M+E JOURNAL

WHAT'S NEXT?

An M&E industry that's learned to adapt and excel after a year like no other, for one

LOCALIZATION
Dubbing from home is a work in progress

SECURITY
How to beat piracy and secure your business during a pandemic

SMART CONTENT
The new ways content players are using data to connect with consumers

NEW WORKFLOWS
Adopting the latest tools fuels a successful change to remote work

20.02

HOW TO BUILD SECURITY *INTO* THE DEVELOPMENT PROCESS

Security done early is less expensive than security done later

ABSTRACT: People often think ‘build first, then secure.” But that’s wrong — it’s both more expensive and less effective to do it that way. Instead, replace that lie with this truth: “secure as you build.’ This article explains why to build security into the development process, and how to do it.

By Ted Harrington, Executive Partner, Independent Security Evaluators

Most mornings, I have a smoothie for breakfast. It’s packed with wholesome stuff: organic spinach, bananas, pea protein, cashew butter, and plenty of good ol’ H2O. Once I’ve poured my smoothie, there are two ways I can clean the blender:

1. Do it later. Let it sit in the sink while I rush off to do other urgent things. When I come back later, those nutritious ingredients have hardened and are a pain in the neck to scrub clean. As a result, I have to soak the blender, disassemble it, scrub it, and reassemble it.

2. Do it now. As soon as I pour out the smoothie, add a little soap and water into the blender, run it for ten seconds, and rinse. The thing literally cleans itself. No disassembly, no scrubbing.

Doing it later is easier at first but a nightmare overall. Doing it now is easier overall but often dismissed at first.

That’s the same scenario you are in as you decide what to do about security. Most people unwittingly pursue the approach that makes their lives harder. Instead, pursue the approach that makes your life easier. If you want to do it the easy way, you need to take action early by building security into the development process. If you don’t, you’ve signed up for the hard way.

Build security into the development process: it is more effective and less expensive than delaying security.

‘BUILD SECURITY IN’ IS LESS EXPENSIVE THAN ‘BOLT SECURITY ON’

Security done early is less expensive than security done later. This is thanks primarily to avoiding that unnecessary work while you also save on consulting fees. Upon analyzing thirteen years of our own assessment data, we discovered that companies who “built security in” spent 10.1 percent less on consulting fees than those who didn’t. That’s not a mind-blowing savings, but hey, 10.1 percent is 10.1 percent! That’s real money that doesn’t go out your door. Why waste it? You might not even realize it, but when you push security off until later, you’re taking on that waste. You’re making it cost more. This costs more because your security partner has to spend more time and effort (which equates to your money) in addressing a higher volume of security issues than if those security issues had been addressed during the development process first.

Better yet, as cool as it is to save 10.1 percent on consulting fees, that pales in comparison to the savings in terms of your effort. It’s easiest to fix a flaw at the moment when it’s introduced. It’s exponentially harder to fix it later. For example, a flaw introduced in the design phase that isn’t addressed until after deployment is going to require a ton more effort to fix. In fact, the data shows it takes twenty-five times more effort.

Twenty. Five. Times. More. Effort.

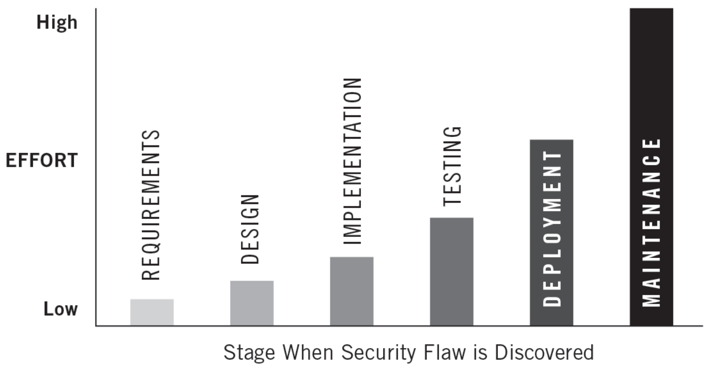
That’s bananas!

Unlike consulting fees, these costs are not easily visible on your financial statements. Something that should take one hour now takes twenty-five hours. This is straight-up lost efficiency. You just burned twenty-four additional hours to get the same outcome if you’d just done it earlier.

That’s nuts.

When developers are fixing systems they already built, they’re not working on other things. Salary, benefits, and overheads appear on company financial statements either way; however, developers’ productivity plummets. This decreases the value the company gets out of each developer. It’s a massive hit to your output. If this isn’t alarming you, you’re not human. No one likes waste, especially when it doesn’t improve outcomes.

The later you secure, the more painful it becomes



No rational person wants anything to be twenty-five times harder than it needs to be. Yet, people do this all the time. This happens not because people want to make their lives harder; it happens because they don’t realize it’s happening. My mission is to help you stop incurring this waste. Whenever you postpone security, you incur this terrible tradeoff. Every single time.

Introducing vulnerabilities is going to happen — that’s a reality of software development (and is the reason that this book exists). But it’s hugely inefficient to let them linger in your system any longer than they need to. The book Applied Software Management spells out this truth. It explains that (a) most vulnerabilities are introduced during development, but (b) most testing isn’t done until after release, yet (c) remediation costs rise exponentially after release.

It’s literally that simple: when you delay, it costs you.

You want to shrink that productivity hit. You want to harvest that effort so you can use it for other things instead. When you “build security in,” you convert this waste into efficiency. You save effort. You maximize productivity.

HOW TO ‘BUILD SECURITY IN’

Now that you’ve explored why to build security into the development process, let’s talk about how. Simply put: in each step of the development process, there’s a security step, too.

Take it.

__Ted Harrington__ is the executive partner at Independent Security Evaluators (ISE), the company of ethical hackers famous for hacking cars, medical devices, and password managers. He’s helped hundreds of companies fix tens of thousands of security vulnerabilities, including Google, Amazon, Microsoft, Netflix, and more. ted@ise.io @ISEsecurity

WHEN DEVELOPERS are fixing systems they already built, they're not working on other things. Salary, benefits, and overhead appear on company financial statements either way; however, developers' productivity plummets.

That might sound like a big undertaking, but really it's not. It's just a small change. You already have all of the right people in the right room having the right conversations; you just need them to consider security as well. The activities that are more involved (such as security testing) are handled by your security partner anyway, so the additional burden on your own developers is offloaded.

Many companies use a linear-sequential software development methodology, such as Waterfall. In this type of development methodology, the entire development project is approached as a whole, with each phase being completed across the entire system before the next phase begins. Here's how you'd build security into this type of methodology:

■ **During requirements gathering,** you discuss the problem that the system needs to solve. Simply expand the discussion to also consider your threat model. The requirements of the system dictate the features you'll need to develop. That determines which assets the system needs to access. These determine which attackers to consider and which attack surfaces to secure.

■ **During design,** you define the system architecture, determining which components to build and how they'll interact. Those inform how to implement Defense in Depth, a security approach that layers defenses in order to both minimize likelihood of breach and minimize damage resulting from a breach. (We'll explore this deeper in a moment.)

■ **During implementation,** you code the system. You know which areas of code are critical to security, and have those areas reviewed for security flaws specifically. You most likely aren't doing the security code review yourself anyway; your security partner is. So this important step isn't even a burden on your developers.

■ **During testing,** you evaluate system performance. In addition to functional testing, this is when you also do security testing. It happens in parallel with other testing efforts and is performed by your security partner. This frees your engineering resources to focus on other things. Vulnerabilities are discovered and reported to you so you can fix them. You'll invest effort in remediations, but as you learned in chapter 5, there's a way to manage that effort so it doesn't overwhelm you.

■ **During deployment,** you roll the solution out to customers and deal with the inevitable challenges that come with that. Now is when you also advise customers on configuration so they deploy your system securely. Whether this comes as documentation or hands-on configuration assessment (or both), they're supplied by your security partner. Thus, the system is deployed securely without too much effort from you.

■ **During maintenance,** you're in the never-ending cycle of resolving bugs. Same idea with security. Through reassessments, your security evaluators help you continually find and fix security flaws. As a result, you keep your system secure over time, all without too much heavy lifting by you.

SO, TO SUMMARIZE, YOUR SECURITY EFFORTS LOOK LIKE THIS:

You can apply these same principles if you adhere to an iterative methodology instead, such as Agile, Scrum, Kanban, or Rapid Application Development. Iterative methodologies are where a large development project is broken down into smaller chunks. Development then cycles through requirements, design, implementation, testing, and deployment on each feature (rather than on the entire project, as you would in a linear-sequential development methodology). These features are often referred to as "user stories," which describes the feature from an end user's perspective. To build security into this type of development process, all of the same actions mentioned above occur, done for each feature (or "user story") as you cycle through them. Here's how:

Whether you use a linear-sequential development process like Waterfall or an iterative one like Agile, you can — and must — build security in. If this guidance still doesn't address your unique development challenges, I'm only an email away: ted@tedharrington.com. Contact me and I'll point you in the right direction.

As one of our security analysts put it, "Think about security when you're doing things, not after you already did things." ⊞