Unified Language for Network Security Policy Implementation

Dmitry Chernyavskiy
Information Security Faculty
National Research Nuclear University MEPhI
Moscow, Russia
milnat2004@yahoo.co.uk

Natalia Miloslavskaya
Information Security Faculty
National Research Nuclear University MEPhI
Moscow, Russia
NGMiloslavskaya@mephi.ru

***Abstract*—The problem of diversity of the languages on network security appliances' interfaces is discussed. Idea of the unified language for network security policy (ULNSP) implementation is proposed. A basic approach to the ULNSP formalization is considered. ULNSP Grammar and syntax examples are given. Further research on UNLSP is briefly discussed.**

Keywords-Network Security Policy, Network Security Appliances, Formal Language, Syntax, Grammar, Translator

I. INTRODUCTION

Communication in the modern world cannot be imagined without such a concept as a language – an effective way of representation and information transfer. At present there is a large variety of different languages created by people for their own needs: a sign language, a body language, a languages of mathematical and chemistry formulas, graphics languages applied in plotting and designer activity and others. All formats of input/output data define a language for information and communication technologies. Very often program systems have really complex languages for their interfaces, including declarations, instructions, expressions and many other kinds of data sets. This is the payment for programs and devices functionality.

The main reason of this diversity of possible languages is a tendency to represent the information in a form as much as possible short and convenient for a particular task solution. Information security (IS) sphere has not avoided this issue. Input and output languages are used for management of the majority of network security appliances (NSA), having their own set of instructions for implementation of the corporate IS policy (ISP), containing many rules (ISPR). Hereafter, by different NSA, we mean such appliances, for which at least one ISPR can be specified so that its implementation in one appliance is available only with the set of commands different from those used for implementation of this rule in another appliance. While protecting systems by different manufacturers, their command languages can differ so significantly that at the first glance it can be seen that there is nothing common among them. Obviously, the diversity of input languages on interfaces of NSA creates problems for IS managers because it is necessary to "translate" their ISPR for each particular device into a specific set of instructions. Such "translation" takes much more time than in case if all NSA could detect the identical commands. Moreover, it can result in errors in ISPR implementation configuring devices, especially if some rules are ambiguously formulated.

There are some solutions on the market that try to make a universal interface for ISP implementation. One of them is Check Point SmartDashboard [1], but this software product supports only Check Point security devices. Cisco Security Manager software [2] also uses policy-based approach to management of routers, firewalls and IPS systems, but, obviously, it supports only Cisco products. There are a lot of different solutions designed in order to improve an existing interface of a particular network security solution and make it more convenient, for instance, Activeworx IDS Policy Manager [3] is an application that provides GUI for policy-based management of Snort IDS. While network security models play an important role in any system, most research effort related to this topic are based on limited concept and do not discuss all the richness of current and emerging NSA. A development of the unified language for network security policy (ULNSP) implementation may solve these problems. The language will allow implementation of network security policy rules (NSPR) in a convenient form without being dependent on a particular device. Obviously, the language itself doesn't have any practical application; therefore its translator should be developed as well. Inherently, ULNSP together with its translator will form a universal interface between the human and NSA and, as a result, will increase the efficiency of the network security management.

Section II of the paper presents requirements and the main idea of UNLSP. Basic approach for formalization of the language is described in Section III. Section IV provides examples of NSPR defined with ULNSP. Practical application of the language is considered in Section V. Conclusions are given in Section VI.

II. UNIFIED LANGUAGE FOR NETWORK SECURITY POLICY BASIS

For the effective implementation of any policy, its rules must meet the requirements of maximum simplicity, clarity and ability for updating. Any policy rule should be formulated so that it could not be interpreted ambiguously. Implementation of these requirements should reduce the

 ISBN: 978-1-61208-133-5

probability that any rule will be ignored or implemented incorrectly, as well as the probability of bypassing the rules.

NSPR are specified in the corporate ISP in a natural human language and are subsequently implemented as a configuration (settings) for NSA. So the problem arises while translating NSPR into NSA commands by human (system, network or IS administrator). But different NSA recognize different languages, making it necessary to translate the same NSPR in a different set of commands. Such kind of translation may lead to typographical errors; so, that the device cannot accept this command. In the worst case, if a wrong command is syntactically correct, it can be applied by device, and this can cause incorrect functioning of the network or appearance of security breaches.

Proposed ULNSP would help to avoid such problems and solve the problem of NSPR portability and consequently improve network security management efficiency. In fact NSPR defined by ULNSP language will be the intermediate between ISPR and commands of a particular NSA (Fig. 1). In this case, network administrator (or other person responsible for the network security management) has to translate the rules from the corporate ISP into the rules in ULNSP, but this operation for a specific ISP must be done only once without concentrating on particular NSA models.

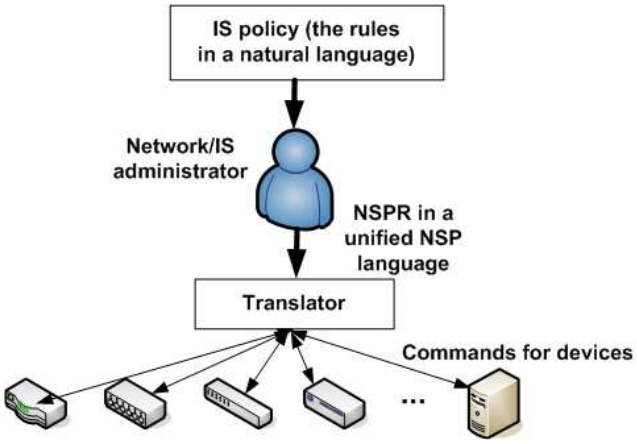

Figure 1. NSP Implementation using ULNSP.

Another important requirement is ULNSP extensibility – if we need to add a new type of devices or rules that the language should support, this process would be as simple as possible, and the changes wouldn't change the overall language structure.

III. ULNSP FORMALIZATION

For ULNSP formalization we use generative grammar G, which defines the rules for constructing sentences of the language $G = (T, N, S, R)$ [4], where T - a finite nonempty set - the terminal vocabulary (its elements are called terminal symbols TS); N - a finite nonempty set - non-terminal vocabulary (its elements are called non-terminal symbols); S - selected item of non-terminal vocabulary, so-called start symbol or axiom of the grammar; R - nonempty finite set of rules (productions), each of which has the form $\alpha \rightarrow \beta$, where α and β - chains on the dictionary $T \cup N$. In addition it is compulsory that $T \cap N = \varnothing$.

The chain β is directly derivable from the chain α in the grammar G (designated by $\alpha \Rightarrow_G \beta$), if the chain α can be represented as a concatenation of the three chains $\beta = \mu\xi\nu$ (some of them may be empty), chain β can be also represented as a concatenation of three chains $\beta = \mu\xi\nu$ and grammar G contains the production $\tau \rightarrow \xi$. Symbol $\Rightarrow_G$ denotes the binary relation on the set of all chains over the union of the vocabularies $T \cup N$. The chain β is directly derivable from the chain α in the grammar G (designated by $\alpha \Rightarrow_G^* \beta$), if in grammar G exists a finite set of strings $\pi_0, \pi_1, \ldots, \pi_n$, n>0 such that $\alpha = \pi_0$, $\pi_n = \beta$ for all $i = 1,\ldots, n$ holds $\pi_{i-1} \Rightarrow \pi_i$. Symbol $\Rightarrow_G^*$ denotes the reflexive transitive closure of $\Rightarrow_G$.

Formal language generated by G is a set of chains composed of TS of the grammar and the vocabulary derived from the grammar's start symbol:

$$L(G) = \{\alpha \in T^*: S \Rightarrow_G^* \alpha\}.$$

It is important to note that, in general, one language can be generated by different grammars.

Here an example of ULNSP grammar. The non-terminal N and the terminal T vocabularies should be defined:

N = {policy rule, identifiers, actions, functions, params, separators, permit action, deny action, traffic filtration, address translation, routing, interface, data link layer, network layer, transport layer, protocol ethernet, protocol IP, protocol ICMP, protocol TCP, protocol UDP, Ethernet params, IP params, ICMP params, TCP params, UDP params, address translation params, routing params, interface params, IPv4 addresses, IPv4 mask, Destination MAC, Source MAC, Type, MAC address, Version, IHL, Type of Service, Total Length, Identification, IP Flags, Fragment Offset, Time to Live, Protocol, Checksum, Source Address, Destination Address, Options, Source Port , Destination Port, Sequence Number, Acknowledgment Number, Data Offset, Reserved, TCP Flags, Window, Checksum, Urgent Pointer, Options, Length, Code, internal name, external name, local address, global address, interface ID, destination address, gateway, interface name, interface address, security level}

T={0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v, w,x,y,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,W,X,Y, Z-,.,_,(,),},*

or in another words T consists of characters with ASCII codes from 33 to 126. In N a *policy rule* is an axiom of G.

Fig. 2 shows the scheme of ULNSP rules construction. This diagram is not a formalized method of representation of a grammar, but we use this figure in order to make an understanding of the language easier. Set of productions R can be found in Appendix.

 ISBN: 978-1-61208-133-5

The grammar of ULNSP is context-free. This grammar type is widespread in computer science and there are a lot of relatively efficient parsing algorithms that are applicable to detect chains of languages generated by context-free grammars.

IV. ULNSP SYNTAX AND SEMANTICS EXAMPLES

To block all traffic coming to the router's interface eth0 from the host with the address 192.168.1.1 to the host with the address 10.1.1.1 the rule in ULNSP will be as follows:

eth0 deny IP (192.168.1.1 10.1.1.1), or
eth0 deny IP 192.168.1.1 10.1.1.1

The following rule allows the transfer of TCP packets from the network 192.168.1.0/24 to the 80th port of the host with the address 10.1.1.2 on eth1 interface of the firewall:

eth1 permit IP (192.168.1.0/24 10.1.1.2) TCP (* 80), or
eth1 IP (192.168.1.0/24 10.1.1.2) TCP * 80

The following rule blocks all ICMP traffic via eth1:

eth1 deny ICMP

The following rule blocks TCP packets with FIN, URG and PSH flags to the host with the address 10.1.1.3 through the interface eth0 on NSA:

eth0 deny IP (* 10.1.1.3) TCP (* * * * * * FIN URG PSH * * *)

The rule for network address translation of the addresses 192.168.1.0/24 on the interface eth0 to addresses 10.1.1.0/24 on eth1 looks like that:

NAT (eth0 eth1 192.168.1.0/24 90.1.1.0/24), or
NAT eth0 eth1 192.168.1.0/24 90.1.1.0/24

For a static address translation 192.168.1.2 in the address 10.1.1.2 you can use the following rules:

NAT eth0 eth1 192.168.1.2 90.1.1.2

Translating of the address range 192.168.1.1 - 192.168.1.10 into the address 10.1.1.1 can be expressed by the following rule:

NAT eth0 eth1 192.168.1.1-192.168.1.10 10.1.1.1

As can be seen from the above examples of usage, ULNS has a convenient syntax and intuitively obvious semantics.

V. NLNSP TRANSLATOR

The main goal of NLNSP development is to create a universal interface between a human and NSA. It is obvious that the language itself has no practical applicability, because all ISPR, described in it, in fact, are a formal expression of rules defined in any natural language. In order to make the language applicable to the real devices a translation system is needed. The functions of this system include establishing a connection (telnet, ssh and so on) to the device; identifying the device, its version and functionality; translating NSPR from ULNSP into commands of a particular device; performing additional device configuration (if necessary).

Thus, in the ideal case, this system allows to use the same unified set of NSPR for all NSA with the required functionality making all required settings to adjust them in accordance with NSP. In reality it is almost impossible to cover all the existing diversity of devices of different manufacturers, types and versions, so the main challenge in this case is to support as many systems as possible.

The final result of the development is a software product, which allows to configure NSA by implementing NSP described in ULNSP. A security administrator will be able to set up a device in accordance with NSP, without being dependent on manufacturer of this device.

Let us consider the main convenience of the system. When designing a network security system for a corporation, the requirements come from its ISP. In fact it doesn't matter what kind of device is applied in a particular part of a network. It is important that this solution should have required functionality. Because the basic ULNSP concept is a function-based approach for formalization of the rules, it won't be a problem to formulate NSPR using the language. For configuring any NSA in accordance with the policy it is just necessary to connect to this device using the proposed system and input ISPR in the unified language. All necessary settings will be done automatically by the system.

VI. CONCLUSION

The main peculiarities of created ULNSP are context-free grammar and its extensibility. The function-based approach in rules formalization used allows to add new types of rules easily. Inherently, any policy rule formalized in ULNSP describes security function policy (defined in ISO/IEC 15408 [5]). The language also provides convenient syntax and intuitively obvious semantics. For example, all parameters for TCP traffic filtration rules follow RFC 793 [6].

ULNSP translator will help to automatically configure NSA in accordance with the corporate ISP by implementing the corresponding rules described with the language. For this purpose it is just necessary to know IP-address of the device and ISPR in ULNSP.

Today, ULNSP and its translator support a basic set of NSPR for firewalls and routers. The future challenge for the development is an extension of the set of rules that could be formalized with ULNSP such as rules for IDS/IPS systems, DLP systems, VPN rules and so on.

REFERENCES

[1] http://www.checkpoint.com/products/smartcenter/smartcenter_manag manag.html (last access date 16/03/2011)

[2] http://www.cisco.com/en/US/products/ps6498/index.html (last access date 16/03/2011)

[3] http://activeworx.org/programs/idspm/index.htm (last access date 16/03/2011)

[4] Karpov Y.G. Fundamentals of translators design. - St.Petersburg.: BHV, 2005 (In Russian).

[5] ISO/IEC 15408 Information technology - Security techniques - Evaluation criteria for IT security.

[6] RFC793 - Transmission Control Protocol.

 ISBN: 978-1-61208-133-5

APPENDIX

Set of productions R:

1) policy rule→ identifiers, separators, actions, separators, functions
2) identifiers→ α|ε
3) separators→ « »
4) actions→ permit action | deny action
5) permit action→ ***permit****|ε*
6) deny action → ***deny***
7) functions→ address translation | routing | interface | traffic filtration
8) address translation n→ ***NAT****,* ***(****, address translation params,* ***)***
address translation → ***NAT****, separators, address translation params*
9) address translation params → internal name, separators, external name, separators, local address, separators, global address
10) internal name→ α; external name→ α
11) local address→ IPv4 address, IPv4 mask
local address→ IPv4 address, -, IPv4 address
12) global address→ IPv4 address, IPv4 mask
global address→ IPv4 address, -, IPv4 address
13) IPv4 address→ $\beta_1.\beta_2.\beta_3.\beta_4$
14) Pv4 mask→ /ξ|ε
15) routing→ ***Route****,* ***(****, routing params,* ***)***
routing → Route, separators, routing params
16) routing params→ interface ID, separators, destination address, separators, gateway
17) interface identifier→ α
18) destination address→ IPv4 address, IPv4 mask
19) gateway→ IPv4 address
20) interface→ ***Interface****,* ***(****, interface params,* ***)***
interface → ***Interface****, separators, interface params*
21) interface params → interface ID, separators, interface name, separators, security level, separators, interface address
*22) interface name→ α|********
*23) security level→ τ|********
*24) interface address→ Адреса IPv4, Маска IPv4|**
25) traffic filtration→ data link layer | network layer | transport layer |
data link layer, separators, network layer |
data link layer, separators, transport layer |
network layer, separators, transport layer |
data link layer, separators, network layer, separators, transport layer
26) data link layer→ protocol Ethernet
27) network layer → protocol IP | protocol ICMP
28) transport layer→ protocol TCP | protocol UDP
29) protocol Ethernet→ ***Ethernet****,* ***(****,Etherne paramst,* ***)***
protocol Ethernet→ ***Ethernet****, separators, Ethernet params*
protocol Ethernet→ ***Ethernet***
30) Ethernet params→ Destination MAC, separators, Source MAC, Type
*31) Destination MAC→ MAC address|**
*32) Source MAC→ MAC address|********
33) MAC address→ $\sigma_1\sigma_2$**:**$\sigma_3\sigma_4$**:**$\sigma_5\sigma_6$**:**$\sigma_7\sigma_8$**:**$\sigma_9\sigma_{10}$**:**$\sigma_{11}\sigma_{12}$
34) Type→ ***0x****$\sigma_1\sigma_2|\beta_1|$********$|\varepsilon$
35) protocol IP→ ***IP****,* ***(****, IP params,* ***)***
protocol IP→ ***IP****, separators, IP params*
protocol IP→ ***IP***
36) IP params → Version, separators, IHL, separators, Type of Service, separators, Total Length, separators, Identification, separators, IP Flags, separators, Fragment Offset, separators, Time to Live, separators, Protocol, separators, Checksum, separators, Source Address, separators, Destination Address, separators, Options
IP params → Source Address, separators, Destination Address
37) Version→ ***0x****$\sigma_1|\varphi|$********
38) IHL→ ***0x****$\sigma_1|\varphi|$**
39) Type of Service→ ***0x****$\sigma_1\sigma_2|\beta_1|$********
40) Total Length→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4|\eta|$********
41) Identification→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4|\eta|$********
*42) IP Flags→ θ|********
*43) Fragment Offset→ η|********
*44) Time to Live→****0x****$\sigma_1\sigma_2|\beta_1|$********
*45) Protocol→****0x****$\sigma_1\sigma_2|\beta_1|$********
*46) Checksum→****0x****$\sigma_1\sigma_2\sigma_3\sigma_4|\eta|$**
*47) Source Address→ IPv4 address, IPv4 mask | IPv4 address, -, IPv4 adress|**
*48)Destination Address→ IPv4 address, IPv4 address|IPv4 address,-, IPv4 address|**
49) Options→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_8$, Options$|\varepsilon|$********
50) protocol TCP→ ***TCP****,* ***(****,TCP params,* ***)***
protocol TCP→ ***TCP****, separators, TCP params*
protocol TCP→ ***TCP***
51) TCP params → Source Port , separators, Destination Port, separators, Sequence Number, separators, Acknowledgment Number, separators, Data Offset, separators, Reserved, separators, TCP Flags, separators, Window, separators, Checksum, separators, Urgent Pointer, separators, Options
TCP params → Source Port , separators, Destination Port
52) Source Port→ η|>η|<η|η_1**-**η_2**|********
53) Destination Port→ η|>η|<η|η_1**-**η_2**|********
54) Sequence Number→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_8|\nu|$********
55) Acknowledgment Number→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_8|\nu|$**
56) Data Offset→ ***0x****$\sigma_1|\varphi|$********
*57) Reserved→ χ|********
*58) TCP Flags→¥| χ|********
59) Window→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4|\eta|$********
60) Urgent Pointer→ ***0x****$\sigma_1\sigma_2\sigma_3\sigma_4|\eta|$********
61) protocol UDP→ ***UDP****,* ***(****, UDP params,* ***)***
protocol UDP→ ***UDP****, separators, UDP params*

 ISBN: 978-1-61208-133-5

protocol UDP→ ***UDP***
62) UDP params → Source Port, separators, Destination Port, separators, Length, separators, Checksum
UDP params → Source Port, separators, Destination Port
63) Length→ $\mathbf{0x}\sigma_1\sigma_2\sigma_3\sigma_4|\eta|^*$
64) protocol ICMP→ ***ICMP****, (, ICMP params,)*
protocol ICMP→ ***ICMP****, separators, ICMP params*
protocol ICMP→ ***ICMP***
65) ICMP params → Type separators, Code, separators, Checksum
66) Code→ $\mathbf{0x}\sigma_1\sigma_2|\beta_1|^*$
where α – a finite chain of TS on the dictionary *{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v, w,x,y,z,-,_} ⊂T*;

βi – string of TS belonging to the set *{0,1,2,3,4,5,6,7,8,9, 10,11,12,…,253,254,255} ⊂T*;
ξ - a chain of TS belonging to *{1,2, …, 32}⊂T*;
τ - a chain of TS belonging to the set *{0,1,2, …, 100}⊂T*;
σ_i – a terminal symbol belonging to the set *{0,1,2, …, 9, A, B, C, D, E, F}⊂T;*
φ - a chain of TS belonging to *{1,2, …, 15}⊂T*;
η - a chain of TS belonging to *{1,2, …, 65535}⊂T*;
θ - a chain of TS belonging to the set *{1,2, …,7}⊂T*;
γ - a chain of TS belonging to the set *{0,1,2, …,31}⊂T*;
ν - a chain of TS belonging to the set *{0,1,2, …, 4294967295}⊂T;*
χ - a chain of TS belonging to the set {0, 1, 2, … ,63}*⊂T;*
¥ - a sequence consisting of space-separated distinct elements of *{URG, ACK, PSH, RST, SYN, FIN}.*

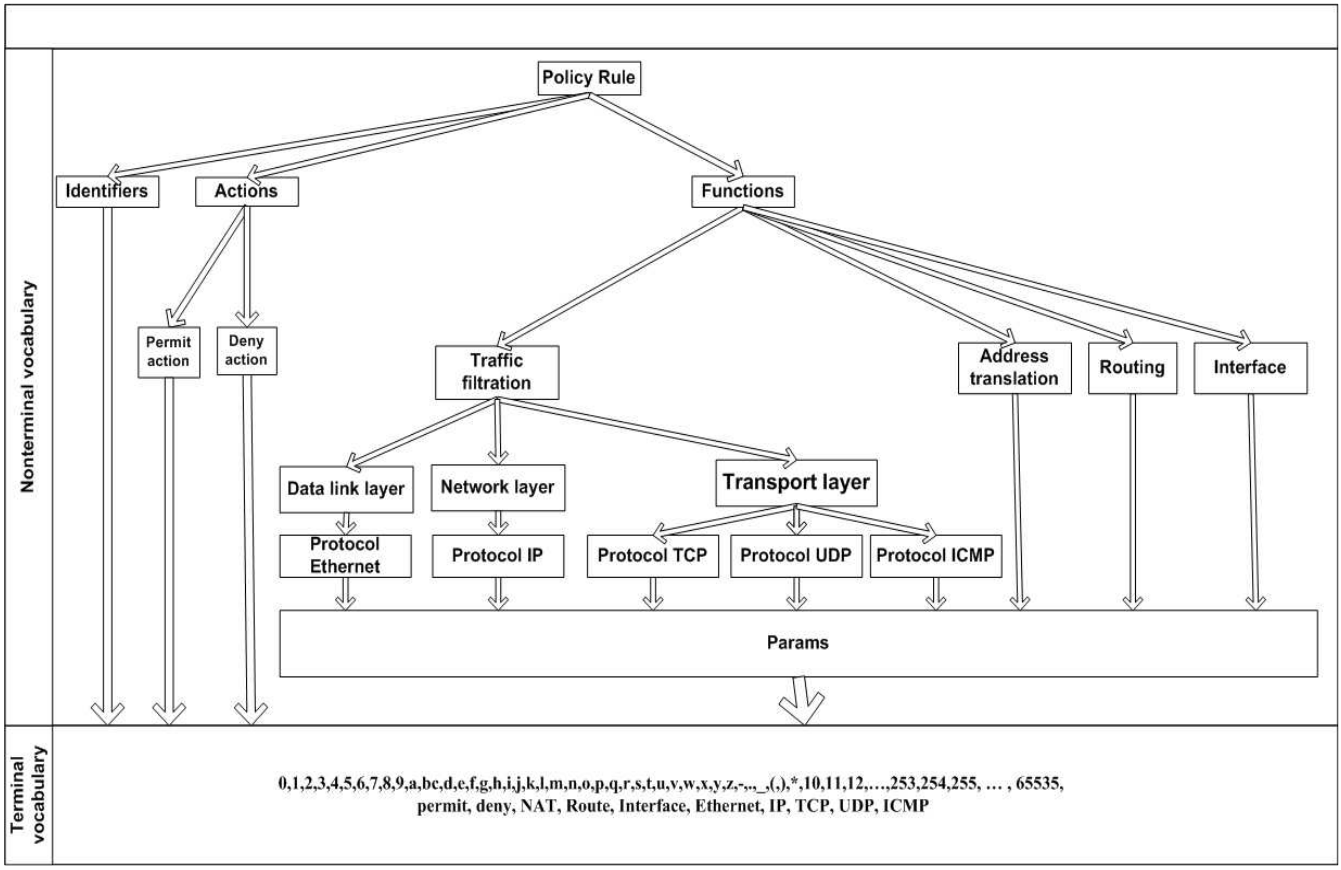

Figure 2. ULNSP rules construction.