DOI:10.1145/3637829

Article development led by acmqueue
queue.acm.org

Access to a system should not imply authority to use it. Enter the principle of complete mediation.

BY PHIL VACHON

The Security Jawbreaker

WHEN SOMEONE STANDS at the front door of your home, what are the steps to let them in? If it is your partner or another member of the family, they use their house key, unlocking the door using the *authority* the key confers: They are authorized to come and go as they please (just lock the door behind you). For others, a knock at the door or doorbell ring prompts you to decide. If this is a friend, acquaintance, or business associate, you can warmly welcome them in. If you are expecting an appliance technician or furniture delivery, you escort them to where you need them to do their work. Maybe law enforcement is at the door, and you can decide what to do based on the credentials they present.

Once in your home, different individuals have differing *authority* based on who they are. Family members have access to your whole home. A close friend can roam around unsupervised, with a high level of trust. A distant acquaintance might rarely be let out of your sight. An appliance repair person is someone you might supervise for the duration of the job to be done. For more sensitive locations in your home, you can lock a few doors, giving you further assurance.

Making these decisions is an implicit form of evaluating *risk tolerance*, or your willingness to accept the chance that something might go against your best interests. It seems unlikely that a close friend would harm you, but there have been enough scary news stories about rogue repair people to engender a low-level distrust of these individuals.[a]

In your home, you install *controls*—mechanisms to protect or safeguard against an adverse outcome—to align with your risk tolerance.[b] A deadbolt lock on your front door and an intruder alarm system can address your "outsider" risk. A lock on your bedroom door or access to the garage ensures the more sensitive—or more dangerous—parts of your home are protected. You might keep your identity and financial documents in a locked safe in a locked office, while your jewelry is in a separate safe in a bedroom. Your partner could have carte blanche access to these, while guests are constrained to specific areas. Untrusted guests might have access to nothing but common areas, while unwelcome visitors are left to cool their heels on your doorstep.

This same mindset extends to companies that use or offer software systems while doing business. Let's

a As unfortunate as this is, the author acknowledges the media fills us with prejudices, thus leading us to act unfairly to others. The author, for example, has never had a bad experience with an appliance repair person.

b An adverse outcome would be a breach of any authenticity, availability, confidentiality, or integrity in your life, as held within the confines of your home. For example, if a visitor accidentally burned down your home, availability of these things would drop precipitously.

explore the evolution of this idea and where improvements could be made.

Step 1: The Tootsie Pop

The problem space for enterprises is not unlike what you are dealing with in your home—but with many facets and at a larger scale. Software developers and security professionals alike had to evolve in their understanding of risks. Many companies have learned (the hard way) that convenience for developers and operations teams must be traded for appropriate access controls to protect customer data or sensitive business information.

In decades past, locks and keys protected computer rooms and datacenters. Dumb terminals were locked in offices—this was enough to protect sensitive data. As the Internet became ubiquitous, customers demanded the convenience of Internet connectedness of services. How do you know, however, that a person accessing a service over the Internet is who they claim to be? The famous line (now a cliché) from Peter Steiner's 1993 cartoon in *The New Yorker* applies: "On the Internet, nobody knows you're a dog." Or Bob from accounting. Or some criminal who is posing as Bob from accounting. When everything was on the Internet, including the internal accounting software running on an ancient mainframe, anyone could assume the authority of Bob. No company has high tolerance for its accounting details being leaked, though.

Out of necessity, companies poured capital into efforts to secure their network edge. This separated sensitive corporate resources and production services running on premises from the Wild West that is the Internet. Resources that were key to offering services over the Internet were all that was exposed. No longer was it as easy as connecting to that accounting server across the Internet. Many companies realized they could extend this secure boundary by exposing sensitive internal resources to employees over a virtual private network (VPN), tunneling an employee from a remote

location through the hard shell.

But what differentiates a user with authority to access a resource from one who does not, once that highly protected exterior is breached? Like a Tootsie Pop, once you break through the hard exterior shell, it's easy to make quick work of the delicious, gooey interior.

Step 2: The Whopper

As network edge controls matured, another problem cropped up: You still end up exposing a lot of applications to the outside world to do business, and many of these applications have elaborate requirements. Ensuring all the protections on the edge are correct is very complex. A simple misconfiguration of an application or firewall rule, a stolen credential, or an unpatched system could become an easy way for a malicious party to get a foothold in your sensitive infrastructure. Putting all your eggs in the network edge as the main control puts the onus on these controls to be perfect, all the time.

With this model, if your edge controls fail, the sensitive interior of your infrastructure could be laid bare for any attacker to access. This also means an employee's credentials being stolen, or a corporate workstation being compromised, is catastrophic: Should an administrative assistant's workstation compromise result in an entire production environment being ransomed? Segmenting resources based on roles and responsibilities starts to make sense: This means moving similar types of controls that you apply at the edge of your network to the various major categories of infrastructure.

It's easy to be lulled into a false sense of security because of strong edge controls and elide away checks for authority. Internal applications, such as your new accounting Web application, should not skip authority checks, but they often do because "only accounts have access to the accounting network segment," right? Even as you segment connectivity between resources, the same problem continues to crop up—one configuration mistake, one missed patch, or one weak credential means an attacker can move into a more sensitive network and wreak havoc. This is a better state to be in than the Tootsie Pop: The attacker now must work harder, compromising more systems to reach sensitive ones. Your network interior becomes crunchier—a texture closer to that of a Whopper.[c]

Step 3: The Jawbreaker

Ideally, authority checks are pervasive, everywhere, every step of the way. Every action, be it the ability to communicate with a service or to perform some action, should be carefully checked against who is making the request and if that person has the authority to make that request. Systems opt-in to allow access for other systems or users. Because every system does its own checks, a misconfigured service in a higher tier will not expose sensitive information: The lower-tier services also make their own authorization checks.

This also implies no shortcuts: The authority of a requester must be

c Like a Malteser, for our colleagues in the Commonwealth.

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK

checked every time they make a request. Shortcuts are opportunities for attackers to reduce the complexity of what they must do to abuse your service. Bringing those checks as close as possible to the sensitive data and resources ensures maximum protection against human error and malice.

The security relationship between services in this case is clear: Other services are kept at arm's length. There is no trust in an authority check being made by another service; each subsequent request has its own authority checks that must be passed. Data dependencies become well-mapped. Operational concerns such as shutdowns, cold starts, and recovery from failures need to be mapped out up-front, forcing a mature resilience posture. After all, a mechanism to drop all authority checks becomes an easy target for attackers, so you must design with these concerns up-front. To an attacker, an environment configured in this way is like trying to chew on a Jawbreaker,[d] requiring significant effort to get to the core.

Such an environment is a long way off for most enterprises and might be unnecessary in many cases. The operational complexities this introduces, as well as the operational risk of grafting authorization systems onto existing environments, mean that this will be a slow evolution toward this very lofty ideal end state. Strategies of mapping out dependencies are key to this evolution because that becomes the baseline for future access policies.

Many enterprises are amid migrations to the public cloud. These migrations are an opportunity to make a quantum leap in authorization tooling: All major cloud service providers make it straightforward to have fine-grained access control built into almost all their service offerings.

Some newer infrastructure offerings—such as Amazon Web Service's virtual private cloud (VPC) Lattice—take this to the next level, removing even the need to think about network connectivity between services, and instead, allowing you to express the relationship between services and systems directly. This is not an excuse to go back to bad habits; each service still needs to make its own authority checks against its callers.

Many companies have learned (the hard way) that convenience for developers and operations teams must be traded for appropriate access controls to protect customer data or sensitive business information.

Take Nothing for Granted

It's very tempting to put all your authorization checks right at the edge of your application as a shortcut. That is the point where the risk of abuse is the highest. A single choke point is attractive, because it is a single point where you must implement controls. Multiple implementations of the same control present a problem to be avoided, after all.

For example, in a traditional three-tier application, a single checkpoint at the front end seems straightforward enough. The middle tier(s) might be able to talk freely to each other, as well as the data stores backing the application. This makes adding new services that use the database easier. Refactoring your application into smaller and smaller microservices becomes a cinch.

Maybe ad-hoc queries against production data are the perfect tool to help quickly troubleshoot an outage, and uptime might be a key performance indicator for your business. But that gooier interior is an easy place for an attacker to gain a toehold and move freely through it.

Security architects and infrastructure builders all wear product manager hats as a part of their roles. Addressing day-two operations challenges is critical to any product's customers. Well-designed processes around day-two operations also make other critical security principles (such as the principle of least privilege) easier to implement.

Beyond the concerns for abuse when a system is out and deployed, a lack of clear responsibility for establishing authority is a common class of security flaws. If an application is expecting another service to perform authority checks, and vice versa, perhaps no authority checks will be made at all. The front end of the application might have unfettered access to sensitive data behind the scenes—a nightmare scenario, since from the outside, it just looks like you had designed things this way. C

d Like a Gobstopper, for our colleagues across the pond.

Phil Vachon leads the information security program at Bloomberg's CTO's office in New York City, NY, USA. Previously, he co-founded and was CTO of a startup that built a high-speed packet capture and analytics platform and worked on spaceborne synthetic aperture radar data processing and applications.