A Concept of Unification of Network Security Policies

Dmitry Chernyavskiy
Information Security of Banking Systems Department
National Research Nuclear University MEPhI
31 Kashirskoe shosse, 115409, Moscow, Russia
+7-905-7751950
d.s.ch@mail.ru

Natalia Miloslavskaya
Information Security of Banking Systems Department
National Research Nuclear University MEPhI
31 Kashirskoe shosse, 115409, Moscow, Russia
+7-495-7885699-9084
milnat2004@yahoo.com

ABSTRACT
Security policy is a main mechanism of information security management. While there are a lot of security-related standards and guidelines which specify requirements for high-level security policies, implementation of network security policy still depends on interfaces provided by network security solutions (NSS). Obviously, diversity of policy representation languages affects efficiency of policy deployment process. The paper proposes a concept of unification of policy rules for NSSs as a solution for the problem. The idea is based on formal language which makes it possible to formalize network security policies independently of particular NSS.

Categories and Subject Descriptors
C.2.0 [Computer-Communication Networks]: General – security and protection;
C.2.3 [Computer-Communication Networks]: Network Operations – network management;
F.4.m [Mathematical Logic and Formal Languages]: Miscellaneous;
H.5.2 [Information Interfaces and Presentation]: User Interfaces – interaction styles, standardization;

General Terms
Security, Standardization, Languages.

Keywords

1. INTRODUCTION
Information security (IS) begins from information security policy in all organizations. In fact, IS policy is a steering mechanism which is used to establish security rules and requirements for entire organization: from business processes security to security of network switch. IS policy aims to “configure” an organization so that to mitigate all security-related risks to minimal acceptable level.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright © 2012 ACM 978-1-4503-1668-2 …$15.00.
2. OVERVIEW OF STANDARDS

The most detailed requirements for IS policies are described in international standards ISO/IEC 27002 [1], ISO/TR 13569 [2] and ISO/IEC 15408 [3].

In accordance with ISO/IEC 27002 "Code of practice for information security management" IS policy is a main instrument for IS management. The standard defines the basic topics that must be reflected in the policy and provides its structure. On recommendations of the standard, IS policy should be published together with documents describing detailed IS requirements. For example, more detailed policies and procedures for protection of specific information systems or security rules to be followed by users.

ISO/TR 13569 "Financial services - Information security guidelines" identifies three levels of IS documentation: IS policy, documents of IS practice and operational procedures. These documents are targeted to cover high-level organizational goals to configuration of devices that implement the policy. IS policy is the most concise document, it must contain high-level goals and be formulated in a way easy to understand, providing specific information about assets requiring protection, such as customers and employees data, banking secrecy, partnership agreements and business processes.

Documents of IS practice define general requirements to be followed by an organization and reflect intentions and objectives established by its management. These documents include requirements for policy implementation methods without technical details of concrete systems.

Operational procedures in accordance with ISO/TR 13569 describe a technology of policy implementation. These documents must be complete, accurate and do not contradict to any practice document or policy. Procedures usually contain requirements for policy implementation, including commands and settings which must be performed in particular system or security solution.

Standard ISO/IEC 15408 "Evaluation criteria for IT security" ("Common Criteria") is meant to be used as the basis for evaluation of security properties of IT products. IS policy is being used on different stages of an evaluation in the Common Criteria model. In terms of this standard, the policy is a component of the security environment of a target of evaluation. Policy statements are included into the protection profile document and the security target document and further used in a process of establishment of security objectives. Moreover, the standard defines three types of policies: organizational security policy, target of evaluation security policy and security function policy. Organizational security policy is one or more security rules, procedures, practices, or guidelines imposed by an organization upon its operations. Target of evaluation security policy is a set of rules that regulate how assets are managed, protected and distributed within a target of evaluation. And finally, security function policy is the security policy enforced by a security function. In the same time, security function is defined as a part or parts of the target of evaluation that have to be relied upon for enforcing a closely related subset of the rules from the target of evaluation security policy [3]. Hence, in accordance with the Common Criteria the security function policy is a subset of the target of evaluation security policy which is a subset of the organizational security policy.

Actually, operational procedure for particular NSS contains security functions policies including their expression in a language provided by NSS interface. Requirements listed in the procedure usually derive from different practice documents, for instance: network security policy, authentication policy, encryption policy, etc. In summary, figure 1 shows interdependency of policies based on discussed standards.

3. SYNTAX AND SEMANTICS

The Unified Language for Network Security Policy is intended to formally represent security policy rules for NSSs. Syntax and semantics of the language is based on security functions performed by NSSs. The ULNSP syntax is formalized with usage of generative grammar [4] which at the current moment consists of about 200 grammar rules. The grammar of the ULNSP is context-free that makes it easy to analyze it using well-known parsing algorithms. Formal expression of the grammar can be found in [5]. In this paper we consider syntax examples of the ULNSP and describe their semantics in order to show the idea of policy rules unification.

In the most common case every ULNSP sentence has the following structure:

\[
\text{identifier action function1(param11 ... param1N)} \ldots \text{functionK(paramK1 ... paramKM)}
\]

where identifier is sequence of symbols which identifies policy rule, it may consist of upper and lower case characters, numbers, symbols "+", ",", "/", "=" or can be empty.

action is an action performed by an NSS, it may have the following values: permit – permit action (for instance, to allow connection or passage of a packet); deny – deny action (for instance, to deny passage of a packet); drop – drop a packet without sending any reply to source; log – write an event to log; alert – notify about an event; permit log – permit action and write an event to log; permit alert – permit action and notify about an event; deny log – deny action and write an event to log; deny alert – deny action and notify about an event; drop log – drop a packet and write an event to log; drop alert – drop a packet and write an event to log. Also, action can be empty, in this case it means permit action by default.

function1 … functionK (K\epsilon\mathbb{N}) are keywords identifying security functions, param11 ... paramN, paramK1 ... paramKM (N,M\epsilon\mathbb{N}) are values of parameters of the functions.
Number of parameters and their sequence depend on particular function.

Current version of the ULNSP allows to specify policy rules with usage of traffic analysis functions, network address translation function, routing function, audit messaging function and also to use comments in policy rules. Let us consider syntax and semantics of traffic analysis rules:

```
Identifier action function1(param11... param1N) function2(param21... param2K)
function3(param31...param3M)
function4(param41...param4L)
```

where identifier and action have the same meaning as described above; function1(param11... param1N) is one of functions of analysis of data link layer traffic; function2(param21... param2K) is one of functions of analysis of network layer traffic; function3(param31...param3M) is one of functions of analysis of transport layer traffic; function4(param41...param4L) is a function of analysis of data field of network protocols.

For example, function of analysis of network layer traffic may refer to analysis of IP header. In this case function2 has the following syntax:

```
IP(Version IHL TOS TotalLength
Identification Flags FragmentOffset TTL
Protocol Checksum SourceAddress
DestinationAddress Options)
```

where IP is a keyword identifying the function of IP-header analysis. Each parameter of the function from semantic point of view is equal to respective IP-header field defined in RFC 793 [7]. All parameters except SourcePort and DestinationPort can be set as a decimal number or a hexadecimal number in the format equivalent to parameters of IP-header analysis function. Parameters SourcePort and DestinationPort can be set in the format $X$ which means single TCP-port, $<X$ for ports lower than $X$, $>X$ for ports higher than $X$, or in the format $X-X$ for a range of ports (here $X$ is a decimal number from 0 to 65535). As well as in IP-header analysis function, symbol "*" can be used instead of any parameter which means that its value is not defined. The function can be specified with the following syntax:

```
TCP(SourcePort DestinationPort)
```

that is equal to

```
TCP(SourcePort DestinationPort * * * * * * * *)
```

Keyword TCP without parameters is equal to

```
TCP(* * * * * * * *)
```

As can be seen from the ULNSP syntax examples, functions are separated with whitespaces, parameters of functions are also separated with whitespaces and enclosed into brackets. However, policy rules can be used without symbols of brackets and it will not affect the semantics. In this case brackets must be replaced with whitespaces, for instance:

```
TCP SourcePort DestinationPort
```

4. **UNIFICATION OF POLICIES**

We will consider two NSSs as equivalent with respect to a set of policy rules, if this set of rules can be identically implemented in both NSSs. In other words, if two NSSs use equal commands (and/or other manipulations with user interface) in order to implement particular set of policy rules, it means that these NSSs are equivalent with respect to this set of rules. Hence, this equivalence relation partitions all set of NSSs to equivalence classes. Also we will say that two NSSs are equivalent if any policy rule is being identically implemented in both NSSs.

Obviously, NSSs of different manufacturers cannot be equivalent. In order to demonstrate the concept of unification of policy rules we consider three NSSs which belong to pairwise different classes with respect to traffic filtration rules. The first one is Cisco router, the second one is Check Point Firewall-1 and the last one is Snort IDS. All of them are capable of performing traffic filtration rules, but these NSSs use different mechanisms for their implementation.

Let us have the following simple policy rule: “Hosts from the network 192.168.1.0/24 are allowed to establish connections to 80 TCP-port on server 10.1.1.10, connections attempts must be logged”. In order to implement the rule in Cisco router it is necessary to add command to access-list (for instance, access-list 101):

```
access-list 101 permit tcp 192.168.1.0 255.255.255.0 host 10.1.1.10 eq 80 log
```

The same rule can be implemented in Check Point Firewall-1 with usage of software SmartDashboard that provides GUI and may look like it is shown on figure 2.
Snort IDS can be configured through changes in configuration files. In order to implement policy rule, it must be added to configuration file. The rule itself will have the following view:

```
log tcp 192.168.1.0/24 any -> 10.1.1.10 80
```

As can be seen from above examples syntax of Cisco commands and Snort rules are completely different. Moreover, Check Point does not use CLI and it is not possible to compare its policy rules with analogues in Cisco and Snort from syntax point of view. However, semantics of these three rules are equal and it is possible to formalize required rule with the ULNSP:

```
permit log IP(192.168.1.0/24 10.1.1.0) TCP (* 80)
```

If we remove brackets and keyword `permit`, the rule will look like following:

```
log IP 192.168.1.0/24 10.1.1.0 TCP * 80
```

The process of implementation also depends on particular NSS. In the case of Cisco we need to connect directly to the device, enter to privileged mode with command `enable`, then switch to configuration terminal mode using command `configuration terminal`, further input the command with the rule described above. In order to implement the rule in Firewall-1 we need to connect to management server (not directly to firewall module) through SmartDashboard, open policy where the rule should be created, add the rule configuring its parameters within GUI and further install the policy on Firewall-1. Snort is being configured as following: connect to Snort box, add the rule to configuration file containing policy and restart Snort.

If we wish to implement policy rules formalized with the ULNSP, we need to utilize its translator [5]. On input of the translator it is necessary to submit a class of an NSS (in described case it could be Cisco router, Firewall-1 or Snort), its IP-address and a file with policy rules. The ULNSP translator performs all configuration actions. It connects to the required NSS and executes appropriate commands as described previously in the case of Cisco or Snort. For establishing connection with Check Point management server, addition rules and implementing policy, the translator applies Check Point Management Interface Application Programming Interface (CPMI API) [8].

Hence, while using the ULNSP and its translator, all implementation steps progress transparently to user and from user’s point of view the process is independent of NSS class. In this case we can say that three discussed NSS belong to one equivalence class with respect to considered policy rules and the rules become portable between these NSS. Moreover, the ULNSP provides syntax with minimal redundancy and quite obvious semantics that makes the language easy to use.

5. RELATED WORKS

A lack of unified language for policy description is a well-known problem and there have been quite a few efforts to develop new or adapt existing formal languages in order to specify network security policy. Classification and analysis of such languages as Ponder, XACML, Rei, LaSCO, EPAL and others can be found in [9]. Adaptation examples of Datalog language are represented in Secure Network Datalog (SeNDlog) [10] and Flow-based Security Language (FSL) [11]. Also there are some policy models that are founded on organization based access control (OrBAC) and used to specify network security policy [12], [13].

Despite the variety of existing policy languages and representation models, however, they are not widely used. The main reason for that is a redundancy of these languages. In order to specify a policy for an NSS applying them, it is necessary to describe additional language structures, because, generally, their policy rules are not self-sufficient and cannot be used outside of a context. Given languages follow top-down approach specifying high-level structures and then descending to low-level rules which to be translated into system configuration. These structures are usually abstract that makes it necessary to reclassify organization's information assets and network systems in accordance with language requirements and consider them in a context proposed by a language. Classification and categorization of assets are commonly being made on the second level of IS policy hierarchy described in section 2 and, obviously, represented in a natural language. Application of mentioned languages requires redefinition of higher-level policies in a low-level policy for an NSS or inclusion of formal language structures into high-level documents that makes the documents tied to low-level policies. All these peculiarities lead to expenditure of additional efforts and raise inconvenience of usage of policies. In addition, most of languages use symbols which are semantically redundant and inconvenient to type.

Through analysis and research of the existing policy languages, the following requirements for a good policy language were presented in [14]:

- **The language should be easy to understand.** Realization of this requirement helps to avoid ambigiuosity in policy representation and to efficiently use the language by a human.

- **Language semantics should avoid absolutes.** Because of difficulty in mapping abstract representations into implementation mechanisms, it is recommended to exclude them from the language.

- **The language should be doable.** The requirement means that policy rules should be easily transformable to a description of underlying systems configurations.

- **High covering of policies.** The more types of policies the language supports the more applicable it is.

- **The language should be scalable.** In order to add new types of policy rules into the language, it should be structured so that the addition will not affect an overall logic.
• *Specification of the language should be open-source.* This should help to effectively apply translation algorithms and add new types of policy rules into the language.

Most of existing languages are not in line with at least first two requirements. Considering the ULNSP it is evident that the language meets listed requirements. In particular, new types of policy rules can be added to the ULNSP specification by adding new functions or combining of existing ones (see [5] for details).

6. CONCLUSION

Nowadays there are plenty of NSSs developed by different manufactures and all of them use different mechanisms for specification and implementation of IS policies. With appearance on the market Unified Threat Management (UTM) solutions which combine variety of different security functions and technologies that allows to embed various security modules into one physical appliance, concentration of all possible security functions in one solution becomes the main trend and borders between firewalls, IDS/IPS, DLP systems and other NSSs are blurring. Nevertheless, as it was shown in the paper, NSSs of different types and manufacturers are able to implement policies that are semantically equal.

Unification as a type of standardization usually leads to increase in efficiency by means of reducing useless diversity and redundancy. Presented model is an effort to unify low-level policies for NSSs and simplify mechanisms of their implementation. By means of partitioning of the set of NSSs to equivalence classes and applying a single policy to each class it is possible to eliminate redundancy of policies for NSSs and, as a result, increase efficiency of policies development process. By the use of translator, unified policies become portable between different NSSs, that simplifies processes of their implementation. And finally, any new NSS can be added to the model by splitting it to simple NSSs and assigning them to some equivalence class. If there is no NSS in the classification to which the new one is equivalent then the new NSS forms its own class and, consequently, unified policy is to be created for this new class following the same principles as other unified policies (by adding new elements to an algebra or new grammar rules to the language). The future challenge for the model is a development of a comprehensive classification of existing NSSs in terms of described equivalence, generation of unified policies for the equivalence classes and progress in construction of policy translation methods.

7. REFERENCES