Steven M. Bellovin
Columbia University

Who Are You?

Many security mechanisms, including authorization, depend on the proper handling of identity, but it's hard to get right. To understand why, we must examine, fundamentally, what identity is.

One traditional answer is that an identity is a name. That's unsatisfactory, for several reasons. First, a name points to an object—an entity—but is not itself that object. Second, authentication is needed when dealing with identity. Thus, we must ensure that the binding is correct, that the asserted name actually points to the proper entity. Public-key certificates try to make that linkage explicit, but instead substitute a different indirection: certificates bind a name to a key, but not to the underlying entity.

If an identity isn't a name, then what is it? And if we must use names as proxies for identities, how should we authenticate the binding? As we shall see, the two questions are linked.

The most common way we authenticate bindings is by the assertion of a trusted third party. For certificates, the third party is a certificate authority. Enterprise logins also rely on a third party: the sysadmin who added the login name for a particular person. The enterprise trusts that sysadmin would not add a login for L337Hack3rD00d, although that trust might be backed up by an audit.

Still, a login is not the entity itself. This is seen most clearly with role logins, which can legitimately be used by more than one person. In most enterprises, the "root" or "Administrator" login can be used by anyone in the system administration group. Some logins are associated with services, such as a webserver, rather than a human being. The name-to-identity mapping, then, is not one-to-one; it can be one-to-many, many-to-many, or—if you don't regard webservers as having their own identity—one-to-zero.

Continuity is another common way to authenticate bindings, although it's often (and incorrectly) regarded as a lesser form. With continuity, a binding is presumed correct today because it was correct yesterday. This is sometimes used to validate public keys independent of certificate authorities; it's the only way we can trust generic mail services like Google's or Microsoft's.

I assert that continuity is the answer. Think of it as a timeline. Initial enrollment in a system is the intersection of two timelines, yours and the system's. At this intersection, information—a password, a public key, a biometric—is exchanged. At subsequent contacts, such as login attempts, this shared knowledge is used to confirm each side's identity. Authentication, then, uses shared knowledge to confirm the previous intersection.

Password theft or key compromise can be understood as sufficient knowledge being stolen to allow one party to deceive the other about continuity. The same is true of identity theft. One approach to this problem is to rely on more shared knowledge; the risk is that the attacker also has this knowledge. The solution is thus to rely on things like a known residence or a long-term mobile phone number. It's not that these methods are inherently stronger; rather, it's that they leverage aspects of continuity other than shared knowledge. Similarly, focusing on continuity teaches us that although certificate authorities might work to validate initial contacts, key continuity is a stronger technique for subsequent website visits.

No authentication framework will completely protect us from hacks, but a good one might guide our choice of techniques. ■

Steven M. Bellovin is a professor of computer science at Columbia University. Contact him via https://www.cs.columbia.edu/~smb.

Copublished by the IEEE Computer and Reliability Societies
1540-7993/17/$33.00 © 2017 IEEE