Cognitive-Aware Plugin for Vulnerability Feedback

Anonymous Author(s)

Abstract

Software vulnerability exploitation remains a issue, as studied by various governmental, academic, and industrial institutions. Secure coding education is an important process for reducing software vulnerabilities. Various approaches have been studied, many with positive success, though students still tend to graduate with security knowledge deficiencies. Previous work has been done to study how large language models, attention/cognitive predictors, and code analyzers independently impact education environments. We propose the creation of a Cognitive-Aware Visual Studio Code Plugin for Vulnerability Feedback (CAPV). This system would use a static code analysis tool to analyze code for software vulnerabilities, cognitive data collected from eye-trackers and webcams, and large language models to create effective, context-aware vulnerability feedback for programmers. The cognitive data could also be used to inform vulnerability mitigation strategies. We plan to develop this plugin and evaluate its ability to reduce software vulnerabilities and improve secure coding abilities and knowledge.

CCS Concepts

• **Applied computing** → **Interactive learning environments**; **Computer-managed instruction**; • **Security and privacy** → *Software security engineering*; • **Computing methodologies** → **Machine learning**.

Keywords

Secure Coding Education, Software Vulnerability Feedback, Adaptive Learning Tools, Machine Learning

ACM Reference Format:

Anonymous Author(s). 2025. Cognitive-Aware Plugin for Vulnerability Feedback. In *Proceedings of (SIGCITE '25).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Description

This lightning talk describes a brief overview of secure coding education approaches, the use of large language models for feedback generation, the study of student attention and comprehension detection, and how these could possibly be combined for effective software vulnerability feedback.

Software vulnerabilities exploitation continues to be an issue. Academic [9, 16, 22], industrial [8, 19], and government [3] findings support this finding and support the idea that programmers need to be trained in secure coding. Computing students, for example, tend to write increasingly insecure code as they advance through the computing curriculum [16] and tend to graduate with security knowledge deficiencies [11]. Secure coding education is a broad research area with many approaches, such as adjusting academic computing curricula to include aspects of secure coding [4, 6? , 7] to courses and creating educational secure coding tools [1, 10].

Recently, the development of Large Language Models (LLMs) has led to it be used in the education setting. It has been used to generate tailored feedback in education settings [12] and even for software vulnerability detection [18]. The direct use of AI tools by students has some notable issues, however. One issue is that direct AI use by students shows weak correlation with achieving intended learning outcomes [13, 14]. Another issue is that student tend to struggle in creating effective prompts that elicit educationally valuable responses [2].

Work has been done to study the attention and cognition patterns of students in classrooms from multi-modal data. Work from authors and others have previously studied how student attentiveness and comprehension could be predicted using webcams and/or eye-trackers [5, 15, 21]. While there has been previous work looking at creating adaptive micro-learning systems for managing cognitive load [23], there does not appear to be a system designed to incorporate cognitive and attentiveness data for creating personalized feedback. Additionally, using cognitive and attentiveness data in combination with LLMs for generating feedback appears to be underexplored.

2 Cognitive-Aware Plugin for Vulnerability Feedback

Our proposed idea explores how cognitive and attention data can be used in combination with software vulnerability detection data and large language models to create effective feedback for programmers. We plan to utilize machine learning models trained on face images and eye-tracking data to infer real-time attentional and cognitive states of the programmer. It would use a static analysis tool for grounded software vulnerability analysis. We plan for this system to be an extension for Visual Studio Code. The title of the tool will be *Cognitive-Aware Plugin for Vulnerability Feedback* or *CAPV*.

The cognitive-aware IDE plugin will work by continuously and unobtrusively collecting multi-modal attention metrics, including gaze pattern data (fixation duration, saccade amplitude, scan-path complexity, etc.) and facial attention indicators. The metrics are processed in real-time to infer the coder's current cognitive state, predict the potential introduction of software vulnerabilities, and to predict the most efficient methods for remedying existing vulnerabilities. The logic of the system will be informed by attention and behavior strategies of experts, machine learning models, analysis from static code analysis tools, and real-time attentiveness levels to provide immediate, targeted feedback and intervention for vulnerabilities as the programmer codes. LLMs will be used for dynamic, personalized explanations and remediation strategies tailored to the attentional state and coding behaviors of users. The LLMs will also be used to filter the outputs from the vulnerability analysis as to not overwhelm the programmer with irrelevant information.

The proposed system is built upon our previous development work integrating SonarQube static analysis with tailored, LLM-generated vulnerability feedback through [*[Anonymized]*]. With [*[Anonymized]*], students submit their assignment code to be evaluated for vulnerabilities. SonarQube identifies vulnerabilities within the code, and ChatGPT generates detailed, actionable feedback regarding these vulnerabilities, guiding students on potential remediation strategies tailored to their submitted code. This existing infrastructure provides a basis for the cognitive-aware feedback component. The work also builds off our previous work involving classroom attention tracking, comprehension analysis, and analysis of student-produced software vulnerabilities.

Our proposed plugin is grounded in Cognitive Load Theory [17] and the Zone of Proximal Development [20]. By minimizing extraneous detail from static analysis outputs and using cognitive data to inform feedback, CAPV could reduce cognitive overload and provide tailored scaffolding based on student expertise. This ensures feedback remains both accessible and educationally effective.

The anticipated outcomes of implementing and evaluating CAPV include reductions in vulnerability introduction rates, improved efficiency in vulnerability detection and remediation, and heightened user awareness and adoption of secure coding practices.

We currently have a preliminary prototype developed that collects gaze data from a monitor-mounted Tobii 4C Eye Tracker, face images from a 1080p webcam, code file content and changes, and software vulnerability analysis data from a locally-ran SonarQube Community Edition instance. The instance is run as a sub-process within the VS Code plugin itself, and the eye-tracker and face image data is collected as a C++ node add-on API sub-extension.

3 Lightning Talk Plans

We plan for the audience of the talk to be geared towards secure coding education researchers and cybersecurity educators. For the talk, we plan to present it as a slideshow presentation with open discussion. We plan to briefly discuss the background research on secure coding education approaches, the motivation for a cognitive-aware approach, and our in-progress work in developing a plugin. We plan on inviting discussion for our proposed idea, and other potential uses for cognitive data and LLMs for secure coding education.

References

[1] Hongmei Chi, Edward L. Jones, and John Brown. 2013. Teaching Secure Coding Practices to STEM Students. In *Proceedings of the 2013 on InfoSecCD '13: Information Security Curriculum Development Conference (InfoSecCD '13)*. Association for Computing Machinery, New York, NY, USA, 42–48. doi:10.1145/2528908.2528911

[2] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*. ACM, 1136–1142.

[3] Department of Homeland Security, US-CERT. [n. d.]. Software Assurance. https://www.cisa.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf

[4] Dieter Pawelczak. 2020. Teaching Security in Introductory C-Programming Courses. *6th International Conference on Higher Education Advances (HEAd'20)* (June 2020). doi:10.4995/HEAd20.2020.11114

[5] Mohamed Elbawab and Roberto Henriques. 2023. Machine Learning applied to student attentiveness detection: Using emotional and non-emotional measures. *Education and Information Technologies* (2023), 1 – 21. https://api.semanticscholar.org/CorpusID:258428677

[6] T. Gasiba, U. Lechner, Jorge Cuéllar, and A. Zouitni. 2020. Ranking Secure Coding Guidelines for Software Developer Awareness Training in the Industry. https://www.semanticscholar.org/paper/Ranking-Secure-Coding-Guidelines-for-Software-in-Gasiba-Lechner/6712c7daa2a614cf0ad7c89601033f1d4a45dd02

[7] Mark Harris and Karen Patten. 2015. Using Bloom's and Webb's Taxonomies to Integrate Emerging Cybersecurity Topics into a Computic Curriculum. *Journal of Information Systems Education* 26, 3 (Jan. 2015), 219–234. https://aisel.aisnet.org/jise/vol26/iss3/4

[8] John Zorabedian. [n. d.]. Veracode Survey Research Identifies Cybersecurity Skills Gap Causes and Cures. https://www.veracode.com/blog/security-news/veracode-survey-research-identifies-cybersecurity-skills-gap-causes-and-cures

[9] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery, New York, NY, USA.

[10] Jun Zhu, Heather Richter Lipford, and Bill Chu. 2013. Interactive Support for Secure Programming Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 687–692. doi:10.1145/2445196.2445396

[11] Jessica Lam, Elias Fang, Majed Almansoori, Rahul Chatterjee, and Adalbert Gerald Soosai Raj. 2022. Identifying Gaps in the Secure Programming Knowledge and Skills of Students. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1 (SIGCSE 2022, Vol. 1)*. Association for Computing Machinery, New York, NY, USA, 703–709. doi:10.1145/3478431.3499391

[12] Mengqi Liu and Faten M'Hiri. 2024. Beyond Traditional Teaching: Large Language Models as Simulated Teaching Assistants in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 743–749. doi:10.1145/3626252.3630789

[13] Stephen MacNeil, Andrew Tran, Juho Leinonen, Paul Denny, Joanne Kim, Seth Bernstein, Brett Steele, and Sami Sarsa. 2023. Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*. ACM, 931–937.

[14] James Prather, Brent N Reeves, Paul Denny, Brett A Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 2023 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 10–16.

[15] Andrew Sanders, Bradley Boswell, Gursimran Singh Walia, and Andrew Allen. 2021. Non-Intrusive Classroom Attention Tracking System (NiCATS). In *2021 IEEE Frontiers in Education Conference (FIE)*. 1–9. doi:10.1109/FIE49875.2021.9637411

[16] Andrew Sanders, Gursimran Singh Walia, and Andrew Allen. 2024. Analysis of Software Vulnerabilities Introduced in Programming Submissions Across Curriculum at Two Higher Education Institutions. In *2024 IEEE Frontiers in Education Conference (FIE)*.

[17] John Sweller, Jeroen JG van Merriënboer, and Fred Paas. 2019. Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review* 31, 2 (2019), 261–292.

[18] Karl Tamberg and Hayretdin Bahsi. 2024. Harnessing Large Language Models for Software Vulnerability Detection: A Comprehensive Benchmarking Study. *ArXiv* (2024). doi:10.48550/ARXIV.2405.15614

[19] Verizon. [n. d.]. Verizon 2023 Data Breach Investigations Report. https://www.verizon.com/business/resources/reports/dbir/

[20] L. Vygotsky. 1978. *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, Cambridge.

[21] Nan Xie, Zhaojie Liu, Zhengxu Li, Wei Pang, and Beier Lu. 2023. Student engagement detection in online environment using computer vision and multi-dimensional feature fusion. *Multimedia Systems* 29 (2023), 3559–3577. https://api.semanticscholar.org/CorpusID:260999785

[22] Tolga Yilmaz and Özgür Ulusoy. 2022. Understanding Security Vulnerabilities in Student Code: A Case Study in a Non-Security Course. *Journal of Systems and Software* 185 (March 2022), 111150. doi:10.1016/j.jss.2021.111150

[23] Bo Zhu, Kien Tsong Chau, and Nur Azlina Mohamed Mokmin. 2024. Optimizing cognitive load and learning adaptability with adaptive microlearning for in-service personnel. *Scientific Reports* 14 (2024). https://api.semanticscholar.org/CorpusID:273700115