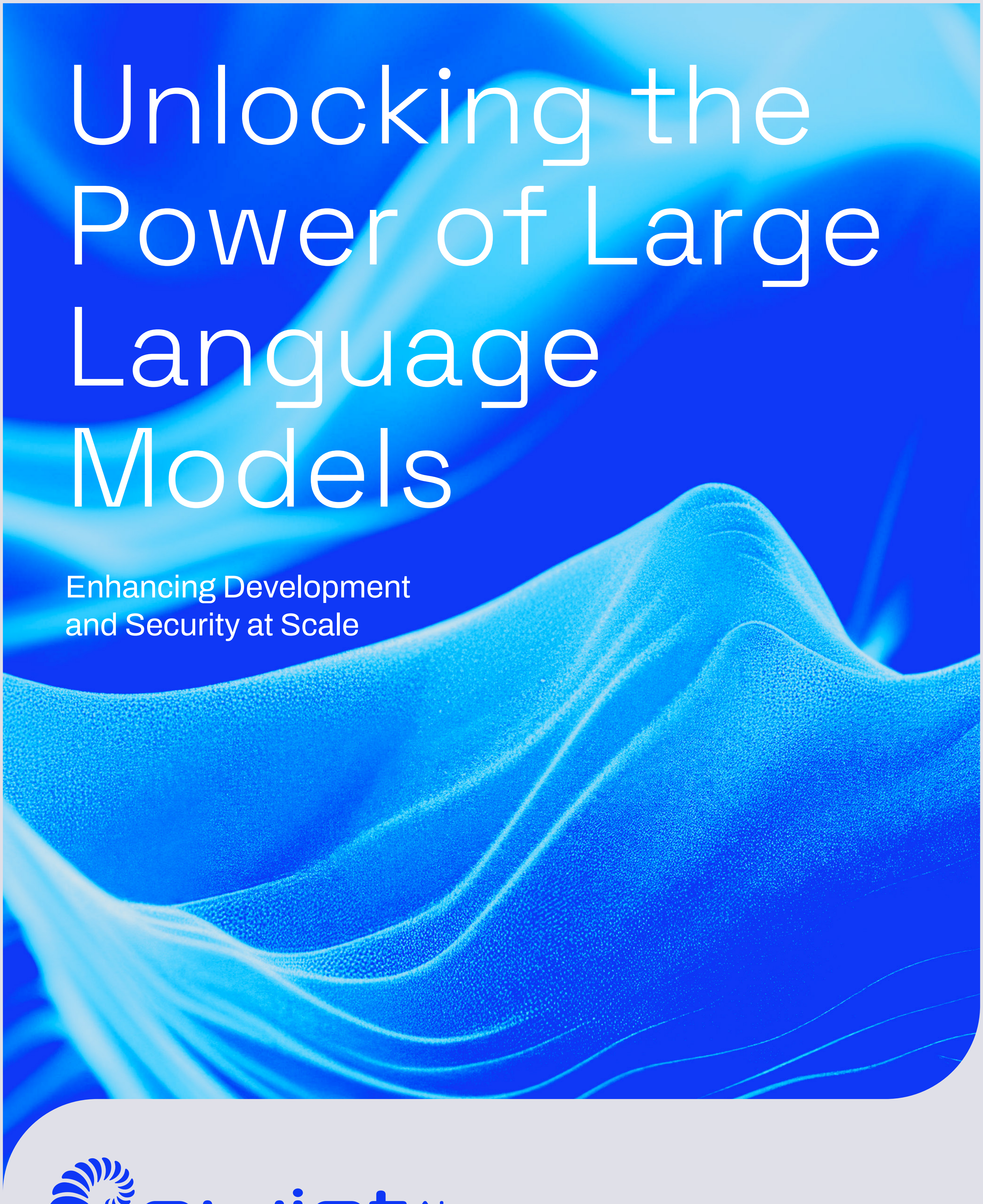
Unlocking the Power of Large Language Models

Enhancing Development and Security at Scale

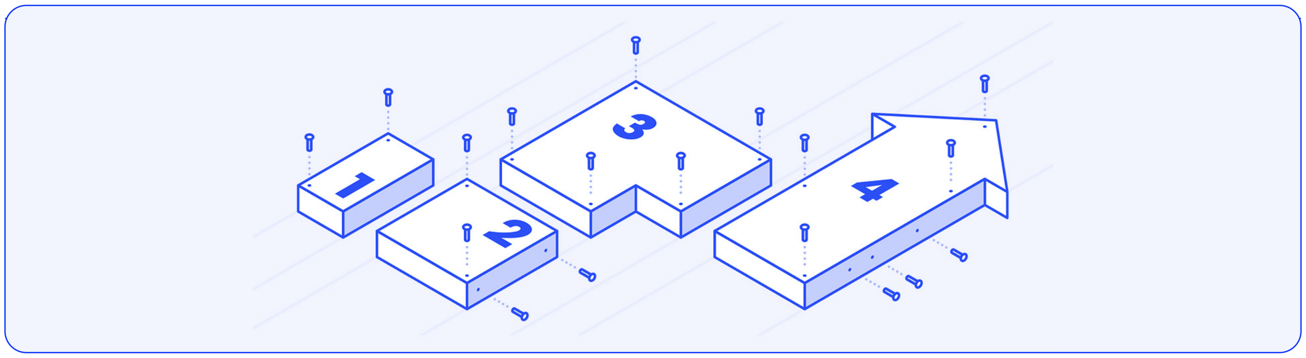


qwiet AI

LLMs guide developers by automating processes and accelerating iteration. However, blind trust in generated code introduces risks. This highlights the advantage of agentic AI, which independently evaluates, validates, and iteratively improves code, moving beyond passive reliance toward more reliable and secure software development.

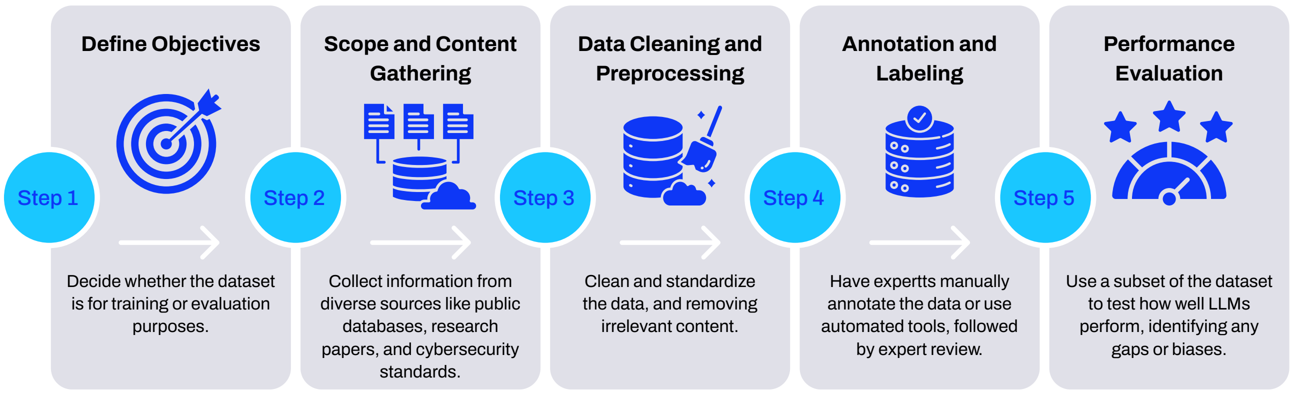
Introduction

Imagine building furniture with an instruction manual that guides you and empowers you to be efficient and precise at every step. LLMs function as such guides for programmers, giving them the power to automate processes, make fixes more efficient, and speed up iteration cycles. As these models mature, their impact goes beyond mere automation, empowering developers to write, test, and protect code in new ways, instilling a sense of control and confidence in their work.



The Intersection of Code and LLMs: Why LLMs and Code Work Together

LLMs and programming systems naturally complement each other, providing developers with advanced linguistic and generative tools to enhance coding workflows. Developer-centric LLMs like GitHub Copilot and OpenAI Codex specifically enable programmers to generate context-aware code snippets, functions, or entire modules, thus significantly boosting productivity and enhancing code quality. For example, GitHub Copilot significantly reduces manual coding by automatically generating functional code. OpenAI Codex similarly offers tailored code completions, streamlining development and facilitating rapid coding processes. DeepMind's AlphaCode further supports iterative testing through dynamic feedback loops, continually refining code accuracy and execution efficiency. Through iterative development cycles and intelligent error handling, these advanced LLM tools help developers achieve faster iteration, more accurate results, and a smoother, more intuitive coding experience.



Define Objectives

Step 1

Decide whether the dataset is for training or evaluation purposes.

Scope and Content Gathering

Step 2

Collect information from diverse sources like public databases, research papers, and cybersecurity standards.

Data Cleaning and Preprocessing

Step 3

Clean and standardize the data, and removing irrelevant content.

Annotation and Labeling

Step 4

Have expertts manually annotate the data or use automated tools, followed by expert review.

Performance Evaluation

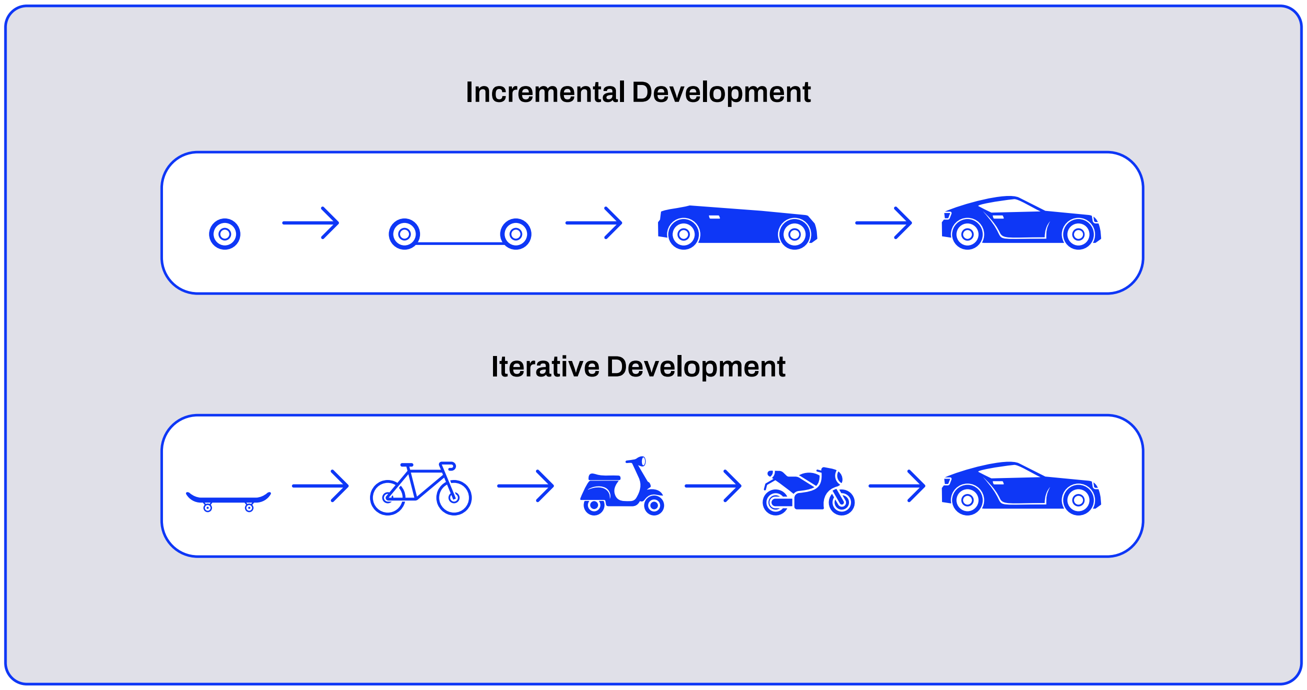
Step 5

Use a subset of the dataset to test how well LLMs perform, identifying any gaps or biases.

DeepMind's AlphaCode has also demonstrated how solutions generated using AI can function in competitive programming (DeepMind), such as iterative testing required in standard development environments. LLMs maximize these processes with feedback loops, refining code accuracy and execution efficiency. With the addition of iterative cycles, models dynamically adapt, enhancing overall performance and workload stability.

The Role of Iterative Gateways in Code Quality

One of the most significant aspects of LLM-based automation is its role in enhancing code quality and security. These are sequential checkpoints that ensure code quality and security at different points in the development cycle. They play a crucial role in maintaining the integrity of the code, ensuring that Generative AI-written code is thoroughly audited by the LLM before deployment, and reducing automated decision-making risks to a minimum. This process is a testament to the flexibility of code. It can be rapidly created, tested, and refined without compromising core integrity. LLMs expand on this process by predicting possible solutions and providing direction, ensuring that development cycles remain efficient and streamlined based on syntax, thereby reassuring the developers about the quality and security of their work.

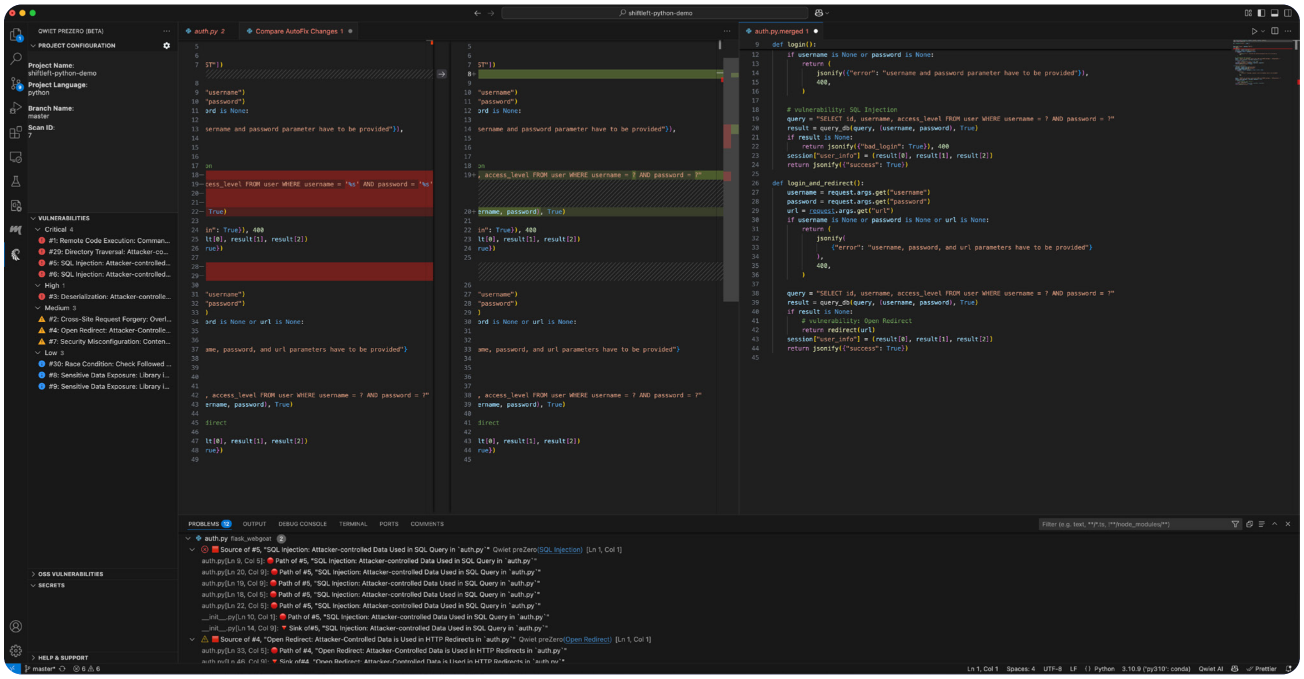


Practical Use Cases

LLMs in Integrated Development Environments (IDEs) like VSCode

LLMs assist developers directly within IDEs by offering real-time security guidance. Features include:

- Detecting security vulnerabilities as developers write code.
- Suggesting secure coding practices to prevent common exploits.
- Automatically recommending fixes for detected issues.
- Enhancing developer workflows with intelligent debugging support.


QWIET PREZERO (BETA)

PROJECT CONFIGURATION

Project Name: shiftleft-python-demo
Project Language: python
Branch Name: master
Scan ID: 7

VULNERABILITIES
- Critical 4
  - #1: Remote Code Execution: Comman...
  - #29: Directory Traversal: Attacker-co...
  - #5: SQL Injection: Attacker-controlled...
  - #6: SQL Injection: Attacker-controlled...
- High 1
  - #3: Deserialization: Attacker-controlle...
- Medium 3
  - #2: Cross-Site Request Forgery: Overl...
  - #4: Open Redirect: Attacker-Controlle...
  - #7: Security Misconfiguration: Conten...
- Low 3
  - #30: Race Condition: Check Followed ...
  - #8: Sensitive Data Exposure: Library i...
  - #9: Sensitive Data Exposure: Library i...

auth.py.merged 1

```python
def login():
    if username is None or password is None:
        return (
            jsonify({"error": "username and password parameter have to be provided"}),
            400,
        )

    # vulnerability: SQL Injection
    query = "SELECT id, username, access_level FROM user WHERE username = ? AND password = ?"
    result = query_db(query, (username, password), True)
    if result is None:
        return jsonify({"bad_login": True}), 400
    session["user_info"] = (result[0], result[1], result[2])
    return jsonify({"success": True})

def login_and_redirect():
    username = request.args.get("username")
    password = request.args.get("password")
    url = request.args.get("url")
    if username is None or password is None or url is None:
        return (
            jsonify(
                {"error": "username, password, and url parameters have to be provided"}
            ),
            400,
        )

    query = "SELECT id, username, access_level FROM user WHERE username = ? AND password = ?"
    result = query_db(query, (username, password), True)
    if result is None:
        # vulnerability: Open Redirect
        return redirect(url)
    session["user_info"] = (result[0], result[1], result[2])
    return jsonify({"success": True})
```

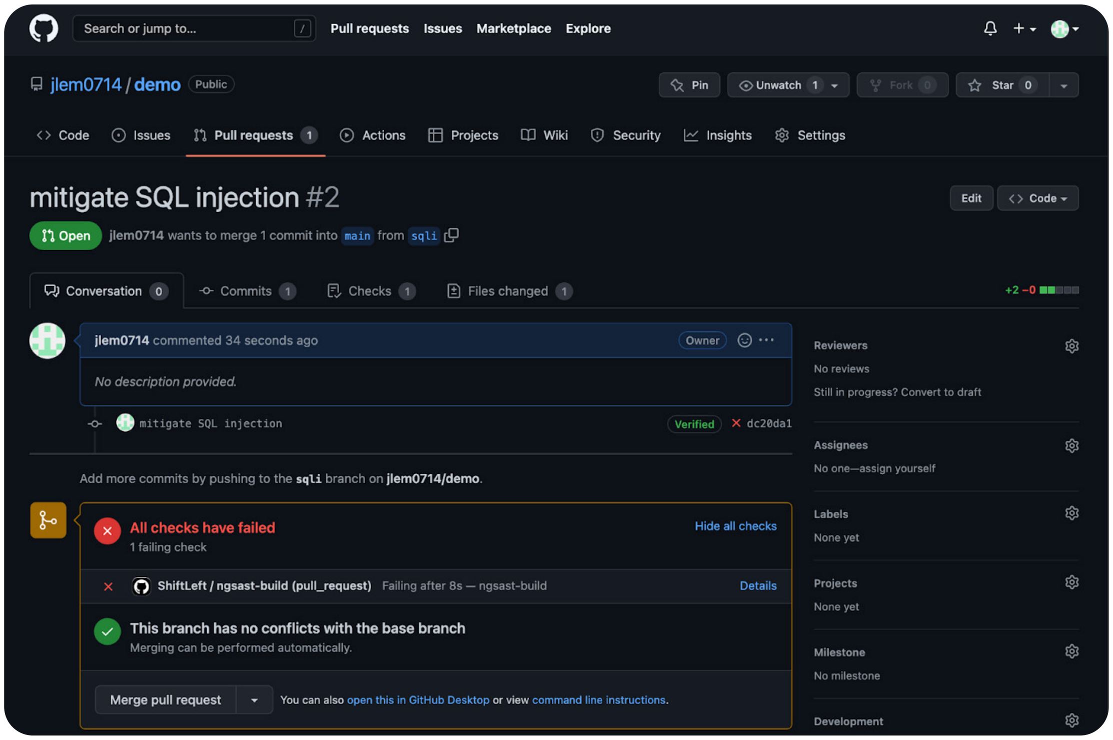
PROBLEMS 12

```
auth.py flask_webgoat 2
  Source of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`" Qwiet preZero(SQL Injection) [Ln 1, Col 1]
    auth.py[Ln 9, Col 5]: Path of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
    auth.py[Ln 20, Col 9]: Path of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
    auth.py[Ln 19, Col 9]: Path of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
    auth.py[Ln 18, Col 5]: Path of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
    auth.py[Ln 22, Col 5]: Path of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
    __init__.py[Ln 10, Col 1]: Path of #5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
    __init__.py[Ln 14, Col 9]: Sink of#5, "SQL Injection: Attacker-controlled Data Used in SQL Query in `auth.py`"
  Source of #4, "Open Redirect: Attacker-Controlled Data is Used in HTTP Redirects in `auth.py`" Qwiet preZero(Open Redirect) [Ln 1, Col 1]
    auth.py[Ln 33, Col 5]: Path of #4, "Open Redirect: Attacker-Controlled Data is Used in HTTP Redirects in `auth.py`"
    auth.py[Ln 46, Col 9]: Sink of#4, "Open Redirect: Attacker-Controlled Data is Used in HTTP Redirects in `auth.py`"
```

Developer Support Tools

Automated Generative AI enhances development processes by:

- Providing intelligent suggestions for debugging and fixing code.
- Enabling continuous iteration through predictive and generative capabilities.
- Assisting security teams by reducing manual workloads and improving vulnerability detection.



mitigate SQL injection #2

Open jlem0714 wants to merge 1 commit into `main` from `sqli`

Conversation 0 · Commits 1 · Checks 1 · Files changed 1 · +2 −0

jlem0714 commented 34 seconds ago · Owner

No description provided.

`mitigate SQL injection` · Verified · dc20da1

Add more commits by pushing to the **sqli** branch on **jlem0714/demo**.

All checks have failed
1 failing check · Hide all checks

ShiftLeft / ngsast-build (pull_request) Failing after 8s — ngsast-build · Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request · You can also open this in GitHub Desktop or view command line instructions.

Reviewers: No reviews
Still in progress? Convert to draft

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development

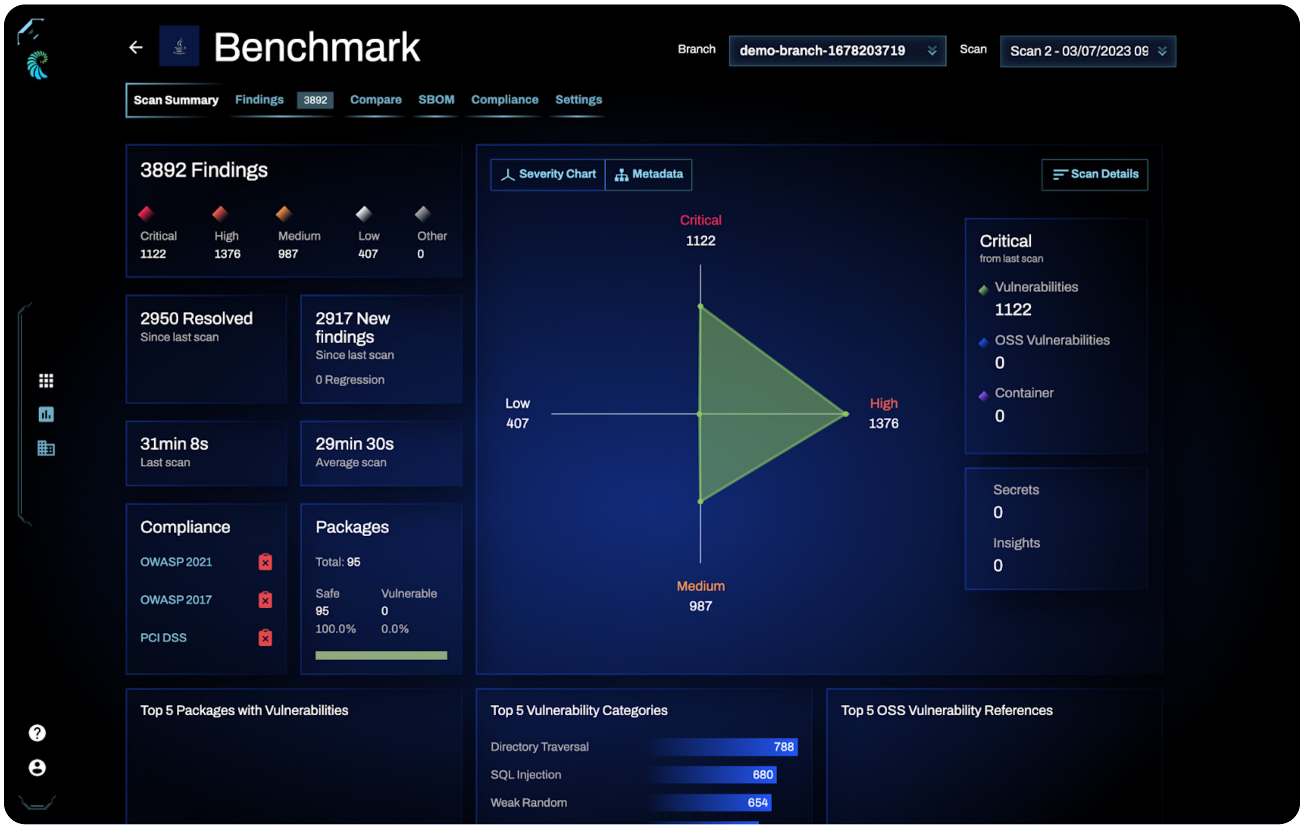
Application Security Enhancements

LLMs play a crucial role in application security by:

- Running scans, analyzing vulnerabilities, and automating remediation of CVEs (Microsoft Security Blog).
- Enhancing the validation of security fixes through pattern recognition.
- Providing real-time security recommendations within IDEs (OWASP Machine Learning Security Guide).
- Optimizing organization-wide security protocols and compliance adherence by making security more accessible.

Scaling with LLMs: Bending the Vulnerability Curve

The goal of integrating LLMs into application security involves bending the vulnerability curve and using automation and Generative AI-driven improvements to reduce the rate at which security flaws build up over time. LLMs can help organizations minimize technical debt and improve security by proactively identifying vulnerabilities, providing a sense of security and protection. This automation foundation frees developers to solve more complex tasks. It fosters a 'shift everywhere' mindset, where automation and AI work collaboratively to minimize dependence on manual interventions and allow for scaling self-regulating ecosystems. In this context, 'shift everywhere' means that automation and AI are not limited to specific tasks. These tasks are integrated into every aspect of the development process, from code writing to security monitoring.



Branch: demo-branch-1678203719
Scan: Scan 2 - 03/07/2023 09
Findings 3892

3892 Findings

Critical	High	Medium	Low	Other
1122	1376	987	407	0

2950 Resolved
Since last scan

2917 New findings
Since last scan
0 Regression

31min 8s
Last scan

29min 30s
Average scan

Compliance
OWASP 2021
OWASP 2017
PCI DSS

Packages
Total: 95

Safe	Vulnerable
95	0
100.0%	0.0%

Critical 1122
High 1376
Medium 987
Low 407

Critical
from last scan
Vulnerabilities
1122
OSS Vulnerabilities
0
Container
0

Secrets
0
Insights
0

Top 5 Packages with Vulnerabilities

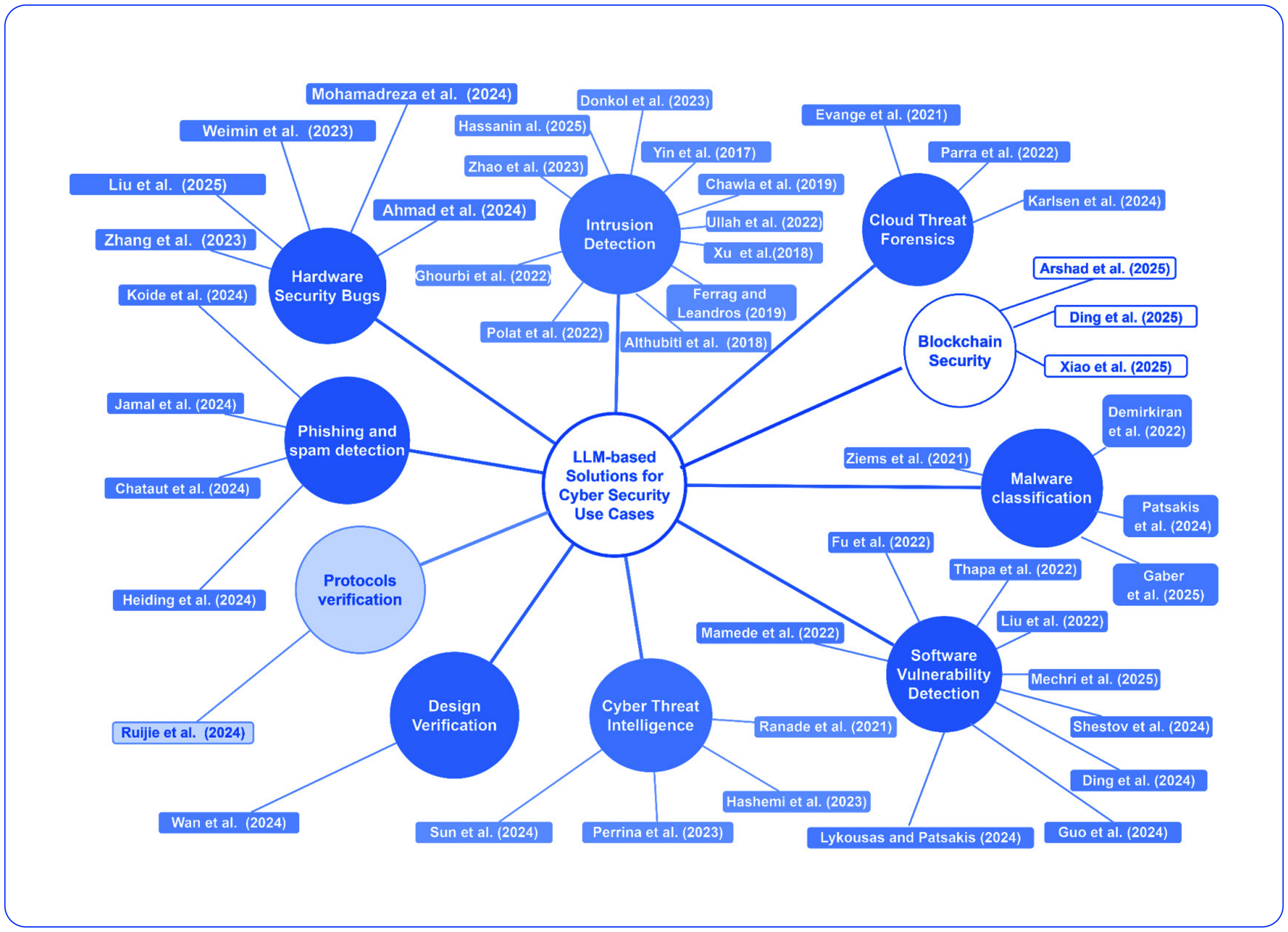
Top 5 Vulnerability Categories

Directory Traversal	788
SQL Injection	680
Weak Random	654

Top 5 OSS Vulnerability References

Future Use Cases

- **AI-Driven Documentation Centers:** LLMs can revolutionize documentation management by centralizing and automating updates. LLLs reduce reliance on manual input and ensure consistency in technical records (Stanford AI Index).
- **Auto-Fix Packets:** Organizations integrating LLMs should focus on deploying automated fix packets. These pre-packaged solutions facilitate system upgrades, reducing the complexity of debugging and maintenance (McKinsey AI in Software Development).
- **Contextual Code Updates:** LLMs can process existing code with contextual awareness, providing targeted improvements. These “hallucinations” enable nearly real-time refinements that enhance software stability and performance without extensive manual intervention.



https://arxiv.org/html/2405.12750v2

Challenges and Considerations

When implementing LLMs, precision in pattern recognition is paramount for security and efficiency. Developers must carefully balance heavy code dependencies and user-friendly automation scalability tools over time. **Over-reliance on Generative AI without human oversight can lead to unintended inefficiencies or exacerbate security risks.** To encourage Developer and Operational buy-in, it's crucial to emphasize that AI isn't here to replace jobs but to enhance productivity. By automating tasks like security, Generative AI allows developers to focus on what they love. However, implementing LLMs requires proper checks and balances to ensure they remain a reliable and secure addition to development projects. Awareness of these challenges and considerations helps organizations navigate potential issues and successfully integrate.

For troubleshooting and automated resolution, LLM-based platforms can build modular suites. These suites provide intelligent code analysis, automation in debugging, and vulnerability detection. LLM modules are designed to cater to specific needs in the development process, from code writing to security monitoring. The modularity massively increases efficiency and usability, catering to customized solutions across a host of different development needs. Such modularity ensures greater efficiency and usability, offering value through customized solutions that will address several needs related to various developments. This modularity is a key feature of LLM-based platforms, allowing them to adapt to the specific needs of different development teams and projects.

Conclusion

LLMs will finally release new efficiencies, generate innovation, and future-proof processes within organizations by embedding themselves into application and development security practices. While active management is necessary to deploy the models effectively, the benefits outweigh the negatives. However, sole reliance on a single LLM can present risks of hallucinations, decreased accuracy, and fewer checks and balances. Advancements toward agentic AI or multi-LLM architectures can mitigate these risks with more validation, dependability, and governance. Integrating LLMs as an integral part of development infrastructure ensures security, continued innovation, and resilience in the evolving tech landscape, making the audience feel secure and prepared for the future. Organizations may also refer to reports like the Gartner Hype Cycle for AI & Software Engineering to measure their approach against emerging AI adoption trends.

Watch this free webinar to learn how LLMs can improve your organization's reachability of AppDev and Security teams.

About Qwiet AI

Qwiet AI empowers security and development teams with Agentic AI-driven solutions that enhance reachability analysis, streamline vulnerability remediation, and accelerate security workflows. Our cutting-edge approach ensures that security remains accurate, efficient, and fully integrated into modern development pipelines.

For more information, visit **Qwiet.ai** and book a free consultation.