Regression testing: The key to ensuring software quality and reliability

BY YANNICK MERCKX · DECEMBER 12, 2024 · LAST UPDATED ON MARCH 6, 2025

Regression testing is a crucial part of software development that ensures new code changes don't negatively affect existing functionality. It comes into play for developers primarily after bug fixes or when adding new features. By running regression tests, developers can catch and fix issues early, maintaining the software's overall quality, reliability, and stability.

In this guide, you'll gain a quick understanding of regression testing, its importance, and how to boost software assurance by combining it with retesting and bug bounty programs.

What is regression testing?

Regression testing is a type of software testing that focuses on ensuring recent changes to the codebase haven't negatively affected existing functionality. It's a critical part of the software development lifecycle, especially in iterative or Agile methodologies where changes are frequent.

Imagine you have a beautifully working application, much like a finely tuned orchestra. Each instrument (or feature) plays its part perfectly. Now, you want to introduce a new instrument (or add a new feature). Before the next performance (or release), you need to make sure that the new instrument doesn't throw off the harmony of the orchestra. This is what regression testing does—it helps you catch any discordant notes (or bugs) that might have been introduced with the changes.

Regression testing involves re-running previously executed tests to compare the output with earlier results. It can be done manually, but it's often automated for efficiency, especially in large and complex projects. Automated regression tests can be run quickly and frequently, providing rapid feedback and helping to identify issues early in the development cycle.

The key benefits of regression testing include maintaining software quality, preventing the reintroduction of old bugs, and ensuring that new features don't break existing ones. It also helps in maintaining the software's stability and reliability, which is crucial for user satisfaction and trust.

In essence, regression testing is like a safety net that catches issues introduced by changes, helping developers to maintain the software's overall quality and consistency as it evolves.

Why is regression testing important?

It can be tempting to skip regression testing after completing a new development project. However, not performing regression testing can have several potential impacts, such as:

Bug reintroduction

Without regression testing, fixes for old bugs might inadvertently reintroduce those bugs or create new ones, leading to a deterioration in software quality.

Broken functionality

New features or changes might unintentionally break existing functionality. Without regression tests, these issues may go unnoticed until they're reported by end-users.

Increased technical debt

Skipping regression testing can lead to a buildup of technical debt, making the codebase harder to maintain and update in the future.

Delayed time to market

Discovering issues late in the development cycle or after release can result in delays, as the team scrambles to fix and re-release the software.

Damaged reputation

Releasing software with avoidable bugs can harm the reputation of the product and the development team, potentially leading to a loss of user trust and reduced adoption.

For example, consider a banking app that introduces a new feature without performing regression testing. If the new feature inadvertently disrupts the app's ability to process transactions, users may encounter errors when trying to transfer funds. This could lead to user frustration, negative reviews, and even financial losses for both users and the bank.

Regression testing vs retesting: What's the difference?

Regression testing and retesting are both crucial components of software testing, but they serve different purposes and are applied in different scenarios.

What is retesting?

Retesting is the process of verifying that a specific bug or defect has been fixed. When a developer addresses a bug, the tester will retest the functionality to ensure the issue is resolved. Retesting is focused and specific, targeting only the areas where defects were found and fixed. It's a straightforward process that aims to confirm that the software now behaves as expected in the previously problematic areas.

For example, if a login feature was vulnerable due to a bug, the developer would fix the issue, and the tester would then retest the login functionality to ensure it works correctly.

According to Intigriti's Ethical Hacker Insights Report 2024, a significant 95% of the security researchers are likely or very likely to retest a vulnerability they submitted if requested, indicating a high level of engagement and willingness to contribute to continuous security improvement.

Bug retesting can prevent several potential issues from slipping into the final product. Here are some key problems that bug retesting can help avoid:

· **Incomplete fixes:** Retesting ensures that a bug has been fully resolved and not just partially addressed. Without retesting, there's a risk that the fix may not completely solve the problem, leading to persistent issues.

· **Incorrect fixes:** Sometimes, developers might misunderstand the bug report and implement a fix that doesn't actually resolve the issue. Retesting helps catch these incorrect fixes and ensures that the right problem is being addressed.

· **Regressions:** Although the primary focus of retesting is the specific bug fix, it can also help identify if the fix has inadvertently introduced new issues or caused existing functionality to break.

· **Undocumented changes:** Retesting can uncover situations where a developer has made additional changes that were not requested or documented. These changes might introduce new bugs or alter the expected behavior of the software.

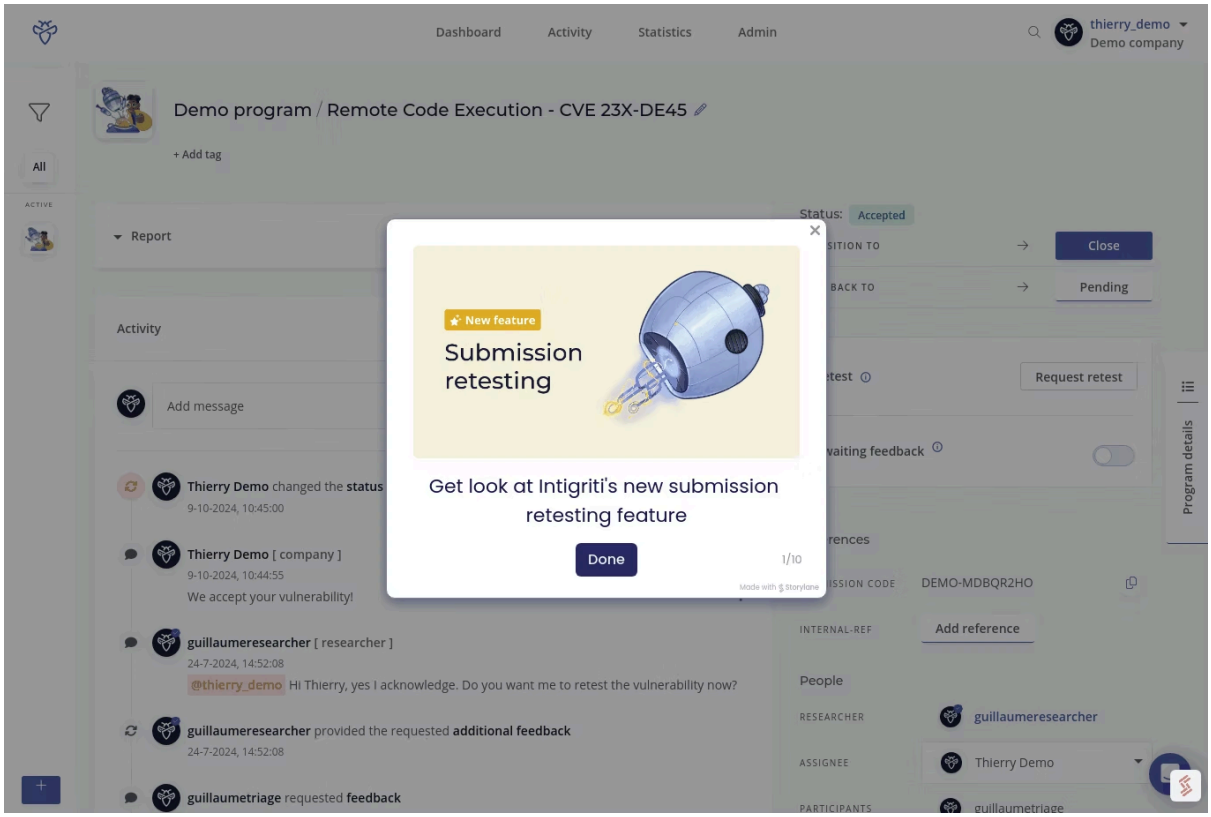
· **Miscommunication:** Retesting helps bridge any communication gaps between testers and developers. It ensures that both parties are on the same page regarding what constitutes a "fixed" bug.

· **Quality assurance:** By verifying bug fixes, retesting contributes to the overall quality assurance process. It helps maintain the software's reliability and user satisfaction.

Retesting is a critical step in the software development lifecycle that helps ensure bugs are genuinely fixed, preventing defects from making their way into the final product and maintaining the asset's integrity.

At Intigriti, we offer **submission retesting** as part of our bug bounty programs. When a vulnerability is reported through our platform, we don't just stop at remediation. Once the issue is addressed by the development team, our community of ethical hackers can perform submission retests to confirm that the fix is effective.

>> **Explore our interactive demo on submission retesting**



submission retesting

How does regression testing differ from retesting?

In contrast to retesting, regression testing is a broader process that involves re-running previously executed tests to ensure that recent code changes haven't negatively affected existing functionality. It takes a holistic approach to validate that the entire system still works as expected after modifications. This is particularly important in Agile or iterative development methodologies where changes are frequent.

To illustrate, imagine the same login feature was fixed, but the developer also made changes to the user profile section. Regression testing would involve not only checking the login functionality but also ensuring that the user profile section and other related features still work correctly.

To summarize, retesting is about verifying specific fixes, while regression testing is about ensuring that new changes haven't introduced new issues elsewhere in the system. Both are essential for maintaining software quality and reliability.

Regression testing vs bug bounty

Regression testing and bug bounty programs are two distinct approaches to identifying and addressing software issues, each with its own purpose and methodology. However, their goals can intersect in the broader context of improving software quality and security.

Regression testing is a process conducted by the development team to ensure that recent code changes haven't introduced new bugs or negatively affected existing functionality. It involves re-running previously executed tests to validate that the software still behaves as expected. Regression testing is typically automated and integrated into the continuous integration/continuous deployment (CI/CD) pipeline. Its primary purpose is to maintain the asset's stability and reliability as it evolves.

Bug bounty programs, on the other hand, are crowdsourced initiatives that incentivize external security researchers and ethical hackers to find and report vulnerabilities in a company's software. These programs are designed to leverage the collective expertise of the security community to uncover issues that internal teams might miss. Bug bounty programs primarily focus on identifying security vulnerabilities, with rewards often tied to the severity and impact of the discovered bugs.

Maximize reliability through complementary testing strategies

Regression testing, retesting and bug bounty programs complement each other in several ways:

Comprehensive coverage

Regression testing ensures that existing functionality remains intact, retesting verifies specific bug fixes, while crowdsourced security testing identifies new vulnerabilities. Together, they provide a comprehensive assessment of the software's health.

Early detection

Regression testing catches issues early in the development cycle, while retesting ensures that fixes are effective. Crowdsourced security testing can uncover vulnerabilities that might have been missed during initial development, providing an ongoing layer of security.

Diverse perspectives

Internal testing methods like regression testing and retesting benefit from the structured and systematic approach of the development team. Crowdsourced security testing, on the other hand, brings diverse perspectives and expertise from the broader security community, enhancing the overall robustness of the testing process.

In summary, while regression testing, retesting, and bug bounty programs have different scopes and methodologies, they share the common goal of improving software quality. Furthermore, they can intersect in identifying regressions and enhancing overall software reliability and security.

Intigriti's bug bounty programs can facilitate your retesting needs to ensure comprehensive software quality. To complement your existing testing strategy and elevate your software's reliability, get in touch today!