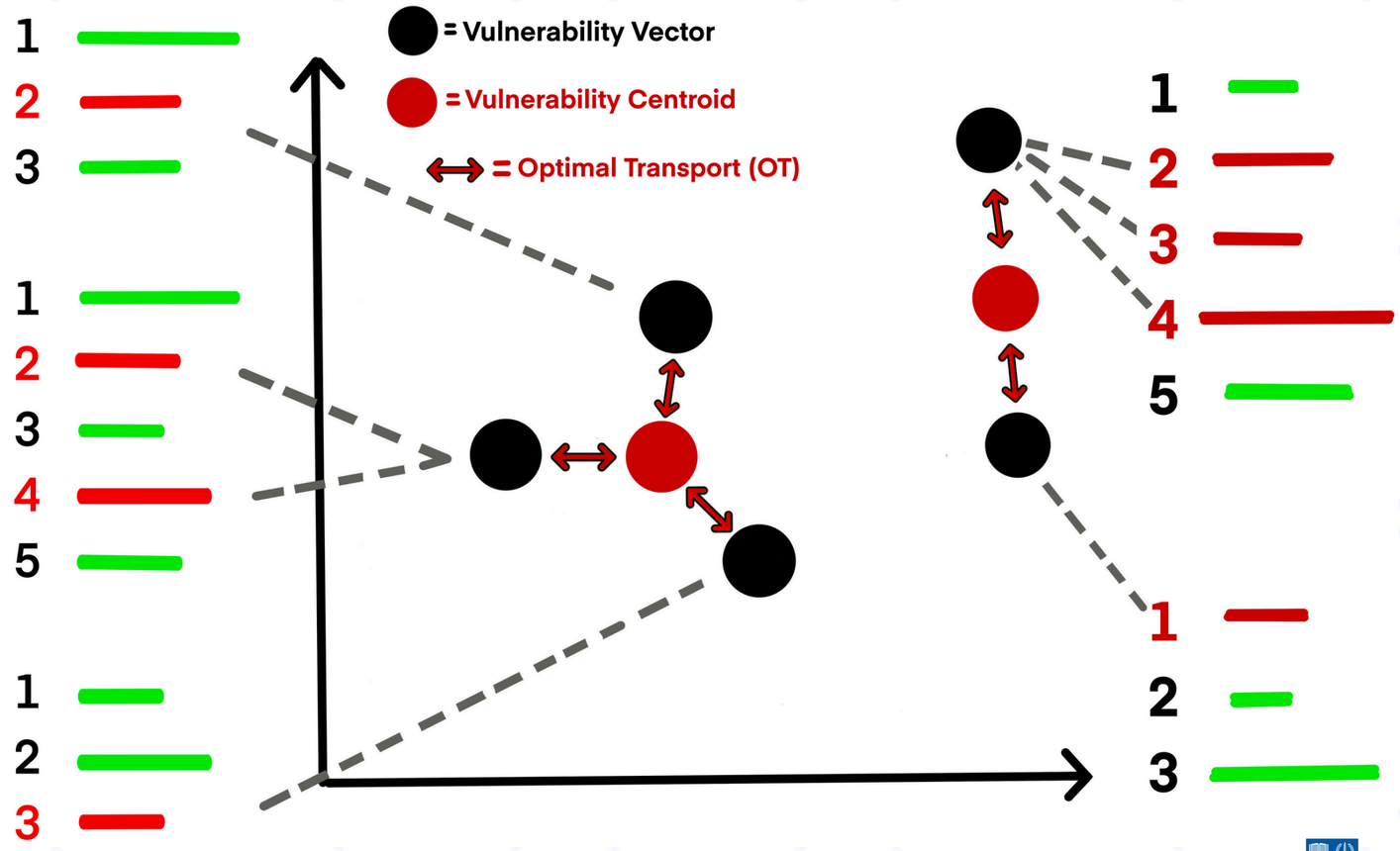
RC2 - OptiMatch: Optimal Transport for Line-Level Vulnerability Detection

RQ1

Learning Vulnerability CodeBook (Training)

Our observation: The embeddings of vulnerable lines tend to form clusters in the feature space with similar embeddings grouped closely

we propose using **vector quantization (VQ)** and **optimal transport (OT)** to capitalize on the observed cluster characteristics of vulnerable line embeddings



MONASH University

04

RQ1

RC2 - OptiMatch: Optimal Transport for Line-Level Vulnerability Detection

Static analysis tools

Match

Pre-defined rules

Locate

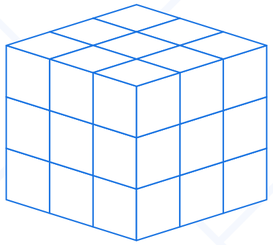
Vulnerabilities

Matching Learned Vulnerability CodeBook (Inference)

Our Method

OptiMatch

Match



Vulnerability Codebook

Locate

Vulnerabilities

MONASH University

04

RC2 - OptiMatch: Optimal Transport for Line-Level Vulnerability Detection

RQ1

Can we help security analysts pinpoint the lines of code involved in the detected vulnerabilities?

Contribution: OptiMatch

Quantitative Evaluation

- Our approach achieves **the highest F1 score for function and line-level** vulnerability detection compared to 12 other competitive baselines.
- Our ablation study **confirms the effectiveness of the proposed components** in our OptiMatch: VQ & OT, and vulnerability matching

Practical

MONASH University

04

RC8 - AI For DevSecOps: A Landscape and Future Opportunities

Motivation

1. Lay the groundwork for research beyond this thesis
2. Why do AI-driven security approaches matter to DevSecOps?
   Ensure security integration in DevOps without compromising agility

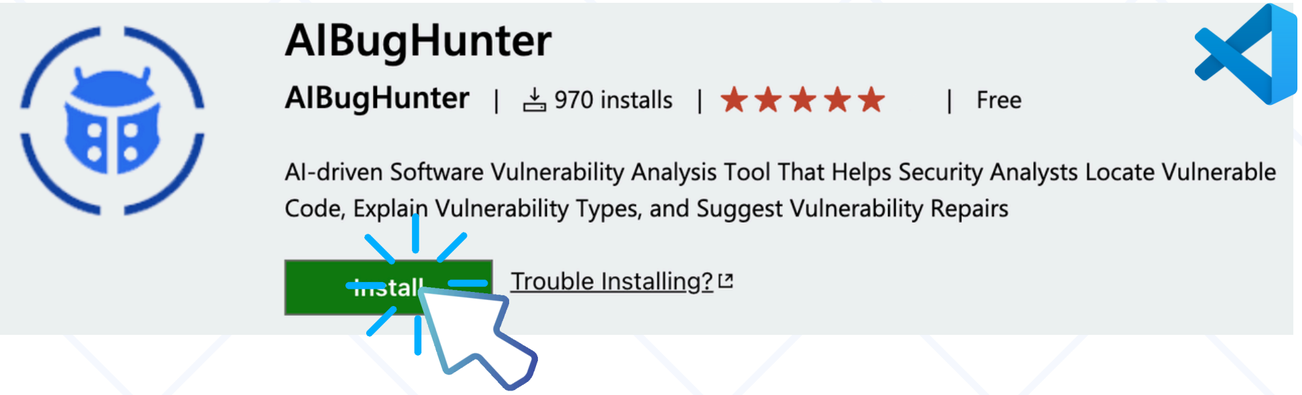
Contribution

- **Landscape of AI-Driven Security Approaches for DevSecOps**: Reviewed 99 peer-reviewed papers and identified 65 existing benchmark datasets used to evaluate AI-driven security approaches.
- **Identification of 15 Challenges and 15 Future Research Opportunities**

MONASH University

05 Research Impact

Tangible Research Output 01



Used by open-source developers and security analysts

02 Media Coverage

1.Gizmodo
2.Australian Cyber Security Magazine
3.Monash University
4.Mirage News
5.National Tribune
6.TechXplore
7.India Education Diary
8.MOEARA
9.Australian Computer Society
10.Cybersecurity Connect

MONASH University

05

Research Impact

Support Open Science 03 229 stars & 65 forks on GitHub to date

LineVul Public

A Transformer-based Line-Level Vulnerability Prediction

C++ • MIT License • 38 • 98 • 3 • 0 • Updated last month

VulRepair Public

VulRepair: A T5-Based Automated Software Vulnerability Repair

Python • MIT License • 12 • 64 • 2 • 0 • Updated on Jul 7, 2023

AIBugHunter Public

AIBugHunter: A Practical Tool for Predicting, Classifying and Repairing Software Vulnerabilities

vulnerability-prediction vulnerability-localization vulnerability-classification vulnerability-repair

Python • MIT License • 7 • 37 • 0 • 0 • Updated on Apr 10

VQM Public

Vision Transformer-Inspired Automated Vulnerability Repair

Python • MIT License • 3 • 9 • 2 • 0 • Updated on Nov 24, 2023

ChatGPT4Vul Public

Python • MIT License • 2 • 8 • 0 • 0 • Updated on Nov 24, 2023

VulExplainer Public

VulExplainer: A Transformer-based Hierarchical Distillation for Explaining Vulnerability Types

Python • MIT License • 3 • 12 • 0 • 0 • Updated on Nov 24, 2023

MONASH University

optimatch Public

Optimal Transport for Function-Level and Line-Level Vulnerability Detection

Python • MIT License • 0 • 1 • 0 • 0 • Updated on May 14

Awesome-AI4DevSecOps Public

MIT License • 0 • 0 • 0 • 0 • Updated 2 days ago

06

Research Community Service

1. Program Committee Roles

- PC for the SVM workshop co-located with **ICSE 2025**
- Shadow PC in the research track of **ICSE 2025**
- PC in the Artifact Evaluation Track of **ICSE 2024**
- PC in the doctoral symposium track of **DSN 2024**
- PC in the technical paper track of **MSR 2024***(Distinguished Reviewer Award)*
- Junior PC in the technical paper track of **MSR 2023**
- Web Co-Chair for **MSR 2023**
- PC in the Artifact Evaluation Track of **ASE 2022**

2. Review and Editorial Work

- Invited Reviewer of 4 Papers Submitted to **TSE** Journal 2024

3. Conference and Event Participation

- Student Volunteer at **ICSE 2023 and 2024**
- Presenter/Staff at **Monash University Open Day** Event 2022, 2023, and 2024
- Presenter at **TechFutureFest Industrial Event** Hosted by Monash University 2022

MONASH University

07

Training Progress

Activity	Status
Monash Doctoral Program - Compulsory Module	
IT Thesis Writing 1, 2, 3, 4 Academic Publishing in Information Technology IT: Ethical Research in IT	
Professional Development Excellence in Research & Teaching	97 / 60 hours
Professional Development Professionalism, Career & Innovation	61 / 60 hours
Total	**158 / 120 hours (100%)**

MONASH University

08

Conclusion

RQ1 Can we help security analysts pinpoint the lines of code involved in the detected vulnerabilities?

- LineVul (Accepted at MSR'22)
- OptiMatch (Under Review at TSE)

Practical

RQ2 Can We help security analysts identify vulnerability types more accurately to provide further explanations of the detected vulnerabilities?

- A Multi-Objective Optimization Approach (Accepted at EMSE'23)
- Data grouping and knowledge distillation to tackle imbalanced data problem (Accepted at TSE'23)

Explainable

RQ3 Can we help security analysts suggest repair patches for vulnerable code more accurately?

- VulRepair (Accepted at FSE'22)
- Vulnerability Query Masking (Accepted at TOSEM'23)

Actionable

RQ4 Can we help security analysts detect and localize vulnerabilities, identify vulnerability types, and suggest corresponding repair patches during the early stage of software development?

- AIBugHunter (Accepted at EMSE'23)

Accessible

Evaluate ChatGPT's performance for vulnerability prediction tasks (Accepted at APSEC'23)

A systematic literature review of AI for DevSecOps (Waiting Final Decision, TOSEM)

Acknowledging the use of generative artificial intelligence

I used ChatGPT (gpt-3.5-turbo/gpt-4o-mini/gpt-4o) to help polish the English and rephrase some of the paragraphs in this thesis to improve fluency and grammar. The tool was used to provide suggestions for rephrasing and grammatical improvements across multiple drafts. The output from this tool was carefully reviewed and modified to align with the intended meaning and context of the content.

MONASH University

MONASH University

Thank You