Multi-Constraint Security Policies for Delegated Firewall Administration

Cássio Ditzel Kropiwiec[1], Edgard Jamhour[2], Manoel C. Penna[2], Guy Pujolle[1]

[1] LIP6, UPMC, 104 Avenue du Président Kennedy
75016 Paris, France
[2] PPGIA, PUCPR, Rua Imaculada Conceição, 1155, 80215-901,Curitiba, Brazil

Abstract. This work presents a new policy based security framework that is able handle simultaneously and coherently mandatory, discretionary and security property policies. One important aspect of the proposed framework is that each dimension of the security policies can be managed independently, allowing people playing different roles in an organization to define security policies without violating a global security goal. The framework creates an abstract layer that permits to define security policies independently of how they will be enforced. For example, the mandatory and security property polices could be assigned to the risk management staff while the discretionary policies could be delegated among the several departments in the organization.

1 Introduction

In large networks, using a collection of firewalls increases the network security by separating public and private resources. It also permits to control the access of internal users to internal resources, reducing the risks of attacks originated from the inside of the network [1]. However, many difficulties arise when configuring large networks. First, it is necessary to determine the rule set that must be applied to each firewall, in a way that the overall security policy is satisfied [2]. Also, as firewalls of diverse models and vendors can be present, it is necessary to consider the specific set of rules that can be interpreted and enforced by each firewall in the network before applying the configuration. The topology of the network and the placement of the firewalls with respect to the users and resources is another aspect that must be considered. Each firewall receives a rule set according to its location in the network. If more than one firewall is present between a user and a resource, the rule set can be combined in order to better explore the distinct features offered by the firewalls. Ideally, the process of defining security policies should be decoupled from the mechanisms that will actually enforce them over the network. In most organizations, security policies are related to business goals, and are not anymore a purely technical issue.

In order to address the aforementioned issues, this paper proposes a policy-based security framework that introduces a new approach related to the security policy definition and the generation of firewall configuration in a distributed environment. The framework adopts a policy model with three dimensions of security policies: mandatory, discretionary and security property. Mandatory policies are coarse grained and reflect the inviolable security restrictions in the organization. Discretionary policies are fine grained, and are subjected to the mandatory policies. While mandatory and discretionary policies are restrictions imposed to the right of access from users to resources, the security property policies are restrictions imposed to the paths connecting users to resources. In our framework a path must satisfy some security requirements in order to be allowed. This permits to create policies which are independent on the user or resource location. The motivation for this division is to support the cooperation of multiple security staff in the security policy definition. For example, the right to define mandatory and security property policies could be assigned to an organizational-level risk management staff while the discretionary policies could be delegated to the local administrators in several departments in the organization.

The process of translating the three-dimensional high level security policy into firewall configuration is highly complex. In order that the firewall configuration respects the high level definition, we have formalized both the policy model and the translating algorithm using the Z-language notation. The proof of some important theorems permits to demonstrate the coherence of our approach with respect of combining multiple policies.

The remaining of this paper is organized as follows: section 2 presents some representative related works. Section 3 describes our proposed framework, presenting both the security policy language and the translation algorithm. Section 4 presents the Z-language representation of the framework and the theorem proving. Section 5 presents a case study, illustrating how the framework works. Finally, Section 6 concludes the paper and points to future developments.

2 Related Work

Presently, it is possible to find numerous academic or commercial firewall languages proposed to simplify the firewall configuration process. These languages can be classified according to criteria such as vendor independence, topology independency and their level of abstraction.

Firewall languages are vendor dependent when they apply only to firewall devices of a specific vendor. Some examples are the Cisco PIX [3] e Cisco IOS [4] languages. On the other hand, vendor independent languages are not limited to a specific vendor. This is the case of INSPECT language patented by CheckPoint [5]. A vendor can create a firewall supporting the INSPECT standard by implementing a compiler that translates the INSPECT language into the firewall's native configuration instructions.

Languages are topology dependent when they were designed to represent the configuration of each firewall isolated, i.e., the placement of the users and resources with respect to the firewalls is taken into account by the network administrator and not by the language compiler. The language used by the framework presented in [6] is an example of topology dependent language. The framework represents firewall configuration as high level policies based on the Ponder specification [7]. Even though the high level language provides the use of symbols for masquerading host and network addresses, it is still topology dependent, because there is no automated strategy for selecting the sub-set of rules that applies for a specific firewall.

The languages are independent of topology when it is possible to represent both the security rules and the network topology independently. In this case, rules are not specific to each firewall. There is a mechanism or algorithm that evaluates the network topology (i.e., the placement of users and resources with respect to the firewalls) and translates the security policies into localized firewall configuration. The framework described in [8] is an example of topology independent firewall configuration. In [8] the access control policies are defined in three levels: organizational, global and local. Policies at the organizational level are described in natural language, and define security goals such as blocking offensive content and scanning actions. Organizational policies are transformed into global filtering rules at the global level. The subset of the global rules that concerns each firewall is separated and distributed at the local level.

The languages employed for firewall configuration can be further classified according to its level of abstraction. In low-level languages, the network configuration is represented by a set of rules of type “if conditions are satisfied than enforce actions”. The conditions are basically described in terms of the packet’s header fields. Most languages found in the literature, such as the one employed by the Firmato toolkit, are low-level [12]. On the other hand, the high-level languages uses a more abstract concept, wherein the security policies says “what must be done” instead of saying “how must be done”, i.e., the policy define an intention independently of the mechanisms used to implement it. Some examples of policy based languages can be found in [9], [10] and [11]. The framework presented in [9] permits to represent high-level policies in the form of a list of data access rules (DACL), that declares permissions of executing simple operations (read or write) on objects. The framework translates the high-level policies into low-level policies suitable to be configured into the firewall devices. An algorithm for checking the fidelity of the low-level policies with respect to the high-level policies is also presented. The project presented in [10] aims to automate the management of security policy in dynamic networks. The central component is a policy engine with templates of the network elements and services that validates the policy and generates the new security configurations for the network elements when the security is violated. The work presented in [11] abstracts hosts and area addresses by using names, which permits to easily determine which firewalls are traversed by the communication flows. It defines an algorithm that, given a specific topology, creates the filter set for each firewall or router. It also defines a second algorithm that verifies if the resulting configuration violates any of access policies.

The work described in [14] adopts a graphic representation of security rules. The work also defines the concept of security goals (e.g., top secret, mission critic, etc), which impose additional security properties that are required in order to access an object or perform a given access mode. At a lower level, a security goal is expressed in terms of a security requirement vector, which defines the minimum levels of properties such as confidentiality, integrity, availability and accountability. A security assumption vector defines the same properties assigned for the principal and elements along the path between the principal and the resource. In order of an access to be granted, it is necessary that all properties of the security assumption vector satisfy the corresponding properties in the security requirement vector. We have borrowed many concepts related to the security property model from this work.

3 The Framework

This work presents a new policy based security framework that is capable to handle simultaneously and coherently mandatory, discretionary and security property policies. The framework supports the definition of network security configuration for systems formed by a set of users willing to access a set of protected resources. A resource is a service delivered at a location, which can play the role of the source or the destination of an access. A source location is the place where a user initiates an access while a destination location is the place where one or more services are delivered.

A user can access a resource if there is permission. Permissions are represented by three different security models: mandatory, discretionary and security property. The mandatory model defines permissions by classifying users and resources with clearances and classifications. In the mandatory model, a permission is defined whenever a user classification is greater then a resource clearance [13]. The discretionary model defines permissions by mean of rules that relate users to resources, including their possible sources and destinations. Finally, the security property model defines permissions by assigning security levels to firewalls, and locations.

The framework has two main components: an information model and a refinement algorithm. The first includes all high level security information, whereas the second allows security policy formulated by high level statements to be consistently translated to firewall security rules.

3.1 The Information Model

The Information model is organized in five main blocks: the Inventory, the Mandatory Model, the Discretionary Model, the Security Property Model, and the Firewall Features Model, and is depicted in Figure 1.

The Inventory contains the objects and relationships necessary to build the security policy. It is organized in two main objects groups. The first includes users, locations, resources and services. A Service is modeled by the combination of a protocol and two port numbers for the source and destination sides. For instance, the Telnet service would be defined as TCP with destination port 23 and any source port. Although users interact with services, permissions are granted to resources. A Resource consists of one or more services delivered from one or more locations. For example, the E-mail resource could be defined to represent SMTP, POP and IMAP services. A Location is the place from where users access resources and also the place where a resource is located. Physically it corresponds to a host or subnet, and is represented by an IP address and a mask. When assigned to users, a location plays the source role, and the destination role when assigned to resources. The User Located At and Resource Located At classes indicate, respectively, the locations from where a user can initiate an access or from where a resource can deliver a service. Users, locations and resources can be organized in groups, what is modeled by a corresponding abstract class. For example, an Abstract User can be a User or a User Group, which in its turn can contains many abstract users, that is, many users or user groups.

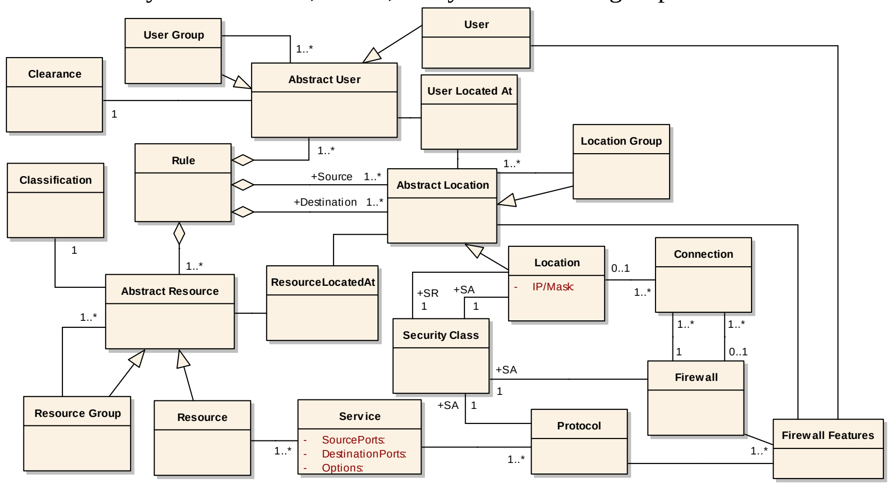

Fig. 1. The Information Model

The second object group includes connections, locations and firewalls. The Connection class represents connections between locations and firewalls or between two firewalls. Together, they model the network topology. The Firewall Feature class models firewall functionality, that is, its ability to perform some processing over the network packets.

The Mandatory Model establishes the mandatory access control policy by determining resource access according to a clearance versus classification schema. Clearance levels are assigned to users wherever classification levels are assigned to resources. A user is allowed to access a resource only if its clearance is equal to or greater than the corresponding resource classification.

The Discretionary Model is constructed by a set of discretionary rules that state the security actions to be enforced for specific service accesses. A service access includes the service, a user, and the location from where he can initiate an access; and a resource, and the destination from where it can be accessed. Examples of security actions are: accept, deny, log, and forward. In this study we just consider the accept action, and adopt the "anything not explicitly allowed is forbidden" strategy.

The Security Property Model provides fine-grained security information by including configuration and location dependent constraints. Security property rules enforce access control based on two properties, the security requirement (SR), which is defined for destination locations, and security assumption (SA), which is defined for source locations, firewalls and protocols.

Security requirements and security assumptions are specified by security levels within a security class. The security level is a natural number ranging from one to four, expressing the "strength" of one of the following security properties: confidentiality, integrity, availability and accountability. The security level establishes a total order over the security property set: the greater is the corresponding number (1, 2, 3 or 4) the stronger is security level. A security class is defined by an array of size four. If *sc* is a security class, *sc*[1], *sc*[2], *sc*[3] and *sc*[4] correspond, respectively, to its confidentiality, traceability, integrity and accountability security levels. Any entities that can be involved in resource accesses (i.e., locations, firewalls, and protocols) have a security class.

Security requirements and security assumptions can change when the related objects are combined. For example, John Doe trying to access a resource from the Engineering subnet would probably have a different SA than when he is trying to access the same resource from JD-Home host. Assuming that the corporation has much more control over the Engineering subnet then the first combination should result in a stronger SA. The protocol used by the object also changes its SA. For example, John Doe at Engineering subnet accessing a resource through HTTPS protocol introduces lower risk than when he is trying to access same resource from the same subnet through HTTP. Consequently a stronger SA should be assigned to the first. The modified security assumption is referred as effective security assumption (ESA). The upper effective class operation ($\bigcup$) over the security class set is defined to compute it. Let sc_1, sc_2, …, sc_n, to be security classes such that $sc_i = [x_{1i}, x_{2i}, x_{3i}, x_{4i}]$.

$$\bigcup_{i=1}^{n} sc_i = \left[\sup_i(x_{1i}), \sup_i(x_{2i}), \sup_i(x_{3i}), \sup_i(x_{4i})\right] \qquad \textbf{(1)}$$

Security assumptions along an end-to-end path are combined together to form the overall security assumption (OSA). The permission to a user (from a source location) willing access a resource (at a destination location) is granted only if the end-to-end path OSA is at least as "strong" as the destination location SR. This involves the comparison of security classes. Because there is a partial order over the security class set, we are able to define its "strength". When sc_1 and sc_2 are security classes such that $sc_1 = [x_1, x_2, x_3, x_4]$ and $sc_2 = [y_1, y_2, y_3, y_4]$, the security class sc_2 is stronger than sc_1 if $y_i \geq x_i$, for $i = 1..4$.

The OSA is calculated as follows: First, compute the ESA for the (source location, protocol) pair and for each (firewall, protocol) pair along the path. Then, the resulting ESAs are combined along the end-to-end path. In this case the calculation should retain the set of weakest security levels. For this, the lower effective class operation (∩) is defined over the security class set as follows. Let sc_1, sc_2, …, sc_n, to be security classes such that $sc_i = [x_{1i}, x_{2i}, x_{3i}, x_{4i}]$.

$$\bigcap_{i=1}^{n} sc_i = \left[\inf_i(x_{1i}), \inf_i(x_{2i}), \inf_i(x_{3i}), \inf_i(x_{4i})\right] \tag{2}$$

The Firewall Features Model contains the objects and relationships necessary to represent the firewall security functionality. The Firewall Feature class models the firewall ability to perform some processing over the network packets. For instance, a stateful firewall has the ability (or feature) of keeping track the state of network connections (such as TCP streams or UDP communication) flowing across it. This concept was introduced in Firmato [12] where the following features, included in the most common firewalls are listed: Names, Groups, IP Ranges, Stateful, Trust levels, Directional, Default Stance, Predefined Services and Layer.

A service access introduces the need for firewall features along the end-to-end path. For example, if the Email-plus resource represents a service that allows the exchange of e-mails with attached videos, then the Predefined Services feature must be present. The required features (RF) operation computes the set of all firewall features that are necessary for a service access. On the other hand, the feature can be supported by any of the firewalls along the path. In other words, the sequence of firewalls within a path supports the union of individual features. The virtual firewall (VF) operation computes the set of features supported along the path. If A = RF(service access) and B = VF(path), the service access is feasible if $A \subseteq B$.

A service access determines a firewall rule, that is, a sequence of conditions that must be satisfied to allow the access. We introduce the concept of firewall abstract rule, a vendor independent syntax firewall rule, that is, a sequence of abstract (vendor independent) conditions and the corresponding action. To build this abstract rule, one should consider the objects included in the service access, obtaining, for each one, the conditions registered in the inventory. For example, if John Doe is named in the service access and its login name in the inventory is JDoe, then the corresponding abstract condition is (login_name, JDoe). Also, if the Engineering subnet is named in the service access and its IP address is 10.1.1.0/24, the corresponding abstract condition is (IP_SRC, 10.1.1.0/24). Each abstract rule must be distributed along the firewalls involved in a service access. The features supported by each firewall are registered in the inventory. Please note that the RF operation assures that all necessary conditions can be enforced by the firewalls along the path.

3.2 The Refinement Algorithm

The refinement algorithm is presented in the following, supported by the example depicted in figure 2.

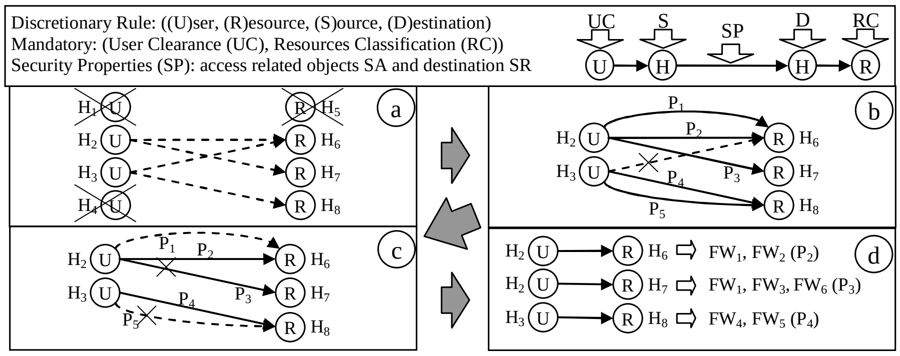
Discretionary Rule: ((U)ser, (R)esource, (S)ource, (D)estination)
Mandatory: (User Clearance (UC), Resources Classification (RC))
Security Properties (SP): access related objects SA and destination SR
UC
S
SP
D
RC
U
H
H
R
a
b
c
d
H1 U
R H5
H2 U
H3 U
H4 U
R H6
R H7
R H8
P1
P2
P3
P4
P5
H2 U → R H6 ⇨ FW1, FW2 (P2)
H2 U → R H7 ⇨ FW1, FW3, FW6 (P3)
H3 U → R H8 ⇨ FW4, FW5 (P4)

Fig. 2. A Refinement Example

1. Identify the set of (user, resource) pairs that hold the discretionary and mandatory models: (i) Take all abstract users and resources that are referenced by discretionary rules. (ii) Expand groups to individual users and resources. (iii) Select all (user, resource) pairs for which user clearance is greater than or equal to resource classification.

2. For the (user, resource) pairs obtained in step 1, compute the set of service accesses, that is, ((user, source), (destination, resource), service) tuples, that hold the inventory and discretionary model (Figure 2a). The source and destination locations must be referred by a discretionary rule and be registered in User Located at or Resource Located at, respectively. Each resource is related to one or more services. If the set is empty, terminate the algorithm.

3. Compute the set of possible paths for the service access computed in step 2 (Figure 2b): (i) For each service access, find a candidate path. (ii) Identify the possible protocols for the candidate path, from the services delivered at the corresponding destination. (iii) For each candidate path, compute the corresponding OSA. If it is at least as stronger as the destination SR, include it in the set of possible paths. (iv) If the set of possible paths is empty for all services accesses, terminate the algorithm.

4. Compute the set of feasible paths for the service accesses that corresponds to a possible path computed in step 3 (Figure 2c): (i) For each service access, compute A = RF(service access). (ii) For each path in the set of possible path, compute B = VF(path). (iii) If $A \subseteq B$, include the path in the set of feasible paths. (iv) If the set of feasible paths is empty for all services accesses, terminate the algorithm.

5. Compute the set of abstract firewall rules for each service access that corresponds to a feasible path computed in step 4 (Figure 2d): (i) For each firewall, compute the set of service access in which it is involved. (ii) For each service access in which the firewall is involved, produces an abstract rule with the conditions it can implement.

4 Formal Representation, Analysis and Validation

The refinement algorithm must guarantee that translation process doesn't cause the violation of the policies. High-level and low-level policies must represent the same set of permissions; otherwise, the whole system can be compromised. Two main theorems must be demonstrated to validate the algorithm:

1. Every access allowed by the higher level policy should be supported by the lower level policy (if they can be correctly enforced), and
2. No action allowed by the lower level policy should be forbidden by the higher level policy.

The formalism used in this work for formal validation and analysis is based on Z notation [15]. The Z notation is a formal specification language used for describing and modeling computing systems. The Z/EVES tool [16] is used to aid in representation and manipulation of Z notation. It is an interactive system for composing, checking, and analyzing Z specifications.

The validation approach used in this work consists of representing the algorithm in Z notation, creating theorems that represents the properties that the system must hold and using the Z-Eves tool to automatically prove these theorems, thus validating that the mathematical representation of the algorithm is consistent and complete.

The complete Z specification of the system and the proved theorems is very extensive and complex to be entirely presented in this paper. However, to illustrate how it is done, we present some excerpts in the following. The full Z specification is available to download at [17]. As an example, Figure 3 presents the procedure for handling discretionary rules.

```
DiscretionarySchema
  $Rule: \mathbb{P}\ User \times \mathbb{P}\ Resource \times \mathbb{P}\ Location \times \mathbb{P}\ Location \times Action$
  ----------------
  $Rule = (users, resources, sources, destinations, action)$
  if $\exists user: users \bullet user = AnyUser$
  then $RuleUsers = Users$
  else if $users \subseteq Users$ then $RuleUsers = users$ else $RuleUsers = \emptyset$
  if $\exists resource: resources \bullet resource = AnyResource$
  then $RuleResources = Resources$
  else if $resources \subseteq Resources$
      then $RuleResources = resources$
      else $RuleResources = \emptyset$
  if $\exists source: sources \bullet source = AnySource$
  then $RuleSourcesExplicit = Locations$
  else if $sources \subseteq Locations$
      then $RuleSourcesExplicit = sources$
      else $RuleSourcesExplicit = \emptyset$
  $RuleSourcesLocatedAt$
    $= \{ u: User; l: Location \mid u \in users \land (u, l) \in UserLocatedAt \bullet l \}$
```

```
18  RuleSources = RuleSourcesExplicit ∩ RuleSourcesLocatedAt
19  if ∃destination: destinations • destination = AnyDestination
20  then RuleDestinationsExplicit = Locations
21  else if destinations ⊆ Locations
22      then RuleDestinationsExplicit = destinations
23      else RuleDestinationsExplicit = ∅
24  RuleDestinationsLocatedAt
25    = { r: Resource; l: Location | (r, l) ∈ ResourceLocatedAt • l }
26  RuleDestinations = RuleDestinationsExplicit ∩ RuleDestinationsLocatedAt
```

Fig. 3. Line 2 defines the structure of the rule. Lines 3 to 5 select the users referenced by the rule in RuleUsers set. If AnyUser is present in the rule then the RuleUsers set must contain all the users registered in the system. Otherwise, the specification checks if the specified users are registered in the system and makes the RuleUsers set to include them if true. If the two previous verifications are false, then RuleUsers set is empty, meaning that the rule is not valid for any user. Lines 6 to 10 state the same logic for RuleResources set. Lines 11 to 18 specify how RuleSources set is built. Note that lines 11 to 15 are similar to RuleUsers set specification. The differences are at lines 16, 17 and 18. In the first two, the sources where the users can be located are selected, while in line 18, the RuleSources set is defined as the intersection between the sources specified in the rule and the locations of the users. The same logic is applied to RuleDestinations set at lines 19 to 26.

With the algorithm modeled in Z, the next step is to specify the theorems and to prove them. The two main theorems previously cited in this section were divided into several small theorems, in order to make the demonstration process simpler. We present some of these theorems in the following paragraphs.

Theorem 1. "If a rule specifies a user and a resource and if the user clearance is less than the resource classification, then the pair (user, resource) can not be a member of the *UsersAndResources* set". The Z-Eves code is presented in Figure 4.

Theorem 2. "For any protocol, source and destination allowed by the rule, if the OSA is smaller than SR, then the tuple *((protocol, source, destination), firewalls)* cannot be a member of *SecurePaths* set". The Z-Eves code is presented in Figure 5.

Theorem 3. "If the ((user, resource), (protocol, source, destination), firewalls) tuple defines a service access across the firewalls (i.e., if the tuple is a member of the *Permissions* set), then the rule must include these user, resource, source and destination; the user clearance must be equal to or greater than the resource classification; and the OSA for the *(protocol, source, firewall)* tuple must be equal to or greater than the destination SR". The Z-Eves code is presented in Figure 6.

```
theorem rule testMandatory
  MandatorySchema
  ∧ user ∈ RuleUsers  ∧ resource ∈ RuleResources
  ∧ Clearance user < Classification resource
  ⇒ ¬ (user, resource) ∈ UsersAndResources
```

Fig. 4. Z representation of Theorem 1

theorem rule *testSecurityProperties*
SecurityPropertiesSchema
$\wedge\ protocol \in Protocol\ \wedge\ source \in Location\ \wedge\ destination \in Location$
$\wedge\ firewalls \in \mathrm{seq}\ Firewall$
$\wedge\ ((protocol, source, destination), firewalls) \in RulePathsAndProtocols$
$\wedge\ \neg\ (OSA\ (protocol, source, firewalls), SR\ destination) \in GOSA$
$\Rightarrow \neg\ ((protocol, source, destination), firewalls) \in SecurePaths$

Fig. 5. Z representation of Theorem 2 - The *GOSA* set represents a relation between OSA and SR, and their elements are those for which OSA is equal to or greater than SR. Thus, the tuples (OSA, SR) that are not members of the GOSA set are those for which OSA is smaller than SR.

theorem rule *testPermissions*
PermissionsSchema
$\wedge\ ((user, resource), (protocol, source, destination), firewalls) \in Permissions$
$\Rightarrow user \in RuleUsers \wedge resource \in RuleResources$
$\wedge\ Clearance\ user \geqslant Classification\ resource$
$\wedge\ ((protocol, source, destination), firewalls) \in RulePathsAndProtocols$
$\wedge\ (OSA\ (protocol, source, firewalls), SR\ destination) \in GOSA$

Fig. 6. Z representation of theorem 3 - If some permission is a member of the *Permissions* set, then it must be present at discretionary, mandatory and security property policies.

5 Example

According to our approach, only the framework has the credentials necessary to create rules in the firewalls. The policy administrators need to use the framework in order to manage the security policies. To illustrate the use of the framework, consider the example in Figure 7. It supposes an imaginary university network (yet realistic), with two firewalls separating 4 networks. The example illustrate how the mandatory and security property policies constraints the discretionary policies, avoiding violation of global security rules. For sake of simplicity, the security assumption and requirement vectors have been reduced to two dimensions: [confidentiality, traceability].

Suppose that the mandatory and security property policies have been previously defined (as presented in Figure 7) by a specialized department in the university, responsible for the overall security. Now suppose that an administrator responsible for creating discretionary policy decide to give full access permissions for all users with a discretionary rule such as: "*Any User from Any Location may Access Any Resource at Any Location*". In spite of this rule the discretionary rules generated by the framework would be defined as follows.

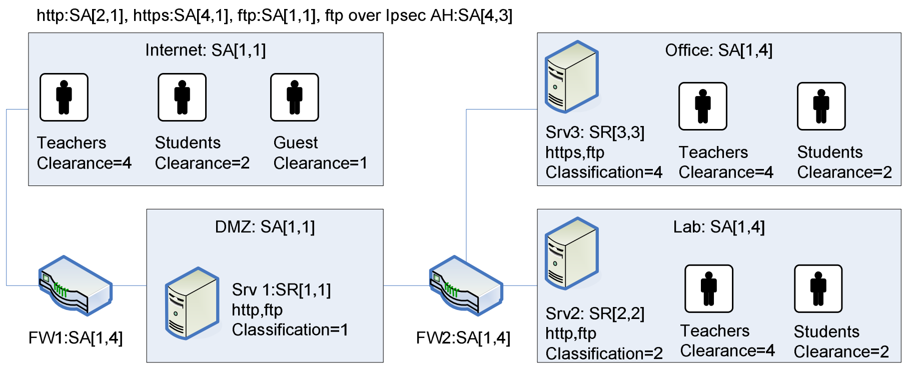

Fig. 7. Teachers and students can be located at Internet, Office or Lab network, and Guest users can only be located at Internet. Teachers have mandatory level (clearance 4) necessary to access resources http and ftp at Srv1 and Srv2, https and ftp at Srv3 (classifications 1, 2 and 3, respectively). Students have mandatory level (clearance 2) to access resources http and ftp at Srv1 and Srv2 (classifications 1 and 2, respectively). Guests users have mandatory level (clearance 1) to access resources http and ftp at Srv1 (classification 1). Considering the security property rules, https at Srv3 can be accessed from Office and Lab networks, and ftp can be accessed from any network if using IPsec AH. Srv2 can be accessed from Office network with http protocol. Srv1 can be accessed from any network (including Internet) using any protocol.

The rules for Firewall1 are:

- Permit Teachers, Students and Guests from Internet to access http and ftp services at Srv1 – because the clearance of these users are greater than the classification of resources at Srv1 and the OSA of the path [Internet, FW1] is equal to SR of Srv1.
- Permit Teachers from Internet to access ftp over IPsec AH at Srv3 – in this case, the OSA of the path [Internet, FW1, DMZ, FW2] combined with ftp over IPsec AH results in [4,3] that is greater than the SR [3,3] of Srv3 (the OSA were obtained from the combination of individual ESA of network elements, that for this situation are the following: Internet: [4,3], FW1: [4,4], DMZ: [4,3], FW2: [4,4]), but only Teachers have clearance (4) greater or equal to classification of resources at Srv3.

The rules for Firewall2 are:

- Permit that Teachers and Students access the http and ftp services at Srv1 from Office and Lab – both Teachers and Students have clearance greater than the classification of resources at Srv1, and the OSA of the combination these paths and protocols are greater than the SR [1,1] of Srv1.
- Permit Teachers and Students from Office to have access to http services at Srv2.
- Permit Teachers from Lab to have access to the https or ftp over IPsec AH services at Srv3.
- Permit Teachers from Internet to have access to the ftp over IPsec AH service at Srv3 – for the same reason of the second rule of Firewall1.

6 Conclusion

This paper has presented a framework capable of handling a multi-constraint security policy model. The security policy permits to create discretionary rules which are constrained by mandatory and security property policies. This is a very flexible approach that permits to describe a large number of discretionary rules without violating the primary security goals in a corporate environment. The motivation for this division is to support the cooperation of multiple security staff in the security policy definition. The policy model has been formalized and validated using the Z-notation and the Z-Eves tool. There are, however, many aspects to be considered in future studies. The methodology “anything not explicitly allowed is forbidden” should be replaced by a more flexible approach capable of supporting negative policies. Also, although a prototype has already been developed in Prolog, a broader scalability study is necessary.

References

1. Markham, T., Payne, C.: Security at the Network Edge: A Distributed Firewall Architecture. DARPA Information Survivability Conference and Exposition (DISCEX II'01), pp. 279, vol. I, (2001)
2. Al-Shaer, E., Hamed, H.: Discovery of Policy Anomalies in Distributed Firewalls. In: 23rd Conference of the IEEE Communications Society (INFOCOMM), pp. 2605-2616 (2004)
3. Cisco Systems Inc.: Cisco PIX Firewall Command Reference. Available at: http://www.cisco.com , (2004)
4. Cisco Systems Inc.: Cisco IOS Reference Guide. Available at: http://www.cisco.com, (2004)
5. CheckPoint Software Technologies Ltd.: Stateful Inspection Technology. Available at: http://www.checkpoint.com/products, (2005)
6. Lee, T.K., Yusuf, S., Luk, W., Sloman, M., Lupu, E., Dulay, N.: Compiling Policy Descriptions into Reconfigurable Firewall Processors. In: 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 39-48, (2003)
7. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In: Policy 2001: Workshop on Policies for Distributed Systems and Networks, pp. 18-39, (2001)
8. Haixin, D., Jianping, W., Xing, L.: Policy-Based Access Control Framework for Large Networks. In: Eighth IEEE International Conference on Networks, pp. 267-273, (2000)
9. Ou, X., Govindavajhala, S., Appel, A. W.: Network security management with high-level security policies”. Technical report TR-714-04, Computer Science Dept, Princeton University, (2004)
10. Burns, J., Cheng, A., Gurung, P., Rajagopalan, S., Rao, P., Rosenbluth, D., Surendran, A.V., Martin Jr, D. M.: Automatic Management of Network Security Policy. In: DARPA Information Survivability Conference and Exposition, vol. II, pp. 12-26, (2001)
11. Guttman, J.D.: Filtering postures: local enforcement for global policies. In: IEEE Symposium on Security and Privacy, pp. 120-129, (1997)

12. Yair Bartal, Alain J. Mayer, Kobbi Nissin and Avishai Wool.: Firmato: A novel firewall management toolkit. ACM Transactions on Computer Systems, vol. 22, no. 4, pp. 381-420, (2004)
13. DOD: Trusted Computer Security Evaluation Criteria. DOD 5200.28-STD. Department of Defense, (1985)
14. Albuquerque, J. P., Krumm, H. ; Geus, P.L.: Policy Modeling and Refinement for Network Security Systems. In: IEEE 6th International Workshop on Policies for Distributed Systems and Networks, pp. 24-33, (2005)
15. Spivey, J. M.: The Z notation: a reference manual. Prentice Hall International (UK) Ltd., Hertfordshire, UK, (1992)
16. Saaltink, M.: The Z/EVES system. In: J. P. Bowen, M. G. Hinchey, and D. Till, (eds.), ZUM 1997 LNCS, vol. 1212, pp. 72–85, Springer-Verlag, (1997)
17. Kropiwiec, C. D.: Z-specification for Firewall Policies, Algorithms and Theorem Proofs. Available at: http://www.ppgia.pucpr.br/~jamhour/Research/, (2008)