ZERONORTH

Are Quality and Security Synonymous in Software?

A ZeroNorth White Paper

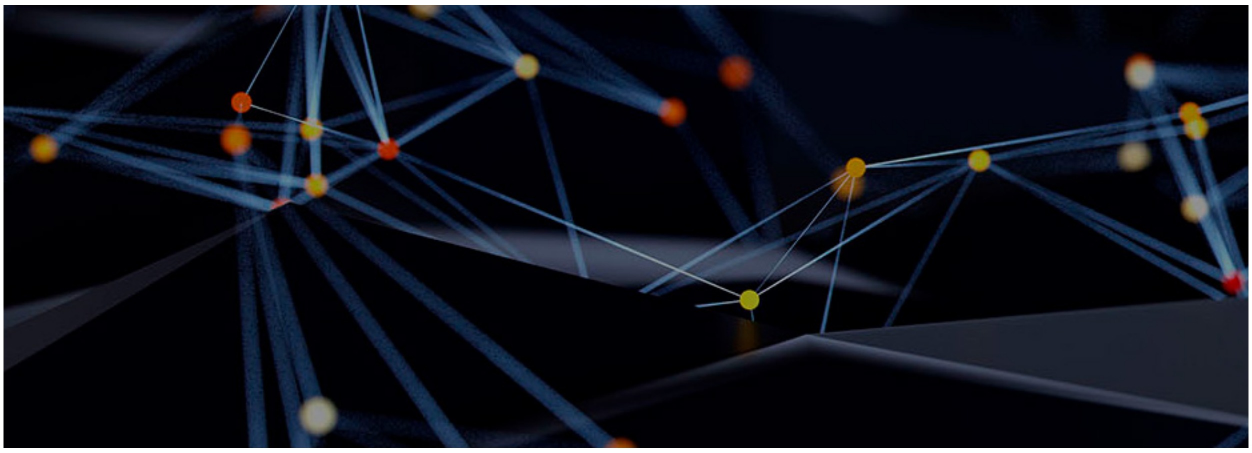
Reframing Security in Relation to Quality

Security and quality. Quality and security. What is the relationship between the two in today's software development world? Are they synonymous? If you're producing quality software, does that mean it's automatically secure? If your software is deemed secure, does that mean it's inherently high-quality? If you have a defect in either quality or security, does that equate to a problem with the other? Why are we even talking about the relationship between quality and security within software development? Is it really that important in today's world?

So many great questions, soon to be answered.

Let's get started by first defining both quality and security as they relate to software development.

Understanding Quality and Security in the Context of Software



By definition, quality software “will execute according to intended design and purpose based on business functionality.” A recent model put out by the American Society of Quality[1] lists eight hierarchical metrics to gauge the quality of software. And it’s interesting to note, security comes in at number five, sandwiched between performance efficiency and compatibility. Though the particulars of software quality depend on things like product type and intended usage, this fixed model helps us consider the true nature of software quality. It also identifies the following elements as being critical to high-quality software:

- maintainability: capacity to upkeep code
- portability: capacity to be maneuvered
- functionality: ability to carry out intended purpose
- performance: ability to work quickly and effectively
- security: ability to protect systems or data from unauthorized access
- compatibility: capacity to work with multiple components
- usability: ease of use without detailed instruction
- reliability: capacity to work consistently and overcome issues

[1] Source: What is Software Quality?

One area that's not mentioned in this list is quality function deployment (QFD), a focused methodology designed to honor the voice of the customer. Considering every organization has customers, whether internal, external, B2B or B2C, it's important to prioritize customer expectations and opinions when trying to assess the overall quality of software. Why? Because customer-driven quality provides a process for developing more secure products. In this way, keeping customers and products secure is precisely what pushes quality concerns to the forefront of software development.

Software Quality

Though quality can appear subjective, we know there are definable metrics for quantification. The following measurements help determine the perceived quality of software and offer some contextual background for assessment:

- **Defect density:** This popular metric measures the number of software defects per line of code. Density grade is the number of confirmed defects in a module divided by the size of said module. IT risk management experts generally agree, a defect density of "one or below" is desirable for an enterprise and translates into quality code—and below one for a DevSecOps model using CI/CD.

  Jim Routh, Head of Enterprise Information Risk Management at MassMutual, shares his thoughts. "Defect density has to be measured to be an effective control. Measuring defects allows owners (development leads) to determine what level of quality they have in their components at any stage of the SDLC. Developers need instrumentation, so they know exactly what their defect density is. When it's above a certain level, they know they have to take action to remove defect density below a pre-determined threshold before they can move code into a build process or production."

- **Defect escape rate:** This phrase may be the most critical, as it measures how many defects can "escape" into the wild and potentially wreak havoc. It's one thing to have exploitable vulnerabilities, but it's far worse to leave them exposed where they can be discovered by customers, leak data and/or tarnish reputations.

- **Code churn rate:** This term gauges how quickly code changes or turns over, possibly indicating problematic or lower quality software. However, some degree of churn is normal, and GitPrime data science found that typical teams can expect to operate in the neighborhood of 25% code churn (75% efficiency)[2].

Software Security

Software security, on the other hand, is a bit harder to quantify. It's defined as "software that won't put systems or data at risk of unauthorized access." Vulnerability management teams can simplify the process by viewing software from the vantage point of security. How easily can software fall victim to misuse or act outside its intended purpose? For reference, developers use the following terms when discussing the security level of software:

- **Rate of remediation:** As no software is 100% flawless, quick and effective remediation of any security gaps is paramount. This metric assesses how quickly internal teams can identify and correct vulnerabilities.
- **Code complexity:** This metric engages simple proxies, like the ease of maintenance and testing, to assess the complexity of written code. Because less complicated code is easier to maintain and test, developers believe it can reduce vulnerabilities and increase security.

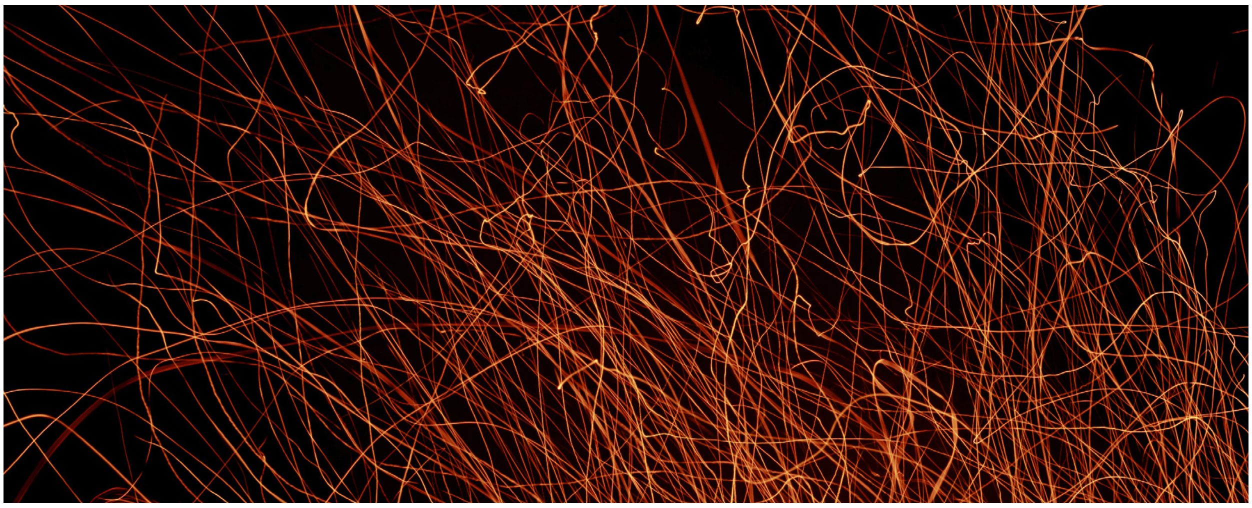
It's safe to say, quality software will perform as designed to suit business functions, including those customer-driven metrics of QFD. And secure software minimizes cyber risk by preventing outsider attacks on systems and data.

But the question remains, how do quality and security relate to each other?

Let's dive deeper.

[2] Source: Introduction to Code Churn

Untangling Notions of Quality and Security



Security and quality share a long, yet separate, history. Dating back to the era of rigid, sequential, non-iterative Waterfall methodology, the two siloed camps functioned autonomously during development, rarely, if ever, interacting.

In the past, development teams focused on quality, and IT oversaw security in a vaguely-defined and mostly inoperable fashion. When the two tribes did interact, security ran checks and issued remediation orders, as development grew frustrated over the slowing workflow. This relationship often lapsed into contention, with security viewing development as an enterprise risk management and compliance monitoring liability, and development eyeing security as an impediment to productivity. The agile, collaborative culture of DevSecOps, embraced by progressive teams, is a relatively recent phenomenon which we will discuss more later.

It's well-documented, security takes a backseat to speed and quality. Security is typically addressed last in the development process and compressed first when time is short. Traditionally, the development process pecking order is speed to market and quality of product, with security at a distant third. Many owners, when forced to decide, will opt to deploy a product with known vulnerabilities rather than fall off schedule and fail to meet business goals. Security's backburner status is most apparent when examining team ratios. A typical team ratio is one hundred developers to ten IT operations personnel to one security staff member.

That said, the days of minimizing the importance of security are over. Just ask any multitude of companies who've made front-page news with embarrassing and expensive data breaches. In February 2019, a large Australian property valuation firm allowed the compromise of a critical online platform, exposing the home loan details of over 100,000 customers[3]. This security lesson proved costly, as the company now estimates a revenue loss of $5-7 million[4]. On March 2, 2020, Walgreens, the second-largest US pharmacy chain, announced a mobile application messaging feature error that exposed the names, addresses, personal messages and prescription information of its users[5]. The number of impacted customers hasn't been divulged, but the Walgreen's app has over 10 million downloads.

Still not convinced of security's importance? Consider these recent numbers. In just the first six months of 2019, 3,800 publicly disclosed breaches occurred[6]. The sophisticated attacks of today don't come cheap, as the average global cost of a data breach is now $3.9 million[7]. Today's security teams can no longer afford a soft security posture, which means the case for reframing security in relation to quality has never been stronger. Despite their divided and unequal history, quality and security are undeniably connected. Here's why:

Quality problems can become security issues.

There's no question—today's bugs can become tomorrow's vulnerabilities. We need look no further than software testing to find the evidence. When testers pummel software with randomized input data, it may trigger the software to perform outside it's designed purpose. By definition, this function is a quality defect that provides pathways for invaders, creates vulnerabilities and ultimately becomes a security issue. Also, when quality assurance (QA) teams test software, they typically don't have all the necessary security controls in place to protect production data, which creates a huge security risk from both external and insider threats.

[3] Source: Home loan details of 100,000 customers hacked in major data breach
[4] Source: Landmark White drops revenue forecast by $11.5 million after data breach
[5] Source: 2020 Data Breaches: The Worst So Far
[6] Source: Data Breaches Expose 4.1 Billion Records in First Six Months of 2019
[7] Source: What's New in the 2019 Cost of a Data Breach Report

John Steven, CTO at ZeroNorth, speaks to the correlation between quality defects and security issues, "It has been endlessly debated as to whether security is a subset of quality, or vice-versa. While this debate is largely academic, security initiatives need to understand that many whole classes of security vulnerabilities emerge from quality defects. The canonical example, 'Command injection,' with specific examples like Buffer Overflow, SQLi or XSS for instance, follows developers from one language and tech stack to another."

Steven goes on to explain, "Attackers exploit underlying quality defects such as a failure to properly delineate code from data, insufficient validation and filter/escaping of input, as well as a lack of encoding/encapsulation of output—all 'quality issues' at their core."

Security issues can also become quality problems.

Even the most well-written and developed code is vulnerable to Cross-Site Scripting attacks (XSS). Malicious scripts can be successfully injected into trusted code when developers and QA teams overlook proper validation and sanitization procedures. These XSS attacks then create a security breach with the strength to compromise otherwise quality code. Quality software that's victimized by outside attacks now contains a security defect, which results in a quality problem. In this way, security and quality are inextricably linked.

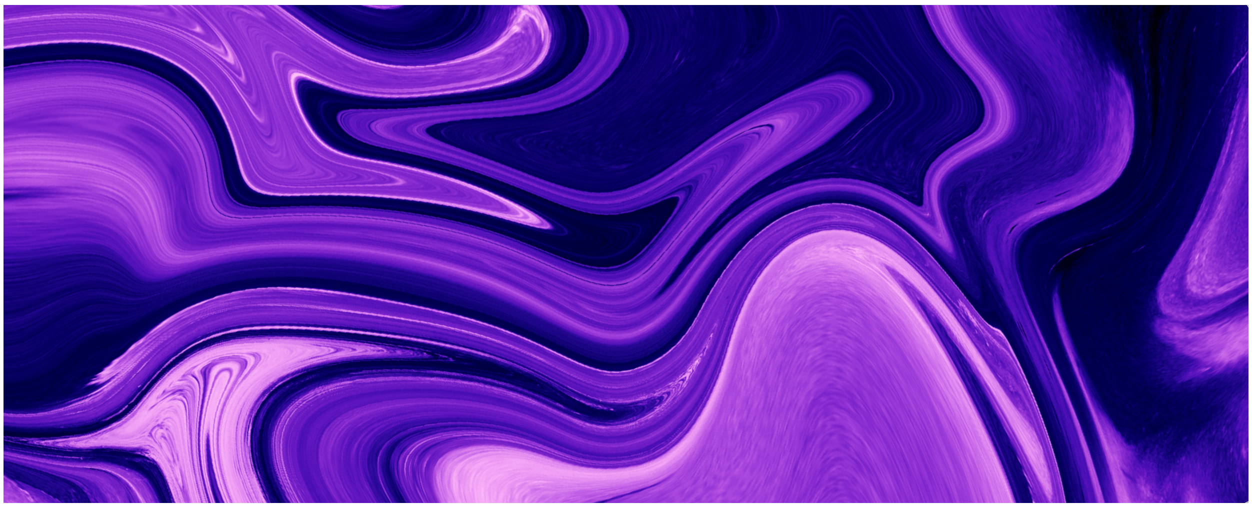
Any definition of quality must include security.

Let's apply some logic to our working definition of quality. Quality software must "execute according to intended design based on business functionality." Can we all agree, a prime "business function" means satisfying the end customer? QFD says yes. Can we also recognize that vulnerable, insecure software will fail to satisfy? If insecure software fails to please the customer, it's not executing on the intended design and can't be labeled as quality.

Though historically viewed as the invisible one, security must be considered when defining quality within software. Customers expect and demand secure products, even if they don't always realize it. Regrettably, security often lives under the radar of appreciation, drawing attention only when absent. However, security's importance and overall impact on software quality are undeniable.

Today's evolving definition of quality isn't complete without security. Dave Hatter, cybersecurity consultant at Intrust IT, describes this importance in what he feels is a close-knit relationship. "Quality and security share a symbiotic relationship, with both needing to be viewed from a risk mitigation perspective. In today's environment, security is equal to, if not more important than, quality in many (if not most) instances."

Blending Quality and Security to Create DevSecOps



The traditionally separate relationship of development and security is long overdue for evolution, a cultural shift now known as DevSecOps. The name suggests a blending of development, security and operations, with security finally claiming its rightful spot in the software development process. The DevSecOps methodology anchors on the "shift left" philosophy of integrating cyber risk management into the architecture and development process from inception. Built-in, not bolted-on, as they say.

With DevSecOps, security is baked into the code from the start, during the early stages of development. Security is part of the architecture, and the application of automated testing throughout the development process drives a higher level of both product quality and DevOps security. Security issues present earlier, making life easier for developers and less costly for management. Hatter explains, "Addressing potential bugs and vulnerabilities is going to be much less expensive if you have security at the table from the beginning and frankly, a lot more successful."

DevSecOps also gives way to a long-overdue cultural change. This new mindset orchestrates the blending of security and quality by placing the responsibility for security in almost every department. Old silos crumble, and the result is a newfound collaboration between development, testing, operations and security. Teams learn more about each other's roles, with developers becoming increasingly educated and skilled in security practices. A DevSecOps methodology encourages developers to view security as integral to software quality and a significant part of their daily work.

It's important to remember, developers don't intentionally write insecure code. Often times, they just don't have the tools or capabilities to enable the creation of error-free code within the new velocity. Jim Goepel, Professor of Cybersecurity at Drexel University's Kline School of Law and the LeBow College of Business, stresses security as culture to support developer buy-in. "Security must be built into the culture early on, becoming top of mind, so developers understand upfront why they need to do these things. Yes, it may slow them down a little, but they need to see the result is definitely worth it."

DevSecOps is changing cultures, allowing teams to write more secure code and impressing industry veterans. Aaron Wise, Head of IT and Engineering at ZeroNorth, considers himself a DevSecOps evangelist, "Shifting left is absolutely where we want to be. It improves the development process in many important ways, like moving education around security closer to development where it belongs. It also embraces the idea of taking security functions, traditionally at the end of the development process, and making them first-class citizens through integration into everyone's daily activities."

For organizations looking to embrace DevSecOps and successfully blend security into their software development process, what are some key action steps they can use to drive change? We're glad you asked…

Actions your organization can take to build a DevSecOps culture:

- *Cultivate a new shared mindset from the top down.* Encourage the adoption of a "security is everyone's job" philosophy.
- *Train developers with security team oversight.* This effort includes conducting "post-mortem" education, sharing source code repositories and establishing authentication/encryption libraries.
- *Support security involvement from the beginning of the development process.* This drives better risk-based decisions around business functions. Security scanning must be integrated consistently and comprehensively across the software development lifecycle (SDLC) to effectively jumpstart product security initiatives.
- *Monitor all security issues in the same work tracking system.* This promotes maximum visibility across all departments and allows you to centrally manage and execute all application security scans accurately, while transparently tracking risk across the SDLC.
- *Deploy automated testing on every code commit to catch vulnerabilities earlier.* For easier and more affordable remediation, this needs to happen in the development process. Scanning should not just be at the repository but across the CI/CD as well. The deployed solution needs to scan across the pipeline to see all vulnerabilities and reduce noise, so critical issues rise to the top and developers don't waste their time chasing low-risk issues.
- *Allow full visibility from a single, centralized platform.* This includes all security metrics, environments and performance to the value stream. Formulate a consolidated view of risk uncovered through vulnerability scanning, while emphasizing the need for all departments to keep security front-of-mind while performing their daily activities.

Though relatively new, DevSecOps is proven, battle-tested and poised to grow. Today, 8% of companies are securing 75% or more of their cloud-native applications with DevSecOps practices, a number that's expected to jump to 68% in two years[8]. Jim Goepel speaks of a dangerous future for organizations who do not bolster their security posture with DevSecOps. "We have so many vulnerabilities right now that criminals can easily exploit. Organizations who fail to prioritize security by building it into the development proess could face disastrous results."

It's clear, today's definition of quality software is a work in progress, forwarded by the reframing of security as a quality issue. But with the arrival of DevSecOps, security now takes its rightful place in the development process, to be owned, valued and supported equally by all departments.

So, where does that leave us?

The most accurate understanding of the relationship between quality and security may be as equals under the metric of software *integrity*. Their equivalent standing under the umbrella of integrity helps reduce both quality and security issues, thereby resulting in safer software that consistently meets design expectations. Ideally, software is both secure and highly functional, always performing at the highest level of integrity to fulfill its ultimate business function of satisfying and protecting the customer.

Equal under the Umbrella of Integrity

Where does this bring us in our quest to understand the relationship between quality and security in software development? Quality and security aren't synonymous in software development, but they do share an undeniable connection—and to many, a symbiotic relationship.

Must software be secure to be considered quality? Yes, if we agree the design is to meet the business functionality of pleasing the end customer.

Can secure software be deemed quality? Not necessarily. Even though secure, it may not satisfy the other functional and non-functional requirements of quality.

[8] Source: For Cloud-native App Security, Few Companies Have Embraced DevSecOps

*ZeroNorth would like to thank the following subject matter experts for their collaboration on this white paper:

- Jim Goepel, Professor of Cybersecurity at Drexel University's Kline School of Law and the LeBow College of Business
- Dave Hatter, cybersecurity consultant at Intrust IT
- Jim Routh, Head of Enterprise Information Risk Management at MassMutual
- John Steven, Chief Technology Officer at ZeroNorth
- Aaron Wise, Head of IT and Engineering at ZeroNorth

To learn more about how ZeroNorth can improve your existing application security program—or even help you build one out for the first time—contact us or request a demo.

ZERONORTH

ZeroNorth is the first company to deliver risk-based vulnerability orchestration across applications and infrastructure. By orchestrating scanning tools throughout the entire software lifecycle, ZeroNorth provides a comprehensive, continuous view of risk and reduces costs associated with managing disparate technologies. ZeroNorth empowers customers to rapidly scale application and infrastructure security, while integrating seamlessly into developer environments to simplify and verify remediation. For more information, follow ZeroNorth on Twitter (@ZeroNorthSec), or LinkedIn—or visit www.zeronorth.io