Potential Security Infrastructure for ECF

Harold Lockhart

January 25, 2023

This document briefly discusses three categories of Security Infrastructure which might be deployed in support of an Electric Court Filing system. They are: Single Signon, Key Distribution and Access Control. The contents of this document represent the professional opinion of the author and not necessarily the ECF TC or any other standards body or other kind of organization. This document does not address the existence or quality of available implementations of the standards mentioned.

**Single Signon**

Most web application systems which restrict access in any way use a single signon system for Authentication and in many cases, Authorization as well. This is because most configurations involve multiple servers, even when the servers performing the same functions are not geographically distributed. If single signon was not used, either each user’s session would have to be processed by the same server or users would have to login repeatedly every time they executed on a different server. Once a single signon protocol is in place, it also permits seamless use of geographically distributed servers as well.

Probably the most widely used such standard is the combination of OpenID and OAuth 2.0 to provide Authentication and coarse grained (application service level) Authorization, which is also known as OpenID Connect (OIDC). The other major alternative to this is SAML, which is an older and more comprehensive Standard. SAML has always supported multiple Identity Providers (Idp). OAuth 2.0 can support multiple Idp’s, but it must be carefully configured to avoid security threats. The most obvious difference between the two is that OAuth uses messages formatted as JSON, whereas SAML messages are formatted in XML. This is undoubtably the reason for the greater popularity of OpenID/OAuth.

**Key Distribution**

ECF makes use of digital signatures to verify the authenticity of court documents. However, ECF does not specify any particular means of securely distributing encryption keys to enable document signing or reliably providing Trust Roots to enable signature verification.

Neither SAML nor OIDC define a key distribution protocol. However, SAML allows queries about attribute values and in general SAML allows all of or portions of messages to be encrypted. This would permit defining a application-specific key distribution protocol to be defined using existing SAML primitives.

Turning to standards, KMIP is the oldest and, perhaps most complete Key Management protocol. However, because it started out mostly aimed at hardware or low-level implementation, it uses numeric coding, perhaps not the best choice for an otherwise XML environment.

WS-Trust is an WS-Security standard which enables the request and issuance of Security Tokens. By defining tokens which correspond to wrapped encryption keys or trusted certificates, key distribution can be implemented. The formats of tokens can be specified by means of WS-Security-Policy. These specifications were developed at WSI and OASIS.

Another alternative is XKMS, which was developed at W3C. Its functionality is specifically related to distributing keys and communicating trust anchors. It uses XML messages.

**Access Control**

If we Ignore simple access control mechanisms like file permissions and ANSI Access Control Lists, then for the last few decades there been mainly two schools of thought: Role-based (RBAC) and Attribute-based (ABAC). (The latter is also sometimes called Policy-based Access Control.) The NIST model which is the best current description of RBAC is ANSI INCITS 359-2012. There are numerous RBAC implementations, but my impression is that they are not interoperable.

The XACML specifications are usually taken as the embodiment of ABAC. NIST once heavily favored RBAC over ABAC, but in the last decade have become more favorable to ABAC as well. My own opinion is that RBAC can be very satisfactory in small or simple environments, where for example, assess is allowed solely based on the person’s job and this is uniform throughout the organization. It also helps if you can dictate the rules to users.

In the real world, users with the same job title may perform very different types of work, organizations also may want to restrict access based on location, time of day and many other factors. XACML makes it much easier to incorporate these requirements whether they are known in advance or not. One symptom of this is the commonly heard lament: we started Role engineering, but it is taking much longer than we thought.

Many people find printed examples of XACML policies to be intimidating, but while the language is quite wordy and not at all abbreviated, it was never the intention that policies be created by typing in raw XML with a text editor. Most XACML implementations include a policy editor or policy creation tool. The ones I have tried make the task much simpler, by proposing legal alternatives and filling in boilerplate items. The critical work, for both RBAC and ABAC is to understand how the new application fits into the current organizations and which functions will be performed by which types of users.