Topic 1: Automated Vulnerability Assessment Tool Development

Academic Sources

Resource 1

- Citation: Shereen, E., Ristea, D., Vyas, S., McFadden, S., Dwyer, M., Hicks, C., & Mavroudis, V. (2024). SoK: On Closing the Applicability Gap in Automated Vulnerability Detection. arXiv.

- Type: Academic Article / Systematization of Knowledge (Survey)

- Synopsis: This paper systematically reviews 79 automated vulnerability detection (AVD) research articles and 17 empirical studies. It examines core components like task formulation, language support, detection approaches, datasets, and performance metrics. It highlights gaps: too much focus on narrow tasks/languages, limited real-world deployment, poor dataset quality, and issues with reproducibility. It suggests directions (e.g., better language support, realistic dataset, evaluations in live codebases).

- Link: https://arxiv.org/abs/2412.11194

- Relevance: 5/5 — This directly informs what to address when building an automated vulnerability assessment tool (tool scope, data, languages, metrics).

Resource 2

- Citation: Sezer, E. C., Kil, C., & Ning, P. (2010). Automated Software Vulnerability Analysis. In S. Jajodia, P. Liu, V. Swarup, & C. Wang (Eds.), Cyber Situational Awareness (Advances in Information Security, Vol. 46, pp. 139-159). Springer.

- Type: Academic Book Chapter

- Synopsis: This chapter provides foundational techniques and challenges for automating software vulnerability analysis. It covers static and dynamic analysis approaches, hybrid methods, and considerations like scalability and safety. Valuable in that it gives early work (though somewhat older) that helps understand core design trade-offs in tool development. Limitations: less focus on modern ML/LLM-based techniques, limited evaluation on current large codebases / Python specifics.

- Link: https://doi.org/10.1007/978-1-4419-0140-8_10

- Relevance: 4/5 — Gives deep insight into techniques which can be reused or compared when designing a tool; though somewhat older, still relevant.

Online / Multimedia Sources

Resource 3

- Citation: ISACA. (n.d.). Vulnerability Assessment: Identifying Security Weaknesses in Your Enterprise [White paper]. ISACA. Retrieved from https://www.isaca.org/resources/white-papers/vulnerability-assessment

- Type: White Paper

- Synopsis: This white paper describes what vulnerability assessments are, the role they play in a security program, best practices, tool usage, and how they fit into enterprise risk and audit. It helps define functional and non-functional requirements (e.g. scope, frequency, reporting) for building or choosing tools. Limitations: more strategic / high-level; not deeply technical implementation details.

- Link: https://www.isaca.org/resources/white-papers/vulnerability-assessment

- Relevance: 4/5 — Good for defining use-cases, requirements, and enterprise alignment for automated assessment tools.

Resource 4

- Citation: Cigniti Technologies. (2025). Security Testing and Vulnerability Assessment Tools – Learnings and Experiences [White paper]. Cigniti. Retrieved from https://www.cigniti.com/resource/white-papers/security-testing-tools-experiences-recommendations/

- Type: White Paper / Industry Report

- Synopsis: This document provides practical insights into security testing and vulnerability assessment tools: types, where they work well, common failures, how organizations combine tools, and what features seem to matter in real deployments. Useful lessons for tool developers on usability, false positives, integration, and context. Limitations: vendor-bias potential; some examples specific to Cigniti's tools or clients.

- Link: https://www.cigniti.com/resource/white-papers/security-testing-tools-experiences-recommendations/

- Relevance: 5/5 — Very relevant for the practical side: what works, what doesn't, what users expect.

Topic 2: Secure Code Review: Common Vulnerability in Python Applications

Academic Sources

Resource 5

- Citation: Wartschinski, L., Noller, Y., Vogel, T., Kehrer, T., & Grunske, L. (2022). VUDENC: Vulnerability Detection with Deep Learning on a Natural Codebase for Python. arXiv.

- Type: Academic Article / Conference/Pre-print

- Synopsis: Builds a deep learning model (with word2vec + LSTM) to detect vulnerabilities in Python code. Trained on over 1,000 vulnerability-fixing commits across several vulnerability types (SQLi, XSS, etc.). Offers good recall/precision, points out likely vulnerable code fragments. Value: matches your topic well; shows what types of vulnerabilities can be caught automatically in Python. Limitations: may miss nuanced context or logic vulnerabilities; may have false positives; trained on certain repository types—generalizability could be limited.

- Link: https://arxiv.org/abs/2201.08441

- Relevance: 5/5 — One of the most directly relevant to Python and vulnerability detection.

Resource 6

- Citation: Charoenwet, W., Thongtanunam, P., Pham, V.-T., & Treude, C. (2023). Toward Effective Secure Code Reviews: An Empirical Study of Security-Related Coding Weaknesses. arXiv.

- Type: Academic Article / Empirical Study

- Synopsis: This paper analyses a very large set of code review comments from two large open source projects (OpenSSL & PHP) to see which coding weaknesses are raised during code reviews; how often they get fixed, how many are acknowledged but not fixed, etc. While not specific to Python, many findings about what types of coding weaknesses slip through review are general (e.g., issues of API misuse, authentication, privilege, cryptography) and useful in understanding how Python applications might suffer. Limitations: Not Python-specific; projects are OpenSSL/PHP not Python; coding weaknesses taxonomy is general so might miss

Python-specific pitfalls.

- Link: https://arxiv.org/abs/2311.16396

- Relevance: 4/5 — Very helpful for understanding what kind of vulnerabilities are often missed in code reviews / what reviewers tend to catch.

Online / Multimedia Sources

Resource 7

- Citation: Mather, D., & Karas, B. (n.d.). Secure Coding Guide for Python – David Mather & Bart Karas, Ericsson [Conference talk / video]. OpenSSF. Retrieved from Class Central.

- Type: Educational Video / Conference Talk

- Synopsis: This is a talk that walks through the OpenSSF guide for secure coding in Python: tip-by-tip, structured around the CWE catalogue. Covers best practices, example vulnerabilities, advice on tool-based detection, and the current state of the guide + improvements. Very useful for understanding what to look for in Python code reviews. Limitations: At times high-level; not always deep code samples; might not cover every kind of Python framework.

- Link: via Class Central: https://www.classcentral.com/course/youtube-secure-coding-guide-for-python-david-mather-bart-karas-ericsson-336107

- Relevance: 5/5 — Directly maps to "common vulnerabilities in Python" and code review practices.

Resource 8

- Citation: "Securing Code with the Python Type System" (2022). Graham Bleaney & Pradeep Kumar Srinivasan. PyCon US talk. Retrieved from PyVideo.

- Type: Conference Talk / Video

- Synopsis: This talk shows how Python's type system (type annotations, stricter typing, runtime validation) can help with exactly those vulnerabilities that often arise from misuse of input, untrusted data, or logic errors. Covers how tools like CodeQL and Pysa use type and taint analysis, and discusses what type systems cannot do. Offers concrete code examples. Limitations: Doesn't catch all vulnerabilities (e.g. logic bugs not visible via types), depends on tool support.

- Link: https://pyvideo.org/pycon-us-2022/securing-code-with-the-python-type-system.html

- Relevance: 5/5 — Highly useful for understanding how to augment code review with static analysis / typing features to catch Python vulnerabilities.