Article humor

Stop Trying To Schedule A Call With Me

Mathew Duggan

10 Jan 2025 • 7 min read

One of the biggest hurdles for me when trying out a new service or product is the inevitable harassment that follows. It always starts innocuously:

"Hey, I saw you were checking out our service. Let me know if you have any questions!"

Fine, whatever. You have documentation, so I'm not going to email you, but I understand that we're all just doing our jobs.

Then, it escalates.

"Hi, I'm your customer success fun-gineer! Just checking in to make sure you're having the best possible experience with your trial!"

Chances are, I signed up to see if your tool can do one specific thing. If it doesn't, I've already mentally moved on and forgotten about it. So, when you email me, I'm either actively evaluating whether to buy your product, or I have no idea why you're reaching out.

And now, I'm stuck on your mailing list forever. I get notifications about all your new releases and launches, which forces me to make a choice every time:

- *"Obviously, I don't care about this anymore."*

- *"But what if they've finally added the feature I wanted?"*

Since your mailing list is apparently the only place on Earth to find out if Platform A has added Feature X (because putting release notes somewhere accessible is apparently too hard), I have to weigh unsubscribing every time I see one of your marketing emails.

And that's not even the worst-case scenario. The absolute worst case is when, *god forbid*, I can actually use your service, but now I'm roped into setting up a *"series of calls."*

You can't just let me input a credit card number into a web site. Now I need to form a bunch of interpersonal relationships with strangers over Microsoft Teams.

Let's Jump On A Call

Every SaaS sales team has this classic duo.

First, there's the salesperson. They're friendly enough but only half paying attention. Their main focus is inputting data into the CRM. Whether they're selling plastic wrap or missiles, their approach wouldn't change much. Their job is to keep us moving steadily toward *The Sale*.

Then, there's their counterpart: the "sales engineer," "customer success engineer," or whatever bastardized title with the word *engineer* they've decided on this week. This person is one of the few people at the company who has actually read all the documentation. They're brought in to explain—always with an air of exhaustion—how this is really my new *"everything platform."*

"Our platform does everything you could possibly want. We are very secure—maybe too secure. Our engineers are the best in the world. Every release is tested through a 300-point inspection process designed by our CTO, who interned at Google once, so we strongly imply they held a leadership position there."

I will then endure a series of demos showcasing functionality I'll never use because I'm only here for one or two specific features. You know this, but the rigid demo template doesn't allow for flexibility, so we have to slog through the whole thing.

To placate me, the salesperson will inevitably say something like,

"Mat is pretty technical—he probably already knows this."

As if this mild flattery will somehow make me believe that a lowly nerd like me and a superstar salesperson like you could ever be friends. Instead, my empathy will shift to the sales engineer, whose demo will, without fail, break at the worst possible time. Their look of pure despair will resonate with me deeply.

"Uh, I promise this normally works."

There, there. I know. It’s all held together with tape and string.

At some point, I’ll ask about compliance and security, prompting you to send over a pile of meaningless certifications. These documents don’t actually prove you *did* the things outlined in them; they just demonstrate that you could *plausibly fake* having done them.

We both know this. If I got you drunk, you’d probably tell me horror stories about engineers fixing databases by copying them to their laptops, or how user roles don’t really work and everyone is secretly an admin.

But this is still the dating phase of our relationship, so we’re pretending to be on our best behavior.

“Very impressive SOC-2.”

via GIPHY

Getting Someone To Pay You

We’ve gone through the demos. You’ve tried to bond with me, forming a “team” that will supposedly work together against the people who actually matter and make decisions at my company. Now you want to bring my boss’s boss into the call to pitch them directly.

via GIPHY

Here’s the problem: that person would rather be set on fire than sit through 12 of these pitches a week from various companies. So, naturally, it becomes my job to “put together the proposal.”

This is where things start to fall apart. The salesperson grows increasingly irritated because they could close the deal if they didn’t have to talk to me and could just pitch directly to leadership. Meanwhile, the sales engineer—who, for some reason, is still forced to attend these calls—stares into the middle distance like an orphan in a war zone.

“Look, can we just loop in the leadership on your side and wrap this up?” the salesperson asks, visibly annoyed.

“They pay me so they don’t have to talk to you,” I’ll respond, a line you first thought was a joke but have since realized was an honest admission you refused to hear early in our relationship.

If I really, really care about your product, I'll contact the 300 people I need on my side to get it approved. This process will take at least a month. Why? Who knows—it just always does. If I work for a Fortune 500 company, it'll take a minimum of three months, assuming everything goes perfectly.

By this point, I hate myself for ever clicking that cursed link and discovering your product existed. What was supposed to save me time has now turned into a massive project. I start to wonder if I should've just reverse-engineered your tool myself.

Eventually, it's approved. Money is exchanged, and the salesperson disappears forever. Now, I'm handed off to Customer Service—aka a large language model (LLM).

The Honeymoon Is Over

It doesn't take long to realize that your "limitless, cloud-based platform designed by the best in the business" is, in fact, quite limited. One day, everything works fine. The next, I unknowingly exceed some threshold, and the whole thing collapses in on itself.

I'll turn to your documentation, which has been meticulously curated to highlight your strengths—because god forbid potential customers see any warnings. Finding no answers, I'll engage Customer Service. After wasting precious moments of my life with an LLM that links me to the same useless documentation, I'll finally be allowed to email a real person.

The SLA on that support email will be absurdly long—72 business hours—because I didn't opt for the *Super Enterprise PlanTM*. Eventually, I'll get a response explaining that I've hit some invisible limit and need to restructure my workflows to avoid it.

As I continue using your product, I'll develop a growing list of undocumented failure modes:

"If you click those two buttons too quickly, the iFrame throws an error."

I'll actually say this to another human being, as if we're in some cyberpunk dystopia where flying cars randomly explode in the background because they were built by idiots. Despite your stack presumably logging these errors, no one will ever reach out to explain them or help me fix anything.

Account Reps

Then, out of the blue, I'll hear from my new account rep. They'll want a call to "discuss how I'm using the product" and "see how they can help." Don't be fooled—this isn't an attempt to gather feedback or fix what's broken. It's just another sales pitch.

After listening to my litany of issues and promising to "look into them," the real purpose of the call emerges: convincing me to buy more features. These "new features" are things that cost you almost nothing but make a huge difference to me—like SSO or API access. Now I'm forced to decide whether to double down on your product or rip it out entirely and move on with my life.

Since it's not my money, I'll probably agree to give you more just to get basic functionality that should've been included in the first place.

Fond Farewell

Eventually, one of those open-source programmers—the kind who gleefully release free tools and then deal with endless complaints for life—will create something that does what your product does. It'll have a ridiculous name like CodeSquish, Dojo, or GitCharm.

I'll hear about it from a peer. When I mention I use your product, they'll turn to me, eyes wide, and say, *"Why don't you just use CodeSquish?"*

Not wanting to admit ignorance, I'll make up a reason on the spot. Later, in the bathroom, I'll Google CodeSquish and discover it does everything I need, costs nothing, and is 100x more performant—even though it's maintained by a single recluse who only emerges from their Vermont farm to push code to their self-hosted git repo.

We'll try it out. Despite the fact that its only "forum" is a Discord server, it'll still be miles ahead of your commercial product.

Then comes the breakup. I'll put it off for as long as possible because we probably signed a contract. Eventually, I'll tell Finance not to renew it. Suddenly, I'll get a flurry of attention from your team. You'll pitch me on why the open-source tool is *actually inferior* (which we both know isn't true).

I'll tell you, *"We'll discuss it on our side."* We won't. The only people who cared about your product were me and six others. Finally, like the coward I am, I'll break up with you over email—and then block your domain.

Topic humor tech rant

Share

Review of Orbit by Mozilla

A review and technical tear-down of the Orbit by Mozilla AI summary add-on fo…

06 Jan 2025