

# It's about what you mean, not always what you say

Don't let your security requirements collapse under their own vague language

Security projects often start with confident declarations.

- + "We need tailgating detection."
- + "We need AI video analytics."
- + "We need badge access that integrates with our VMS."

These statements sound precise. They feel actionable. They're a great start. But in many cases, they are incomplete in ways that only become obvious once systems are tested in the real world. If you never move beyond these assertions and do the work to examine their hidden assumptions, you can set yourself up for confounding surprises when you go to deploy.

Consider what happens when you abstract the "what" out to something more like "why":

- + "We think we have unauthorized people entering the building."
- + "We're not capturing all the insights we need from our VMS data."
- + "Our investigators spend too much time looking at footage."

By proceeding from the problem, not what you assume to be the solution, you gain clarity on your current position and are situated to produce more detailed, complete requirements.





The problem isn't that security teams don't know what they want. It's that our efficient use of language often compresses complexity instead of expressing it. A single term often stands in for multiple behaviors, conditions, and risks that are not fully articulated at the outset. "Tailgating," for example, can mean several different things depending on how people move through a space, how doors are used, and what outcomes actually matter operationally.

As we talk about in our [recent white paper](#), testing has a way of exposing these gaps.

When a solution is exercised under realistic conditions, it becomes clear that what sounded like a single requirement is actually a bundle of related expectations. To continue the tailgating example, detecting one person following another through a door is not the same as detecting cross-entry. It's also not the same as knowing that under some circumstances, violation is innocuous—when one person has mobility problems, say, or when the two people know and trust each other. Technology often detects anomalies that are socially acceptable and procedurally allowed, so policy and procedure must pick up where tech leaves off. Each variation on a presumed requirement carries different implications for sensing, analytics, alerting, and response.

Untested assumptions can make a security deployment quietly go off track, even when your intentions are the best they can be. Vendors respond faithfully to the language they're given. Integrators design to the letter of the requirement. And only later does the organization realize that the system is very good at solving the wrong problem.

Testing doesn't magically "discover" better requirements. Rather, focused and intentional testing will drive the specificity you need to see the big picture. It reveals what a requirement is actually asking a system to do, once words give way to behavior. In that sense, testing is less about validation and more about translation: converting intent into something technology can reliably support.

That shift alone can snatch success from the jaws of frustration, chaos, and expensive uncertainty. The right testing yields precision at a level that can save entire security deployments from disaster.



For more details on testing philosophy and best practices for physical security systems, read our white paper, [Making test plans worth your investment: How innovations in security solution testing pave the way to on-budget results.](#)



ZBeta provides lab space for testing just about any solution under conditions that will closely resemble your built environment. For more information, visit [zbeta.com/labz](http://zbeta.com/labz).

**Our office:**

700 Larkspur Landing Circle, Suite 150  
Larkspur, CA 94939

**Email and web:**

info@zbeta.com  
www.zbeta.com

**Phone:**

(855) 559 2382

