rezilion

Rezilion's Guide to Vulnerability Management

EACH VULNERABILITY THAT YOUR SCANNER FINDS GENERATES WORK. But what is the risk associated with each vulnerability? This guide will take you through the steps to accurately assess vulnerability risk and reduce code bloat.

Your attack surface probably isn't as wide as you think

For overwhelmed and overtasked security teams, finding meaningful signals amid all of the noise from numerous security tools is a constant challenge. Vulnerability scanners — automated tools that allow organizations to check for weaknesses in their networks, systems, and applications — add to this environment of feeling overwhelmed, overworked, and always behind.

Make no mistake: scanners are essential. Industry standards and government regulations often mandate vulnerability scanning, and there is little debate that scanners are part of best practices for security defense today. And scanners have improved over the years since they were first introduced into common practice. They are now used in multiple places along the Software Development Lifecycle (SLDC), which means they're finding more and more vulnerabilities and helping with overall security efforts.

But when scanners identify a vulnerability — often hundreds or thousands daily — those bugs are identified, prioritized, and added to a backlog, creating a massive amount of work for security teams who must investigate whether they pose a real threat and then patch accordingly. This mass of vulnerabilities that scanners identify is referred to as the "Perceived Attack Surface." Most organizations measure themselves against this attack surface because they don't have anything else to refer to for context when it comes to risk.

It's time to focus on your *actual* attack surface rather than perceived risks

The reality is, not all vulnerabilities require patching. The vast majority of deployed code is never actually used in runtime. Only vulnerabilities running in memory are exploitable. Your real attack surface — the one that matters for patching — is your exploitable attack surface. Most vulnerabilities identified by scanners are in code and components that are never run in memory and therefore pose no risk.

What's your actual attack surface? It varies, of course, but Rezilion data reveals that, on average, the real attack surface for most organizations is less than 30% of the identified or perceived attack surface. That means more than half of vulnerabilities do not require patching. The potential for time savings through patching is massive.

Reducing code bloat: Another essential step in vulnerability management

Another challenge for Dev and Security teams is code bloat. Bloated code occurs where the runtime environment contains useless pieces of code, like libraries or service binaries, that will invariably lead to software inefficiencies and security vulnerabilities. It is typically caused by inadequacies in the language in which the code is written, the compiler used to compile the code, or even a feature-focused programmer who writes more lines of code than required for the application to work.

Of course, most programmers will insist that their code is clean, so how does code bloat even happen, and how does it differ from tech debt?

Bloated code in the runtime environment is one of the most common tech debt issues. At the beginning of the development process, you start with an image that "seems fine for our current needs," then install whatever it takes to make everything work without ever removing any of the installation components or reducing the base image contents. By continuously adding features and functionality without going back and eliminating unused components, you get tech debt that accumulates over the life of the image. The use of open-source libraries amplifies this problem because developers will pull entire libraries into the image and only actually use a fraction of the functionality. The remaining 80+% of the library becomes bloated code.

Because many teams find themselves in a ceaseless development cycle, it's easy to overlook bloated code until later in the process. Then, when it comes time to test the application, all the flags go up, adding otherwise unwarranted delays and risk to the release.

There are three main reasons for removing bloated code, benefiting developers, DevOps, and Security.

1. FASTER DEVELOPMENT: Every piece of code in the image comes with a price. Unnecessary code slows down the whole building and testing process, leading to more time-consuming tests and lengthening the development lifecycle. Removing bloated code means you're only testing the code required to run and not wasting cycles on unused code.

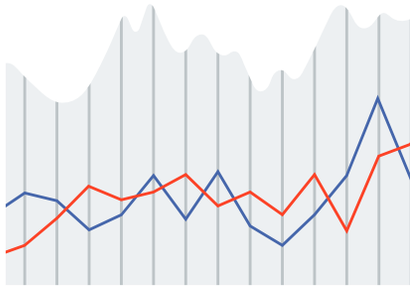
2. INCREASED EFFICIENCY: Every piece of code consumes memory space. The more memory used, the slower the uploading, loading, and performance time. It also requires every monitoring tool (testing, performance monitoring, and profiling) to take more time and resources, even though it offers no value.

A significant portion of development time is spent on running pipelines to ensure no changes made at the codebase step on someone else's toes. So you can imagine how speeding up this process can improve the product release velocity.

3. IMPROVED SECURITY: Although it's not used, when loaded, bloated code is still vulnerable and can expose the whole product to attacks. If that weren't enough — and really, it should be — scanners can't distinguish between code that merely exists, and code that is actually used and loaded. That means the unfortunate application security manager will be inundated with false alarms from the multiple alerts generated during security scans.

Get the real picture of your vulnerabilities for time and cost savings

Vulnerability Management (VM) programs, and the CISOs that run them, are judged on their ability to reduce risk. Vulnerability validation and backlog reduction are both critical, measurable, and reportable statistics that CISOs can use to convey the performance of their VM programs to their teams, to executive management, and their Boards.



Your real attack surface is less than 30% of the identified or perceived attack surface.

Please **visit Rezilion** to learn more about assessing your vulnerability risk and reducing code bloat.