A rule language for network policies

Jan Nicklisch (jan@ccrle.nec.de)
C&C Network Product Development Laboratories
NEC Europe Ltd.
Hardenbergplatz 2
10623 Berlin, Germany

Addressed Topics

• Notations for specification of policy
• Policy implementation issues and techniques
• Tool support

Abstract

This position paper outlines the design and implementation of a multi-purpose policy definition language, referred to as PFDL. The language is used in a prototypical policy server to implement the centralized view of a network administrator with regard to signaled QoS provisioning via RSVP, access control based on the recognition of TCP/IP traffic flows, and customized treatment of Web applications.

Introduction

This work is based on the IETF model of network policy provisioning and enforcement, which is described in the internet drafts of the “Policy Framework” working group [1].

In short, the framework consist of “policy enforcement points” (PEP, the network elements), a “policy decision point” (PDP, typically referred to as the policy server), a repository to store policies, user and network resource information, and a management console. The single components are linked by the following protocols:

• The Common Open Policy Service (COPS) is used to forward requests from the PEPs to the central PDP and to pass back corresponding policy decisions. The COPS protocol supports synchronization of policy decisions and aims at reliability using keep-alive messages.
• A policy definition language (e.g. the Policy Framework Definition Language, PFDL) is used to define new policies in terms of policy rules with condition and action lists.
• The simple version of the X.500 directory access protocol called LDAP is used to retrieve information from the repository from the policy server.
Background

Our work is particularly related to a multi-purpose policy definition language and concentrates on providing a prototype implementation to gain practical experience.

Such definition language is of major interest since it is the least agreed-on and the most flexible part of the IETF policy framework. In the current view, the PFDL is the network administrator's tool language to express various kinds of network policies. This can include security policies, policies related to signaled and/or provisioned QoS, access restriction policies, and so on.

However, the IETF policy group has decided to suspend pursuing work on the policy definition language, since it turned out that there would not be consensus within the available time. The current framework focuses on QoS provisioning policies only, which shall be validated within the “Differentiated Services” framework.

We therefore had to base our work on the draft of November 1998 [2], which outlines a possible policy rule language to fit into the IETF framework. Unfortunately, this early draft was neither fully consistent nor did it provide enough details to answer most of the immediately arising questions related to it.

As a first step, we did hence extend the suggested language with detailed derivations, such that one could start an implementation. Second, we extended the language and the prototype by an application specific domain (HTTP related policies) in order to validate the flexibility of the framework and the implementation. We still call the resulting language PFDL, but one should keep in mind the difference between the original draft language and the version used in our implementation.

The Policy Framework Definition Language PFDL

The PFDL can simply express lists of IF <condition> THEN <action> type of rules. Any such list forms a policy. <condition> above is in fact a disjunctive normal form of single condition expressions, and <action> is a list of single action statements. The semantics of such rules is clear: if the evaluation of the condition expression with the current state of the policy server (policy decision point) and the specific client (policy enforcement point) request succeeds, the action list can be performed.

Example policy rules are:

1. If anyone accesses the draft archive at IETF, rather retrieve the draft from our local mirror.

   IF HTTP_URL_HOST   EQ  www.ietf.org       AND
   HTTP_URL_PATH      EQ  "/internet-drafts/*"  THEN
   HTTP_URL_HOST      REPLACE  "chiba.ccrle.nec.de"  AND
   HTTP_URL_PATH      REPLACE  "/Mirror/ftp.ietf.org/internet-drafts"

2. If any of the named users below tries to listen to real audio streams during working hours, then deny the service.

   IF USER            IN  {"peter", "frank", "mary"}  AND
   TIME              IN  Mon - Fri                  AND
   TIME              IN  09:00 - 17:00             AND
   HTTP_URL_PATH     ENDS  ".ra"                    THEN
   DENY

3. Users of an RSVP featured video conferencing application (vic) from within a subgroup of machines can reserve up to 500 kByte for the video stream.
IF APPL EQ "vic" AND SENDER IN 192.168.100.44 - 192.168.100.112 AND RSVP_IS_BANDWIDTH LE 500 AND THEN PERMIT

Generally speaking, the language can expresses the following condition domains:

- **Time domain**: conditions limiting the applicability of a policy rule to a certain time period.
- **RSVP domain**: conditions related to signaled QoS provision via RSVP, i.e. message type (PATH, RESV), flow identity, bandwidth etc.
- **User domain**: conditions related to user identity and user groups.
- **Application domain**: conditions related to TCP applications based on application name or port number.
- **Address domain**: conditions specifying the sender or receiver of TCP/IP flows.
- **COPS domain**: conditions related to the COPS in/out interface and the request type.
- **HTTP domain**: conditions related to tasks that a WWW proxy could perform, e.g. denial of specific services and mirroring.

**Implementation**

All parts of the prototype are implemented in Java, i.e. the PEP clients, the COPS protocol parts, the rule evaluation engine, the parser and GUI. Additionally, we are assessing a version of the evaluation engine written in C++. We used JavaCC and JTB (Java tree builder) for compiler construction and to manipulate the abstract syntax tree. A graphical policy editor is used to manipulate the policies in the PDP. The policies are stored to (and retrieved from) an LDAP directory implementing the scheme described in [3].

We used a C implementation to link the PEP software to the RSVP daemon [4]. As a test application we chose the video conferencing tool “vic”.

**Discussion Points**

Our prototype allows for experimentation with the policy framework being worked on at the IETF. We can assess whether a multi-purpose rule language for the definition of various network policies can be practical, and how easy it may be extended.

Since rules are simply composed of domain-specific condition expressions and corresponding action statements, it is relatively straightforward to cover a new domain type. In practice one would therefore only need to standardize a new client type and afterwards extend the framework implementation.

An interesting question is how to automatically test the consistency of an operator-supplied rule set.

Another way of looking at the flexibility of the policy framework is by investigating the use of procedural action statements. Currently, the actions are declarative, and therefore have to comply to a previously defined, fixed policy domain. In view of an open programming interface to the policy framework, one would perhaps consider a generic action language to be interpreted at the policy enforcement points.

**References**


   draft-ietf-policy-framework-pfdl-00.txt

   draft-ietf-policy-core-schema-03.txt