INSA
CENTRE VAL DE LOIRE

Institut National des Sciences Appliquées

Transformer-based Vulnerability Detection in Code at EditTime : Zero-shot, Few-shot, or Fine-tuning ?

Étude Bibliographique

Auteurs de l'article :
Aaron Chan
Anant Kharkar
Roshanak Zilouchian Moghaddam
Yevhen Mohylevskyy
Alec Helyar
Eslam Kamal
Mohamed Elkamhawy
Neel Sundaresan

Auteurs de l'étude :
Mohamed Mokrani
Lamiaa Benejma
Mouna El Arraf
Thomas Aubin

Responsable du module :
Pascal Berthomé
Relecteurs :
Prénom Nom
Prénom Nom

Si la détection de vulnérabilités logicielles est désormais universellement assistée par IA et en particulier par l'utilisation de LLM, ces derniers présentent une marge de progression importante lors des phases de développement. Nous étudions en quoi les approches proposées par l'article sont novatrices
Mots-clés : Transformeurs, Vulnerabilités logicielles, Détection de vulnérabilités

25 mars 2025

Résumé

Table des matières

Résumé **1**

Introduction **6**

I Contexte et problématique **1**

I Présentation du domaine et des enjeux en cybersécurité **2**
I.1 . . . 2
I.1.1 . . . 2
I.1.1.1 . . . 2

II Importance de la détection des vulnérabilités **4**
II.1 . . . 4
II.1.1 . . . 4
II.1.1.1 . . . 4
II.2 . . . 5

III Problèmes des méthodes classiques et défis posés par la détection en temps réel **6**
III.1 . . . 6
III.1.1 . . . 6
III.1.1.1 . . . 6

II Apports scientifiques de l'article **7**

IV Explication des trois approches (Zero-shot, Few-shot, Fine-tuning) **8**
IV.1 . . . 8
IV.1.1 . . . 8
IV.1.1.1 . . . 8

V Présentation des modèles utilisés (CodeBERT, Code-Davinci-002, Text-Davinci-003) **9**
V.1 . . . 9
V.1.1 . . . 9
V.1.1.1 . . . 9

VI Expérimentations et résultats observés **10**
VI.1 . . . 10
VI.1.1 . . . 10
VI.1.1.1 . . . 10

III Impacts et applications **11**

VII Améliorations du développement logiciel : **12**
VII.1 Des outils de détection classique : quel point de départ ? . . . 13
VII.1.1 Détection de vulnérabilités par IA : un bref état de l'art . . . 13
VII.1.1.1 Copilot : brève analyse de l'existant . . . 13
VII.1.1.2 Tabnine : un exemple de standard industriel pour la sécurisation de code par IA 13
VII.1.2 Cas particulier du code généré par des LLM . . . 13
VII.1.2.1 Pratiques actuelles de développement par LLM : exemple d'IntelliCode . . . 14

VII.2 Correction et complétion pendant la phase de développement : promesses et difficultés rencontrées 14
VII.3 Interprétation des métriques de classification présentées 14

VIII Études de cas et intégration dans un IDE 15
VIII.1 Résultats préliminaires avec un déploiement des modèles sur VSCode 15
VIII.1.1 Méthodologie inhérente au déploiement 15
VIII.1.2 Résultats obtenus 15
VIII.1.3 Cas de figure non ou partiellement couverts par l'étude 15

IX Conséquences pour l'industrie et la recherche en cybersécurité : 16
IX.1 16
IX.1.1 16
IX.1.1.1 16
IX.2 16
IX.3 Continuité de la recherche 16
IX.3.1 Sélection et préparation des données : des méthodes encourageantes 16

IV Analyse critique et perspectives 17

X Problèmes éthiques et limites des modèles d'IA 18
X.1 18
X.1.1 18
X.1.1.1 18

XI Biais, responsabilité et risques d'utilisation malveillante 19
XI.1 19
XI.1.1 19
XI.1.1.1 19

XII Suggestions d'améliorations et directions futures 20
XII.1 20
XII.1.1 20
XII.1.1.1 20

Conclusion 20

Annexe 1 : 1

Annexe 2 : 2

V Bibliographie 3

Table des figures

I.1 Exemple de figure 2
I.2 Exemple avec plusieurs figures 3

VII.1 Distribution of LLM usages in security domains [10] 13

Liste des tableaux

I.1 Exemple de tableau 3
I.2 Exemple de tableau coloré 3

Table des Équations

I.1 Une équation simple 2

Table des éléments de code

I.1 Un code Python 2

Introduction

Première partie

Contexte et problématique

Chapitre I

Présentation du domaine et des enjeux en cybersécurité

I.1

I.1.1

I.1.1.1

$$a + b = c \tag{I.1}$$

Figure I.1 – Exemple de figure

```
print("This line will be printéd.")
print("Another line to print."
```

Listing I.1 – Un code Python

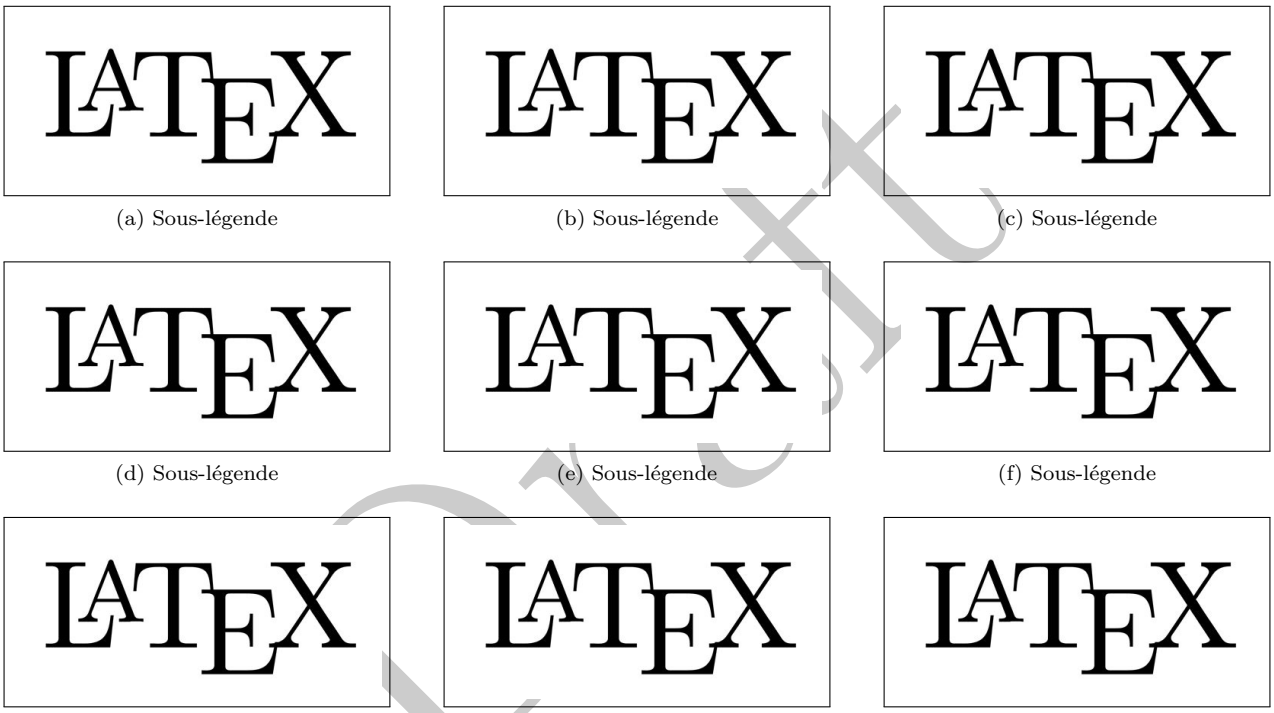
Ceci est un exemple d'encadré. Il sert à mettre en évidence des parties importantes du rapport

Donnée

TABLE I.1 – Exemple de tableau

Tâche				
Donnée			0	0

TABLE I.2 – Exemple de tableau coloré



(a) Sous-légende (b) Sous-légende (c) Sous-légende

(d) Sous-légende (e) Sous-légende (f) Sous-légende

(g) Sous-légende (h) Sous-légende (i) Sous-légende

FIGURE I.2 – Exemple avec plusieurs figures

Chapitre II

Importance de la détection des vulnérabilités

II.1

II.1.1

II.1.1.1

II.2

some text

Chapitre III

Problèmes des méthodes classiques et défis posés par la détection en temps réel

III.1

III.1.1

III.1.1.1

Deuxième partie

Apports scientifiques de l'article

Chapitre IV

Explication des trois approches (Zero-shot, Few-shot, Fine-tuning)

IV.1

IV.1.1

IV.1.1.1

Chapitre V

Présentation des modèles utilisés (CodeBERT, Code-Davinci-002, Text-Davinci-003)

V.1

V.1.1

V.1.1.1

Chapitre VI

Expérimentations et résultats observés

VI.1

VI.1.1

VI.1.1.1

Troisième partie

Impacts et applications

Chapitre VII

Améliorations du développement logiciel :

VII.1 Des outils de détection classique : quel point de départ ?

VII.1.1 Détection de vulnérabilités par IA : un bref état de l'art

VII.1.1.1 Copilot : brève analyse de l'existant

VII.1.1.2 Tabnine : un exemple de standard industriel pour la sécurisation de code par IA

VII.1.2 Cas particulier du code généré par des LLM

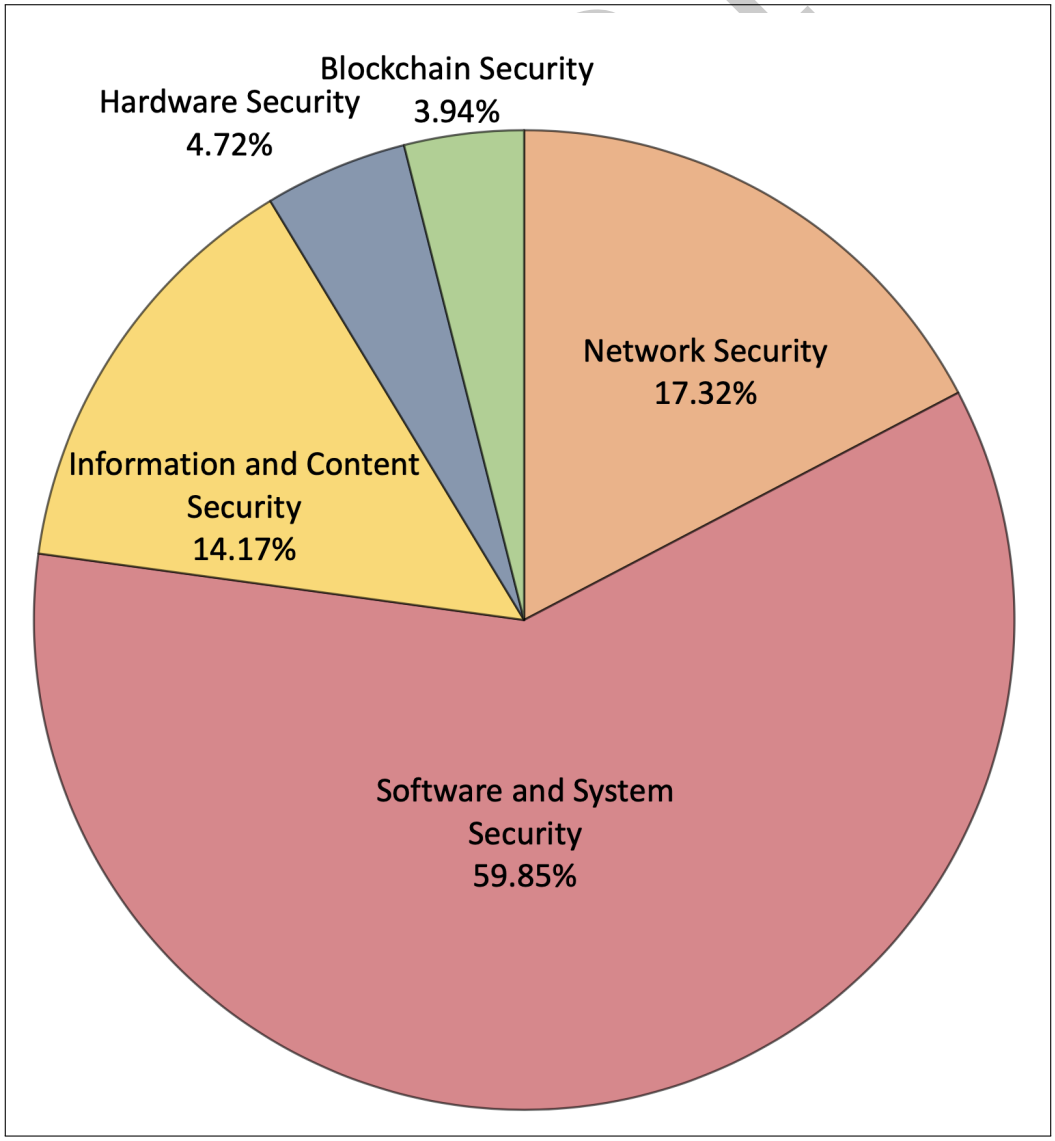


FIGURE VII.1 – Distribution of LLM usages in security domains [10]

VII.1.2.1 Pratiques actuelles de développement par LLM : exemple d'IntelliCode

VII.2 Correction et complétion pendant la phase de développement : promesses et difficultés rencontrées

VII.3 Interprétation des métriques de classification présentées

Chapitre VIII

Études de cas et intégration dans un IDE

VIII.1 Résultats préliminaires avec un déploiement des modèles sur VSCode

VIII.1.1 Méthodologie inhérente au déploiement

VIII.1.2 Résultats obtenus

VIII.1.3 Cas de figure non ou partiellement couverts par l'étude

Chapitre IX

Conséquences pour l'industrie et la recherche en cybersécurité :

IX.1

IX.1.1

IX.1.1.1

IX.2

IX.3 Continuité de la recherche

IX.3.1 Sélection et préparation des données : des méthodes encourageantes

Quatrième partie

Analyse critique et perspectives

Chapitre X

Problèmes éthiques et limites des modèles d'IA

X.1

X.1.1

X.1.1.1

Chapitre XI

Biais, responsabilité et risques d'utilisation malveillante

XI.1

XI.1.1

XI.1.1.1

Chapitre XII

Suggestions d'améliorations et directions futures

XII.1

XII.1.1

XII.1.1.1

Conclusion

Annexe 1 :

Annexe 2 :

Cinquième partie

Bibliographie

Bibliographie

[1] Aaron Chan et al. « Transformer-based Vulnerability Detection in Code at EditTime : Zero-shot, Few-shot, or Fine-tuning ? » In : (mai 2023).

[2] Mark Chen et al. « Evaluating Large Language Models Trained on Code ». In : (nov. 7). URL : https://arxiv.org/pdf/2107.03374.

[3] *GitHub Copilot : Your AI pair programmer*. Mars 2025. URL : https://github.com/features/copilot.

[4] Xinyi Hou et al. « Large Language Models for Software Engineering : A Systematic Literature Review ». In : (déc. 2024). URL : https://arxiv.org/pdf/2308.10620.

[5] Hammond Pearce et al. « Asleep at the Keyboard ? Assessing the Security of GitHub Copilot's Code Contributions ». In : (nov. 2020). URL : https://arxiv.org/pdf/2108.09293.

[6] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, nov. 2024. URL : https://udlbook.github.io/udlbook/.

[7] Alexey Svyatkovskiy et al. « IntelliCode Compose : Code Generation using Transformer ». In : (déc. 2021). URL : https://arxiv.org/pdf/2005.08025.

[8] *Tabnine : Industry-leading AI code assistant*. Mars 2025. URL : https://www.tabnine.com/about/.

[9] Towards an Understanding of Large Language Models in Software Engineering Tasks. « Zibin Zheng and Kaiwen Ning and Qingyuan Zhong and Jiachi Chen and Wenqing Chen and Lianghong Guo and Weicheng Wang and Yanlin Wang ». In : (déc. 2024). URL : https://arxiv.org/pdf/2308.11396.

[10] Hanxiang Xu et al. « Large Language Models for Cyber Security : A Systematic Literature Review ». In : (juill. 2024). URL : https://arxiv.org/pdf/2405.04760.

[11] Zibin Zheng et al. « A Survey of Large Language Models for Code : Evolution, Benchmarking, and Future Trends ». In : (jan. 2024). URL : https://arxiv.org/pdf/2311.10372.