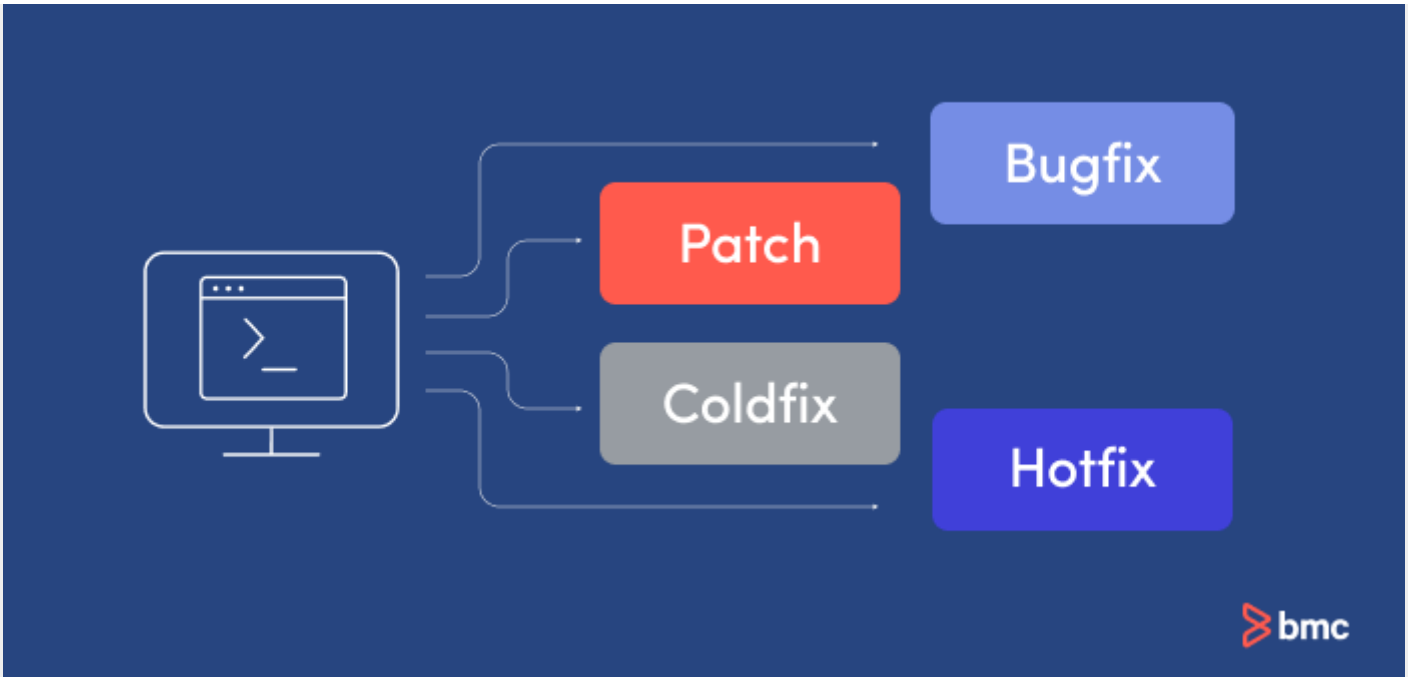
HOTFIX VS. PATCH VS. COLDFIX VS. BUGFIX: DIFFERENCES EXPLAINED



Programmers have to be on their A-game to fix bugs, especially once the software is live and people are actively using it. Depending on the type of bug, you'll have to decide the best way to debug it—with the least amount of impact to the user experience.

Things get even more urgent when a security vulnerability is discovered, and it's all-hands-on-deck until a solution is implemented that will prevent successful exploitation of the system's weakness, removing or mitigating a threat's capability.

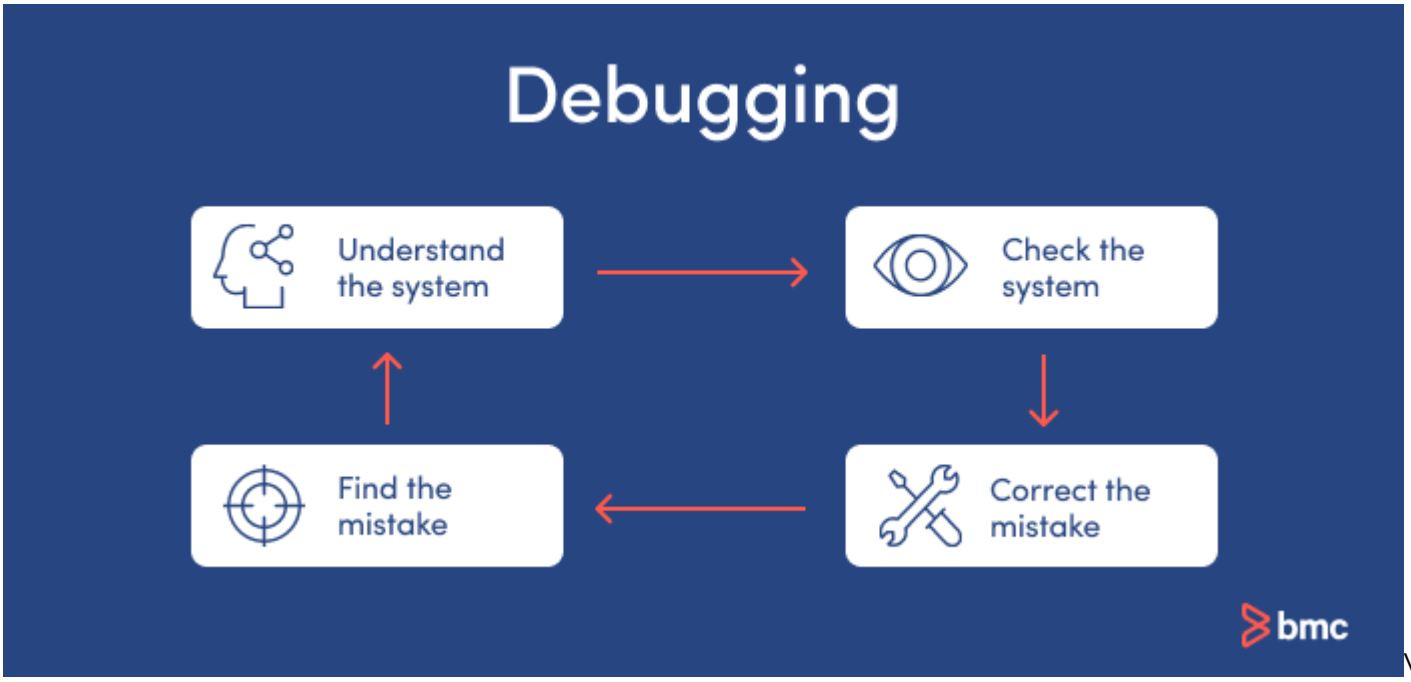
Finding and fixing the problem is just the first step. Then, you have to decide how to fix the problem—and how to roll it out in a way that minimizes the impact to users. That remedy might be delivered via:

- Patch
- Hotfix
- Coldfix
- Bugfix

Those terms are often used interchangeably, but there are differences in each one based on how a programmer incorporates their solution into the software.

[for enterprise operation ›]

What is debugging?



We're all familiar with the term "software bug"—a programming defect or glitch that creates errors within a system or software. Removing these bugs is a practice called "debugging."

Debugging is the process of finding, analyzing, and fixing errors in software or hardware to make sure it works as intended. Common computer bugs include:

- Syntax or type errors
- Typos and other simple errors
- Implementation errors
- Logical errors

Even though the name makes these errors sound small and only mildly irritating, like a gnat, the presence of errors and process for fixing them can be time-consuming and risks missing deadlines and necessary fixes.

For this reason, developers follow a specific process to systematically find each bug, understand its behavior, diagnose the root cause of the problem, fix or patch it, then test and document the fix. A reliable process leads to reliable results.

Types of software updates that can fix bugs

What is a patch in computing?

In today's computing terminology, a patch is data that updates an existing software version's code by modifying or replacing it using a publicly released, executable program.

Generally, computer patches are automatic updates that self-install packages in various sizes, from

a few kilobytes to over 100 megabytes. As any Windows user can confirm, the installation of certain large patches (on Patch Tuesday, of course) can cause interruptions and downtimes, and sometimes even require a system restart or two.

Most software patches are delivered on a set schedule. They can be included in the product's new version release, with additional updates and fixes.

Patching has come a long, digital way.

In the early days of computing, a patch fix was, quite literally, a patch. Analog computers used punched cards and paper tapes to input programs that machines used for performing their calculations. These "decks" contained rows of holes and spaces that were a computer's software and, just like today, the software suppliers would need to make changes to the programming.

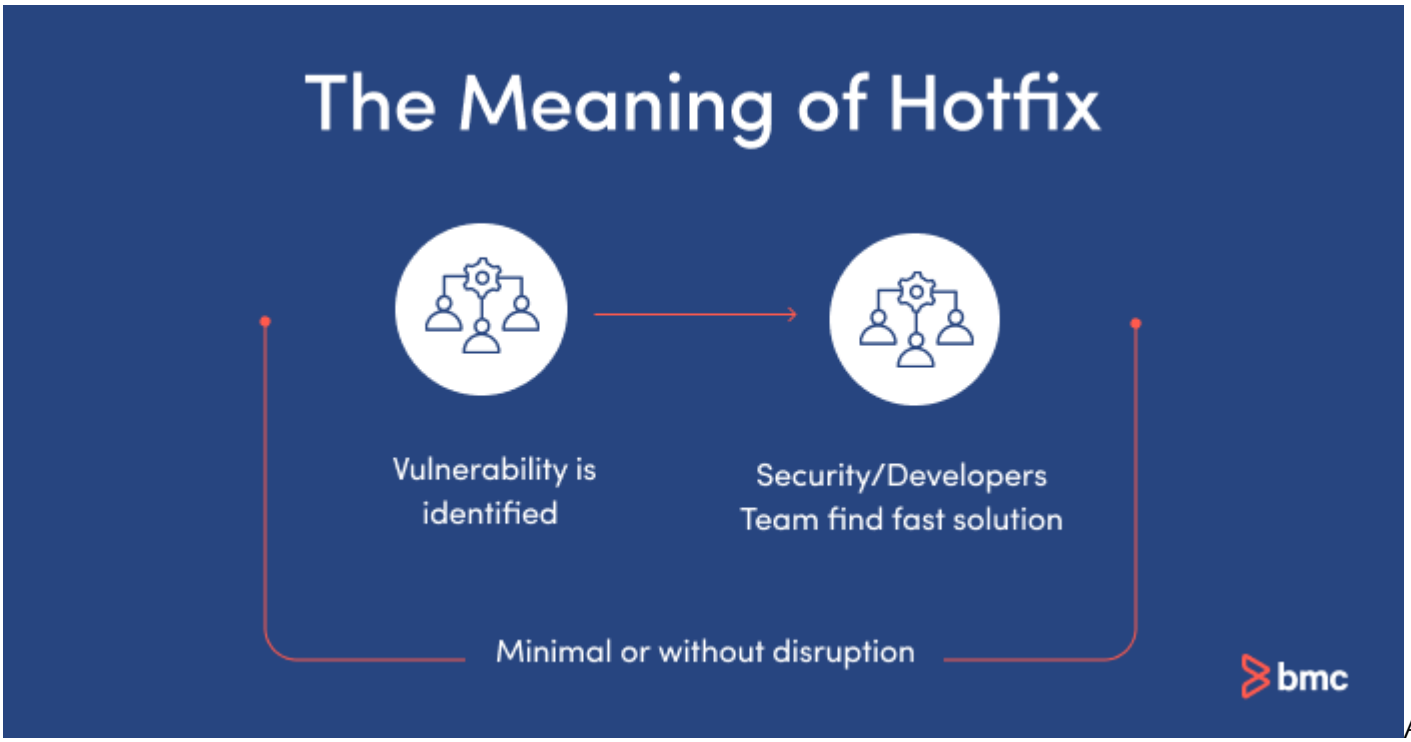
These updates were distributed on smaller pieces of paper tape or punched cards, and the recipients were expected to cut the bad part of the code out of the deck and patch in the replacement segment—hence the name.

When do software developers use patches?

Computer patches are often temporary fixes between full releases of software packages. Patches are used to correct both large and small issues that may or may not require immediate attention, such as:

- Fixing a software bug
- Installing new drivers
- Addressing new security vulnerabilities
- Addressing software stability issues
- Upgrading the software

What is a hotfix?



A

hotfix is a correction done in a quick and targeted manner to live, in-use software. Another term for it is a quick-fix engineering (QFE) update.

Hotfix vs. patch

What differentiates a hotfix from a regular software patch is that a hotfix is:

- Done to a live system.
- Applied Immediately.
- Done without creating downtimes or outages.
- Meant to resolve an urgent concern, to prevent a system failure, security breach, or issue that could lead to significant disruptions.
- Not part of a regular software update release cycle or normal development flow.
- Not typically part of a public release.

Unlike patches, hotfixes address very specific issues, such as adding a new feature, bug fix, or security fix, or changing the database schema.

Here's a hotfix example: Let's say a bank learns that their banking app could be hacked, exploiting and revealing user data like passwords, usernames, and account information. Even if the hack hasn't occurred yet, it's a risk so significant that identifying its potential requires urgent action. The security team will scramble to deliver a hotfix that solves the vulnerability as soon as possible, with minimal disruption.

What is a coldfix?

A coldfix is a correction to address a bug or problem that is done as part of a regular maintenance process. Coldfixes are planned, scheduled, completed, and tested in a controlled manner. They are not deployed to live software, and are announced during regularly scheduled updates.

Hotfix vs. coldfix

Where a hotfix is executed quickly without restarting any systems or hardware, a coldfix is just the opposite. Implementing a coldfix requires users to log out of the software while entire systems need to be rebooted for fixes to go into effect.

Coldfix example: These types of updates are common in online multiplayer games, for example, so they're normally announced several days ahead of time to give users advanced notice the service will be unavailable while the fix is completed. Notices generally include estimated times the service will be back online since outages can last from a few minutes to several hours depending on the update.

What is a bugfix?

A buxfix is an update to software that addresses a flaw in the code.

Implementing a bugfix, also known as a program temporary fix (PTF), can be as simple as adding missing parentheses in a piece of code. But the fix can become quite challenging if the symptoms don't clearly point to a cause.

For instance, the cause and the symptom may be remote, with either located in the program code

and the other in the execution of the program, or both.

Symptoms can also be difficult to reproduce for better understanding the problem. Once you've uncovered the root cause and issued a bug fix, however, it's not uncommon for your programmers to find that one bugfix can actually introduce a new bug.

Hotfix vs. bugfix

The difference between a hotfix and a bugfix lies in the timing and execution of the correction. Bugfixes generally describe issues that are found and resolved during production or testing phases or even after deployment as part of the normal release cycle of a product. Hotfixes are applied only after the product has been released and is live.

Summing up the software fixes

bmc

Patch	• A temporary fix on a product • Often automatic updates with self-install packages • Usually planned; between full releases of software packages
Hotfix	• A fix on live, active software or apps • Zero to minimal downtime for users • Immediately for live systems
Coldfix	• A fix on live, active software or apps • Visible impact, via downtime or system restarts • Planned out and communicated early to users
Bugfix	• A fix for issues found through development lifecycle, but before release • Zero impact on users • Applied during production or testing phases

What are bug bounties?

As software increases in complexity, debugging before and after a product launches is vital for protecting your brand.

Applications are increasingly complex, multi-threaded, and large, with a greater number of developers working on them. All this complexity makes tracking down bugs more difficult and

unpredictable. Multithreaded programs:

- Lengthen the time elapsed between the root cause of the bug and its detection.
- Make bugs difficult to track down and reproduce.

Software bugs are a risk too big for you to ignore. Programmers will spend weeks hunting them or even offer bug bounties to get help finding the problems in their code before they can apply the right fix.

How to avoid software bugs

The only ways to avoid computer bugs and the time spent on fixing them are to write better code. And until everyone starts writing perfect code, we can expect a few more software bugs to get into places they shouldn't be.

Related reading

- BMC DevOps Blog
- Patch Management: A Brief Introduction
- Error Budgets Explained: Risk & Reliability in One Metric
- Deployment Pipelines (CI/CD) in Software Engineering
- Deploying vs Releasing Software: What's The Difference?
- What Is Spaghetti Code?