Security for Smart Objects beyond COMSEC: Principals and Principles

Richard Barnes <rbarnes@bbn.com>, BBN Technologies

Security problems are problems of relationships: Who are you, and what do I authorize you to do? Cryptographic security protocols exist so that policies can be applied based on good information. So as we think about frameworks for securing systems of smart objects, we need to consider both the mechanisms for secure interactions (protocols, crypto), as well as the policy framework that these mechanisms need to support.

To a large degree, the former question is not so difficult. Even devices that are "constrained" by modern standards have sufficient computational capacity for most modern ciphers and hash functions. Protocols such as TLS/DTLS and IPsec enable secure communications with minimal overhead beyond that required by the cryptographic primitives [1][2]. With long-lived sessions and fast session resumption, DTLS and IPsec can even provide robust support for intermittently connected devices [3][4].

So the more interesting questions are how to establish the inputs to these protocols -- who the security principals are -- and by what principles the relationships between these principles should be managed.

Consider the paradigm with which we're most familiar today: A user logs connects to a web site over HTTPS [5][6]. The principals here are the user and the domain of the web site. They are both represented, in a one-to-one fashion, by automata, a browser and a web server. The server has a defined identity, with global scope, verifiably attested by an external authority in a digital certificate. The browser verifies this credential and applies the user's policy -- that the connection should only proceed if the credential is valid and represents the desired site. If the user logs into the site, he will again be identified by credentials (a user name and password), but only within the scope of the site. The server will use these credentials and their validity as a basis for deciding what capabilities to grant the user; in a sense, these credentials entirely and indivisibly represent the capabilities of the user.

The first wave of smart objects that are entering peoples' daily lives -- smart televisions, for example -- are already pushing beyond this simple paradigm. These objects and similar trends in the web have led to the development of systems such as OAuth that allow one principal to grant a second principal authorizations with regard to yet a third principal [7]. At the same time, there has been more thinking about deaggregation of capabilities, so that a user might delegate only some capabilities or authorizations to a second party, for example,

granting a smart TV the ability to play movies from a streaming service, but not access the user's credit card information or other preferences.

In some senses then, we have crossed the first important bridge, from conflating identity and authorization toward the decoupling of those two concepts. As this process is currently instantiated, it is mostly "one-hop"; a user delegates authorization to a device or third-party service. One could imagine, however, much more complex interactions through the same mechanism. Imagine, for example, delegating to one device the authorization to delegate capabilities to other devices.

And this point returns us to the level of policy. As a user today, my role is to dictate very simple policies -- "Accept only credentials for bankofamerica.com" -- about the other principals with which I directly interact. As a user of multiple autonomous interconnected devices, my role is more that of an orchestrator, laying out the rules of a miniature society at my command, with internal laws and rules of engagement with other societies.

All of which raises a host of practical questions:
-- How do we manage cryptographic identities for smart objects? If they are to have long-lived keys or certificates, how do we deal with key rollover or revocation?
-- In order to facilitate the engagement of objects in policies, how do we communicate device identities through very limited interfaces? How do we tie cryptographic identities to something that can be interpreted by a human?
-- How does a user of smart objects create a sensible system of rules for how they should interact? Are there sensible defaults? What's the "ceremony" that objects go through to get introduced into an environment?
-- Are all of the answers to the above questions application-specific, or are there some general rules that we can pull out?

[1] TLS <http://tools.ietf.org/html/rfc5246>
[2] IPsec <http://tools.ietf.org/html/rfc4301>
[3] IKE Session Resumption <http://tools.ietf.org/html/rfc5723>
[4] TLS Session Resumption <http://tools.ietf.org/html/rfc5077>
[5] HTTPS <http://tools.ietf.org/html/rfc2818>
[6] Server Identity Checking <http://tools.ietf.org/html/rfc6125>
[7] OAuth <http://tools.ietf.org/wg/oauth>