POSTED ON SEPTEMBER 30, 2025 TO AI RESEARCH, ML APPLICATIONS, SECURITY & PRIVACY

LLMs Are the Key to Mutation Testing and Better Compliance

By Mark Harman

- Following our keynote presentations at FSE 2025 and EuroSTAR 2025, we're delving further into the development of Meta's Automated Compliance Hardening (ACH) tool, an LLM-based tool for software testing that is automating aspects of compliance adherence at Meta, while accelerating developer and product velocity.
- By leveraging LLMs we've been able to overcome the barriers that have prevented mutation testing from being efficiently deployed at scale. This allows us to greatly simplify risk assessments, reduce cognitive load for developers, and, ultimately, create a safer online ecosystem by enabling continuous compliance.
- We're also inviting the community to join us in exploring new challenges and opportunities for leveraging LLMs in software testing through efforts like our Catching Just-in-Time Test (JiTTest) Challenge.

Today, AI is accelerating the pace and complexity of technology development worldwide, requiring compliance systems to keep up. However, compliance has traditionally relied on manual processes, which can be error-prone and challenging to scale.

At Meta, we've been investing in advanced AI-enabled detection mechanisms to help us ensure we're upholding our responsibility to keep our products and services safe for everyone while adhering to compliance obligations at scale. AI-powered solutions help our engineers, developers, and product teams meet global regulatory requirements more easily and efficiently so they can spend more time focusing on building new and innovative products and services.

Earlier this year, we released new research into leveraging large language models (LLMs) for mutation-guided test generation – where faults (mutants) are deliberately introduced into source code as a method of assessing how well a testing framework can detect those faults.

Meta's Automated Compliance Hardening (ACH) tool successfully combines automated test generation techniques with the capabilities of LLMs to generate highly relevant mutants for testing as well as tests that are guaranteed to catch those mutants. Through simple, plain-text prompts where engineers describe the mutant to test, ACH makes this process intuitive and reliable. It's one of our latest AI-powered detection mechanisms that helps us safeguard our operations and catch code that is out of compliance. With ACH we can more easily and proactively identify bugs that would negatively impact our compliance, and prevent them from entering our systems in the future. This technology provides Meta engineers and our product teams with the consistency and confidence they need to ensure our codebase remains risk-resilient.

Since empowering ACH with our research findings, we've presented our work at keynote presentations at FSE 2025 and EuroSTAR 2025. Our presentations shared insights into how we've used LLMs to solve the major barriers that have prevented mutation testing at scale and highlighted new areas in automated software testing where LLMs can have a significant impact.

For a long time people thought of mutation testing as a way of assessing test quality but less as a way to generate tests. By leveraging generative AI, we've been able to make what studies have consistently shown to be the most powerful form of software testing even more efficient and scalable.

The Challenge of Scaling Mutation Testing

The idea behind mutation testing is to go beyond traditional structural coverage criteria like statement coverage or branch coverage (which only show if lines of code are run), to a more robust system of testing. Where statement or branch coverage might still fail to detect a bug if a line still runs, mutation testing reveals whether a test fails after inserting a mutation, indicating that the tests are not effectively checking the code's behavior. As an example, ACH can simulate privacy faults that would introduce compliance risk (such as messages being shared with unintended audiences) to model a potential real-world issue. It then creates unit tests to catch these bugs, preventing them from reaching production, even if they're reintroduced in future code changes.

Even though mutation testing cannot exist on its own (it requires a test to already exist), it helps engineers and developers identify weak assertions and encourages them to write tests that truly validate code behavior instead of just executing it.

In practice however, mutation testing has been notoriously difficult to deploy. Despite over five decades of research, mutation testing has traditionally faced five major barriers:

1. Mutation Testing Isn't Scalable

Traditional mutation testing generates a very large number of mutants, making it computationally expensive and difficult to scale to large industrial codebases. The sheer volume of mutants can overwhelm testing infrastructure and slow down development cycles.

2. Mutation Testing Can Create Unrealistic Mutants

Mutants generated via traditional means can be unrealistic or irrelevant to real faults that developers are interested in.

This can happen for a few reasons:

- **Rule-based mutation operators**: Traditional mutation testing relies on predefined, rule-based mutation operators that apply generic syntactic changes (e.g., flipping boolean conditions, changing arithmetic operators). These operators do not consider the specific context or domain of the code, leading to mutants that do not represent faults that developers would realistically introduce.
- **Lack of specific focus**: Mutants generated without targeting a specific class of faults or domain concerns often produce changes that are irrelevant to the actual risks or issues faced by the system.
- **Semantic irrelevance**: Some mutants may syntactically change the code but do not affect the program's semantics in a meaningful way or do not simulate realistic fault conditions. These mutants do not help in improving test quality because they do not represent faults that tests should catch.
- **Overgeneralization**: Applying broad mutation rules uniformly across all code can generate mutants that aren't useful in the context of the specific software, leading to wasted effort in trying to kill mutants that do not correspond to real-world bugs.

3. Equivalent Mutants Waste Time and Resources

Equivalent mutants – mutants that are syntactically different but semantically equivalent to the original code – have been a persistent challenge for mutation testing that wastes developer time and computational resources. Determining whether a mutant is equivalent or not is known to be mathematically undecidable, adding to the technical challenge of the problem

4. Mutation Testing Requires a Lot of Computational Resources

Mutation testing is costly in terms of computational resources and developer effort. Running tests against many mutants and analyzing results requires a significant amount of infrastructure and time, which can be prohibitive in fast-paced industrial environments.

5. Mutation Testing Can Overstretch Testing Efforts

Mutation testing can overstretch testing efforts by focusing on killing mutants that may not correspond to meaningful or high-impact faults. This can lead to diminishing returns where additional testing effort does not translate into better fault detection or software quality.

How LLMs Solve the Challenges of Mutation Testing

While it has been challenging for large organizations like Meta to deploy mutation testing at scale, what they have been able to do is collect vast amounts of data on the bugs found in various stages of their software development. All of this data can be used to train an LLM to guide test generation.

When we construct mutants that are both highly-relevant and currently not caught (unkilled) by any existing testing framework, we can use these mutants as prompts for LLM-based test generation (hence, mutation-guided, LLM-based test generation). The end result is ACH – a system and workflow that can generate both problem-specific mutants *and* the tests that can catch them, using plain text instructions.

By leveraging LLMs, ACH solves for each of the barriers to mutation testing deployment:

1. ACH Enables Scalable Mutant Testing

Meta's ACH system uses LLMs to generate fewer, more realistic, and highly specific mutants targeted at particular fault classes (e.g., privacy faults), increasing scalability and relevance. This mutation-guided approach focuses on faults relevant to the specific problem domain, which improves the relevance and quality of mutants and also resolves scalability issues by significantly lowering the number of mutants that need to be generated in order to be relevant and useful.

2. ACH Creates Realistic Mutants

With ACH, a security or privacy engineer can use textual descriptions of issues they are concerned about to generate very realistic problem-specific bugs that apply directly to an area of concern.

3. ACH Detects and Kills Equivalent Mutants With LLMs

ACH features an LLM-based Equivalence Detector agent that is often capable of judging whether a mutant is equivalent to the original code. In our own research and testing with ACH we found that when combined with simple static analysis preprocessing (e.g., stripping comments), this approach achieves high precision (0.79) and recall (0.47) – rising to 0.95 and 0.96 with simple preprocessing – in detecting equivalent mutants, efficiently filtering out unkillable mutants.

ACH also automatically generates unit tests that kill the mutants, so engineers only ever need to look at tests and, if they wish, mutants that are **guaranteed** to be non-equivalent.

4. Tests Generated by ACH Are Computationally Efficient and Easier To Deploy

From October to December 2024, we ran a trial where ACH was deployed for privacy testing use cases on several platforms at Meta, including Facebook, Instagram, WhatsApp, and our wearables platforms (Quest and Ray-Ban Meta glasses). Over thousands of mutants and hundreds of generated tests, privacy engineers at Meta accepted 73% of the generated tests, with 36% judged as privacy relevant. Feedback showed engineers found tests useful even when they weren't directly relevant to privacy. Our engineers appreciate the additional safety net AI can provide and the augmentation of their skillset at scale for handling edge cases. But importantly, they valued being able to focus on evaluating tests rather than having to construct them.

5. ACH Helps Prevent Overstretching

ACH generates mutants that are closely coupled to the issue of concern and produces tests that catch faults missed by existing tests. Our empirical results show that many generated tests add coverage and catch faults that would otherwise go undetected, highlighting mutation testing's superiority over structural coverage criteria alone.

The Catching JiTTest Challenge: More Frontiers for LLMs in Software Testing

LLMs have opened up exciting new challenges and areas of exploration in the domain of automated software testing, specifically around generating hardening tests and catching tests. Hardening tests protect against future regressions by ensuring that new changes do not break existing functionality. Catching tests detect faults in new or changed functionality.

Based on our work with ACH, we believe there is even more opportunity to leverage LLMs to improve test generation. Currently, we're particularly interested in using LLMs to tackle the challenge of generating just-in-time (JiT) tests, where tests are generated for human review just in time for pull requests to catch faults before code ends up in production. What makes this particularly challenging is the Test Oracle Problem – the challenge of distinguishing the desired and correct behavior based on a given input from an incorrect behavior.

To that end we're proposing the Catching Just-in-Time Test (JiTTest) Challenge to the wider community. We want to encourage engineers and developers to build systems capable of generating tests that reveal bugs in pull requests with high precision, while also keeping humans in the loop to ensure low false positives.

A "Just-In-Time" Call to Action

Our paper, "Harden and Catch for Just-in-Time Assured LLM-Based Software Testing: Open Research Challenges," which was recently presented as a keynote at FSE 2025, shares more about the JiTTest Challenge as well as the open problems around applying LLMs to automated software testing.

LLMs and the Future of Software Testing

AI has helped us streamline and optimize our compliance and overall risk management frameworks at Meta. Processes that have historically been time consuming, error prone, and difficult to comprehensively identify potential risks, are being transformed into systems that save engineer and developer time while also enhancing compliance.

However, there is still a lot of exciting work ahead to be done for ACH and in the larger area of applying LLMs to software testing to enable continuous compliance.

While our own testing with ACH explored its uses in privacy testing and focused on Kotlin as the main language, we're currently working to expand into other domains and more languages. We're also investigating ways to leverage techniques like fine-tuning and prompt engineering to make mutant generation even more precise and relevant.

More broadly, our work with ACH, as well as the JiTTest Challenge, will focus on addressing the Test Oracle Problem – exploring ways to enable testing of existing faults with high precision while avoiding false positives.

We also cannot ignore the human element in all of this. In addition to examining ways to ensure that human reviewers are present to help prevent false positives, we should also investigate how developers are interacting with LLM-generated tests to improve their adoption and usability.

We'll be presenting more of our work in the near future, including at the upcoming Product@Scale conference. We hope you'll join us on our journey to further explore AI's potential to transform software testing and raise the bar for risk management across industries.

Share this:

Facebook | Threads | X | LinkedIn | Hacker News | Email

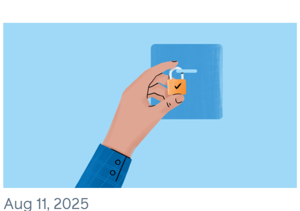
◂ Prev
Meta 3D AssetGen: Generating 3D Worlds With AI

Next ▸
Accelerating our Android apps with Baseline Profiles

Related Posts

Feb 05, 2025
Revolutionizing software testing: Introducing LLM-powered bug catchers

Jun 24, 2024
Leveraging AI for efficient incident response



Aug 11, 2025
Federation Platform and Privacy Waves: How Meta distributes compliance-related tasks at scale

Related Positions

Research Scientist, Machine Learning (PhD)
SUNNYVALE, US

Research Scientist, Machine Learning (PhD)
BELLEVUE, US

Research Scientist, Machine Learning (PhD)
REDMOND, US

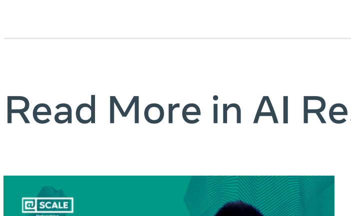
Research Scientist, Machine Learning (PhD)
MENLO PARK, US

Research Scientist, Machine Learning (PhD)
SEATTLE, US

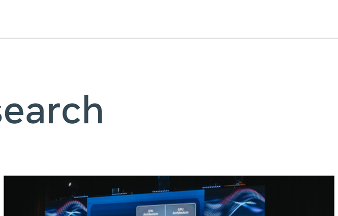
See All Jobs

Read More in AI Research

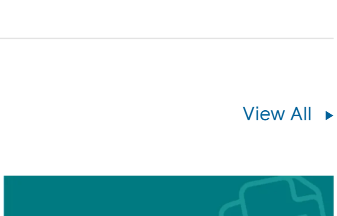
View All ▸



OCT 17, 2025
Scaling LLM Inference: Innovations in Tensor Parallelism, Context Parallelism, and Expert Parallelism

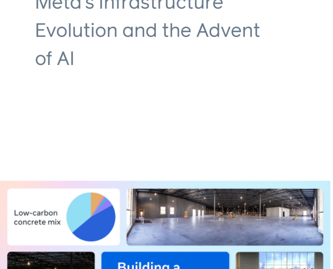


SEP 29, 2025
Meta's Infrastructure Evolution and the Advent of AI

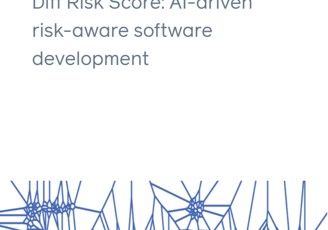


AUG 6, 2025
Diff Risk Score: AI-driven risk-aware software development

AUG 4, 2025
Building a human-computer interface for everyone



JUL 16, 2025
Using AI to make lower-carbon, faster-curing concrete



MAY 8, 2025
Accelerating GPU indexes in Faiss with NVIDIA cuVS

Available Positions

Research Scientist, Machine Learning (PhD)
SUNNYVALE, US

Research Scientist, Machine Learning (PhD)
BELLEVUE, US

Research Scientist, Machine Learning (PhD)
REDMOND, US

Research Scientist, Machine Learning (PhD)
MENLO PARK, US

Research Scientist, Machine Learning (PhD)
SEATTLE, US

See All Jobs

Technology at Meta

Engineering at Meta - X
Follow

AI at Meta
Read

Meta Quest Blog
Read

Meta for Developers
Read

Meta Bug Bounty
Learn more

RSS
Subscribe

Open Source

Meta believes in building community through open source technology. Explore our latest projects in Artificial Intelligence, Data Infrastructure, Development Tools, Front End, Languages, Platforms, Security, Virtual Reality, and more.

ANDROID | iOS | WEB | BACKEND | HARDWARE

Learn More