Enforcing Safety and Consistency Constraints in Policy-Based Authorization Systems

Adam J. Lee    Marianne Winslett

Speaker: Brian Wongchaowart
October 13, 2009
Motivation

- When an access control policy is satisfied by providing multiple credentials, it is often the intention of the policy author that all of these credentials must be simultaneously valid.
- Discussion: Are there situations where simultaneous validity is not required?
Motivation

- Simultaneous validity cannot be enforced by checking the issue and expiration times of the submitted credentials alone, since credentials can be revoked prior to their natural expiration.
Impact of Revocation

- In the figure below, credential $c_1$ was issued (or first became effective) at time $\alpha(c_1)$ and expires at time $\omega(c_1)$, while credential $c_2$'s stated validity period is from time $\alpha(c_2)$ to $\omega(c_2)$.
Impact of Revocation

- It is possible, however, that $c_1$ was revoked before time $\alpha(c_2)$.
What Time is $\alpha(c)$?

- In this paper, credential issue and expiration times are interpreted as local timestamps.
- This assumes that the local clock is perfectly synchronized with the credential issuer’s clock.
- In reality, clocks can both differ by an offset and run at different rates (which are not necessarily constant).
Synchronized Clocks
Unsynchronized Clocks

\[ \alpha(c_1) \quad c_1 \quad \omega(c_1) \]

\[ \alpha(c_2) \quad c_2 \quad \omega(c_2) \]
Nota Bene

The algorithms in this paper require perfectly synchronized clocks to be correct as written.
Criteria for Interval of Simultaneous Validity

- *Incremental consistency* means that the submitted credentials were all *individually* valid at the point when they were received.
Criteria for Interval of Simultaneous Validity

- *Internal consistency* means that the submitted credentials were all **simultaneously** valid at some point during the authorization process.

\[
\alpha(c_1) \quad c_1 \quad \omega(c_1) \\
\alpha(c_2) \quad c_2 \quad \omega(c_2)
\]

\[
c_1 \text{ received} \quad c_2 \text{ received}
\]

authorization process
Criteria for Interval of Simultaneous Validity

- *Endpoint consistency* means that the submitted credentials were all simultaneously valid at the end of the authorization process.
Criteria for Interval of Simultaneous Validity

- *Interval consistency* means that the submitted credentials were all simultaneously valid at the end of the authorization process and all credentials were individually valid *when received*. 
The rationale given for interval consistency in this paper is that the requirement that each relevant credential must be valid from the time that it is received “may imply some level of stability in the system” (p. 14).

However, neither endpoint consistency nor interval consistency place a constraint on the length of the interval of simultaneous validity. That is, the period of simultaneous credential validity may be arbitrarily short.

Discussion: Why do we need interval consistency?
Syntactic Validity

• A credential is syntactically valid at time $t$ if it appears to be valid at time $t$ based on its contents alone.
• This means that the credential is properly signed and time $t$ falls within its stated validity period.
A credential is semantically valid at time $t$ if its issuer asserts that it is valid at time $t$ and has been valid since the start of its stated validity period.

This means that the credential issuer is either contacted online or has issued a signed statement containing this information.
Enforcing Simultaneous Credential Validity

- Digital credentials do not provide real time information.
- Only querying the issuer of a credential provides real time validity information.
- Although the algorithms in this paper are credential-based, they are perhaps more accurately described as simple distributed proof protocols.
Algorithm for Enforcing Internal Consistency

- Check the syntactic and semantic validity of each credential as it is received.
- If a credential is still valid, use the time of the validity check as the second endpoint of its effective validity interval and the beginning of its stated validity period as the first endpoint (this assumes clock synchronization).
Algorithm for Enforcing Internal Consistency

- If all of the credential validity intervals computed in this way overlap within the span of the authorization process, **accept**.
What if a credential is received with an issue time later than the time of another credential’s validity check?

$\alpha(c_1) c_1 \omega(c_1)$

$\alpha(c_2) c_2 \omega(c_2)$

$c_1$ checked

$c_2$ checked

authorization process
Algorithm for Enforcing Internal Consistency

- In this case $c_1$’s semantic validity needs to be rechecked.
Incompleteness Due to Credential Revocation

- What if $c_1$ was revoked before being rechecked?

\[ \alpha(c_1) \rightarrow c_1 \text{ checked} \rightarrow c_1 \text{ revoked} \]
\[ \omega(c_1) \rightarrow c_1 \text{ rechecked} \]
\[ \alpha(c_2) \rightarrow c_2 \text{ checked} \]
Incompleteness Due to Credential Revocation

- If the recheck of $c_1$’s semantic validity simply returns false, it may not be possible to determine that the validity intervals of $c_1$ and $c_2$ actually overlap.
This paper proposes an online credential status protocol in which the response contains a nonce chosen by the client in addition to a status of valid or invalid. (This functionality is actually defined as a standard extension in the OCSP RFC from 1999.)
Online Credential Status Protocol

- The client can only obtain $\text{sign}(\text{cred}, \text{nonce}, \text{status})$ after it knows the nonce.
- A principal requesting access to a service can construct a proof that the credentials submitted in support of that request were all still valid at the time of the request by obtaining signed status messages containing a fresh nonce generated by the service.
Application of Online Credential Status Protocol

- Request for service
- nonce
- Credentials
- Signed status messages containing nonce

Discussion: Does this protocol eliminate the need for clock synchronization?
Algorithm for Enforcing Interval Consistency

- Check the syntactic validity of each credential as it is received.
- Check the semantic validity of each credential after all credentials have been received. If a credential is still valid, use the time of the validity check as the second endpoint of its effective validity interval and the beginning of its stated validity period as the first endpoint (this assumes clock synchronization).
Algorithm for Enforcing Interval Consistency

- If all of the credential validity intervals computed in this way overlap at the endpoint of the authorization process, **accept**.
Incompleteness Due to Credential Revocation

- This algorithm may fail when a credential is revoked after the endpoint of the authorization process, but before its semantic validity has been checked.
A Sound and Ideally Complete Solution

- Since the algorithms for both internal and interval consistency rely on clock synchronization to establish the beginning of credential validity intervals, it is also possible to rely on clock synchronization to precisely determine the end of a revoked credential’s validity interval (given support from the online credential status protocol).

![Diagram of client-service interaction with credential signing]
A Sound and Ideally Complete Solution

\[ \alpha(c_1) \]

\[ \alpha(c_2) \]

\[ c_1 \text{ received} \]

\[ c_2 \text{ received} \]

\[ c_2 \text{ checked} \]

\[ c_1 \text{ checked} \]

\[ c_1 \text{ revoked} \]
Subsequent Work

- It is possible to completely eliminate the need for clock synchronization.
- The basic idea is to query for credential validity twice, once to establish the first endpoint of a validity interval, and again for the second.
Strengths

- This paper shows that the paradigms of trust negotiation and distributed proof are in fact closely related. While credentials can be issued once and held for long periods of time, there is no way to prove that a credential was valid in the recent past without a fresh assertion from the credential issuer that the credential has not been revoked.
Strengths

• This paper calls attention to the problem of using unsynchronized clocks when trying to enforce the simultaneous validity of credentials. This immediately allows us to draw on the rich body of work on clocks and causal orderings in the field of distributed systems.
Weaknesses

- If the credentials that allow a user to gain access to a service must be simultaneously valid at the time when the authorization decision is made, isn’t it just as important that the credentials remain simultaneously valid while the service is being used?
- One-time authorization is often only an approximation to the real access control policy of a resource.
Weaknesses

- The algorithms in this paper require perfectly synchronized clocks to operate correctly, since $\alpha(c)$ and $\omega(c)$ are interpreted as local timestamps.
- Questions about simultaneous validity are trivially answered when clocks are perfectly synchronized, however.
Weaknesses

- Credentials may not be revoked instantaneously when their assertions become invalid (continuous monitoring of some predicates may be hard).
- In practice it is difficult to be sure that a set of facts is simultaneously true in the real world based on the evidence of digital credentials.
Summary

- There are two reasons why ensuring the simultaneous validity of credentials is difficult: credential revocation and clock synchronization.
- This paper addresses the first problem, not the second.
- Assuming that an online service exists for checking the current validity status of every credential, this paper proposes algorithms for enforcing the simultaneous validity of credentials at some point during the authorization process and at the endpoint of the authorization process.
Questions/Comments