Enhancement Misconfiguration Management of Network Security Components Using Range Algorithm

Ahmed Farouk[1], **Hamdy N.Agiza**[2], **Elsayed Radwan**[3]
[1,3]Faculty of Computer and Information Sciences, Mansoura University, Egypt,

[2]Faculty of Sciences, Mansoura University, Egypt,

Summary
Many companies and organizations use firewalls to control the access to their network infrastructure. When processing packages, conflicts due to rule overlaps can occur within the filtering policy. To solve these conflicts most firewall implementation use a first matching strategy through the ordering of rules. This way each packet processed by the firewall is mapped to the decision of the rule with highest priority. This strategy introduces however new configuration errors such as shadowing of rules and redundancy lead to inaccurate results. In this paper new algorithm called range algorithm introduced to get the best case for solving conflict and shadowing problems. Also get result rules that is free inconsistency and finding rules that cause inconsistency.

Keywords: *Network Security, Firewalls, Redundancy and Shadowing of Rules, Conflict, and Range Algorithm*

1. Introduction

Network security is essential to the development of Internet and has attracted much attention in research and industrial communities. With the increase of network attack threats, firewalls are considered effective network barriers and have become important elements not only in enterprise networks but also in small-size and home networks. A firewall is a program or a hardware device to protect a network or a computer system by filtering out unwanted network traffic. The filtering decision is based on a set of ordered filtering rules written based on predefined security policy requirements. Firewalls can be deployed to secure one network from another. However, firewalls can be significantly ineffective in protecting networks if policies are not managed correctly and efficiently. It is very crucial to have policy management techniques and tools that users can use to examine, refine and verify the correctness of written firewall filtering rules in order to increase the effectiveness of firewall security [12].

Firewalls are network security components which provide means to filter traffic within corporate networks, as well as to police incoming and out coming interaction with the Internet [6]. Firewall ACLs can contain inconsistencies. There is an inconsistency if different actions can be taken on the same flow of traffic, depending on the ordering of the rules. Inconsistency rules should be notified to the system administrator in order to remove them. Minimal diagnosis and characterization of inconsistencies is a combinatorial problem. Although many algorithms have been proposed to solve this problem, all reviewed ones work with the full ACL with no approximate heuristics, giving minimal and complete results, but making the problem intractable for large, real-life ACLs [11] in this paper a different approach introduced.

First, we deeply analyze the inconsistency diagnosis in firewall ACLs problem, and propose to "change" split the process in several parts that can be solved sequentially: Division Process, inconsistency detection, and Test Completeness & correctness

Filtering Rule Format It is possible to use any field in IP, UDP or TCP headers in the rule filtering part, however, practical experience shows that the most commonly used matching fields are: protocol type, source IP address, source port, destination IP address and destination port. Some other fields, like TTL and TCP flags, are occasionally used for specific filtering purposes. The following is the common format of packet filtering rules in a firewall policy: <Order><protocol><src_ip><src_port><dst_ip><dst_port > <action> [11] In this work the rules defined will be written as follows Ri : {condition}$\rightarrow$ decision [3] equivalent to where i is the relative position of the rule within the set of rules, decision i is a Boolean expression in {accept; deny}, and {condition} I is a conjunctive set of condition attributes such that {condition} I equals A1 ^ A2 ^ ::: ^ Ap, and p is the number of condition attributes of the given filtering rules. Attributes of conditions are Source and Destination IP address, in our work we suppose Source port, Destination port and IP protocol true value, there're two types of decision are Accept and Deny, Use ipv4 and network class C that address size is 32 bits, four octets, each one is 8 bit. Although firewall security has been given strong attention in the research community, the EMPHASIS was mostly on the filtering performance and hardware support issues. On the other hand, few related work [2] present a resolution for the correlation conflict problem only. Other approaches [1] propose using a high-

Manuscript received June 5, 2009
Manuscript revised June 20, 2009

level policy language to define and analyze firewall policies and then map this language to filtering rules. Firewall query-based languages based on filtering rules are also proposed in [10]. So in general, In this paper a new progress in this area because it offers new techniques for complete anomaly discovery and rule editing that can be applied on legacy firewall policies of low-level filtering rule representation, solving Disadvantages of latest approach are Inaccurate results, Number of algorithms used to perform functions, Time is huge, Performance is low.

2. Preliminaries

2.1 Analysis of Consistency Problems

To understand the problem, it is important to firstly review the inconsistencies characterized in the bibliography. A complete characterization that includes inconsistency, shadowing, and redundancy has been given in [3,5and11]. Although all of these are inconsistencies, usually not all are considered to be errors, as it can be used to cause desirable effects.

--Inconsistency: -- Two rules Ri, Rj ∈RS are inconsistent if and only if the intersection of each of all of its selectors R [k] is not empty, and they have different actions, independently of their priorities. The inconsistency between two rules expresses the possibility of an undesirable effect in the semantics of the rule set. The semantics of the rule set changes if an inconsistent rule is removed.

Definition 1

$$Inconsistent(R_i, RS), 1 \le i \le n \Leftrightarrow \exists R_j \in RS, 1 \le j \le n, j \ne i \bullet$$
$$R_i[k] \cap R_j[k] \ne \varnothing \wedge R_i[Action] \ne R_j[Action]$$
$$\forall k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

Inconsistency of one rule in a RS

$$Inconsistent(R_i, R_j, RS), 1 \le i, j \le n, i \ne j \Leftrightarrow$$
$$R_i[k] \cap R_j[k] \ne \varnothing \wedge R_i[Action] \ne R_j[Action]$$
$$\forall k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

--Shadow-- A rule Ry is shadowed by another rule Rx, with Rx>Ry, if all of its selectors to or supersets of the selectors of Ry, and Rx and Ry have different action.

Definition 2

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Shadow(R_y) \Leftrightarrow$$
$$\forall k \bullet R_y[k] \subset R_x[k] \wedge R_x[Action] \ne R_y[Action]$$
$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

Shadow

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet ExactShadow(R_y) \Leftrightarrow$$
$$\forall k \bullet R_y[k] = R_x[k] \wedge R_x[Action] \ne R_y[Action]$$
$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

Exact shadow

--Redundancy-- A rule Rx is redundant to another rule Ry, with Rx>Ry, if all of its selectors are subsets or equal to the selectors of Rx, they have the same action, and if there is no rule between Rx and Ry which is correlated or subset of Rx. Redundancy of Ry respect to Rx is symmetrical. Redundancy is not really an inconsistency, since if all redundant rules are removed; the semantic of the rule set does not change.

Definition 3

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Redundant(R_x) \Leftrightarrow$$
$$\forall k \bullet R_x[k] \subseteq R_y[k] \wedge R_x[Action] = R_y[Action] \wedge$$
$$\neg \exists R_z \in RS, R_x > R_z > R_y \bullet$$
$$Correlation(R_x, R_z) \vee Generalization(R_z)$$
$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Redundant(R_y) \Leftrightarrow$$
$$\forall k \bullet R_y[k] \subset R_x[k] \vee \forall k \bullet R_y[k] = R_x[k] \wedge$$
$$R_x[Action] = R_y[Action]$$
$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

2.2 Related and Previous Work

A first approach to get a firewall configuration free of errors is by applying a formal security model to express the network security policy. Nonetheless, this approach is not enough to ensure that the firewall configuration is completely free of errors [4].

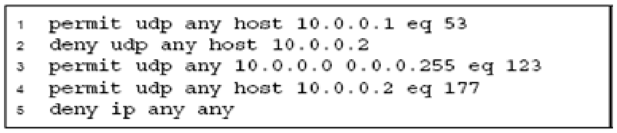

```
1  permit udp any host 10.0.0.1 eq 53
2  deny udp any host 10.0.0.2
3  permit udp any 10.0.0.0 0.0.0.255 eq 123
4  permit udp any host 10.0.0.2 eq 177
5  deny ip any any
```

Figure 1: An example of Cisco router access list. Note that the fourth rule is never matched because of the second rule

Simple packet filters usually use simple ordered lists of rules. An example of a Cisco router access list is shown in Figure 1. When a packet is received, the list is scanned from the start to the end, and the action (either “permit” or “deny”) associated with the first match is taken. If a packet doesn’t match any of the rules, the default action is “deny”. Often a “deny all” rule is included at the end of the list to make it easier to verify that a list has not been truncated. Separate lists can be specified for each network interface. The rules can use the following fields from the IP protocol header: next level protocol (e.g., TCP or UDP), source and destination IP addresses, type-of-service, and precedence. In addition, some fields for upper level protocols, such as TCP and UDP port numbers can be used. For a more complete discussion of the syntax of the rules used by Cisco routers, see [8].
A second approach in a configuration set, two rules are in conflict when the first rule in order matches some packets that match the second rule, and the second rule also matches some of the packets that match the first rule. This approach is very limited since it does not detect what we consider serious misconfiguration errors, redundancy and shadowing of rules [9].
Latest approach goal is to find minimum set of rules that cause the security policy don’t change by Detection of shadowing rules and removes it, Detection of redundancy rules and solves it and Test completeness of security policy and ensure that don’t change Disadvantages of latest approach are Inaccurate results, Number of algorithms used to perform functions, Time is huge, Performance is low

2.3 IPv4 and class C

In our work, IPv4 will use to represent source and destination the original designers of TCP/IP defined an IP address as a 32-bit number IP addresses are stored as binary numbers, they are usually displayed in human-readable notations, such as 208.77.188.166. IPv4 addresses are normally written in a format known as "dotted decimal notation". In this format, each byte of the 4-byte address is expressed as a decimal (base 10) number (i.e. 0 to 255). The four decimal numbers are separated by "dots" or "periods"[7]

C
192-223
254
NETID
1 1 0
3 Octets
Figure 2:- graphical representation of class c

Class C addresses, indicated by two 1s followed by a 0 in the first three bits of the address, are intended for small subnetworks. Class C addresses have a 24-bit NETID and an 8 bit HOSTID, permitting over two million possible network addresses. The first number of a Class C address always falls in the range 192 through 223 as Figure 2.

3. Range Algorithm

In this paper, range algorithm will work on IPv4, class c, on last byte on source IP and destination IP that mean only that all IP addresses belongs to only one network That means don’t care about first three bytes, work only on last byte Last byte means from [0- 255], that we will exclude 0 “represent network itself “and 255 “represent broadcast “Now working on range [1-254] on last byte
Pervious approaches to solve conflict and overlap problems leads to inaccurate result that cause inaccurate and weak system, division of IPv4 leads to detect redundant rules easier and inconsistent rules that has same source and destination different action, Range algorithm lead to this because all rules compared to same divided IPv4 Division process ” divide firewall rules on basis of range “, Detecting process “ detect redundant, overlap, inconsistency rules depend on hybrid comparison and intersection modules ,Test correctness “ test that result set of rules in independency case by drawing result that means free of inconsistency and achieving best solution “
In this paper our main objective is the discovering of both shadowing and redundancy Errors inside an initial set of filtering rules R. Such a detection process is a way to alert the security Officer in charge of the network about these configuration errors, as well as to remove all the useless Rules in the initial firewall configuration the data to be used for the detection process is the following. A set of rules R as a dynamic linkedlist of initial size n, where n equals count(R), and where each element is an associative array with the strings condition, decision, shadowing, and redundancy as keys to access each necessary value. To simplify, assume one can access a linked-list through the operator Ri, where i is the relative position regarding the initial list size count(R), also assume one can add new values to the list as any other normal variable does (element value), as well as to remove elements through the addition of an empty set (element ;). The internal order of elements from the linked-list R keeps with the relative ordering of rules. In turn, each element Ri [source] is an indexed array of size p containing the set of source conditions of each rule; each element Ri [destination] is an indexed array of size p containing the set of destination conditions of each rule; each element Ri [decision] is a Boolean variable whose values are in accept; deny For simplicity detection process and the removal of misconfiguration split in five different processes.
Thus, first divided IP function (Algorithm 1), divide ipv4 address whose input is Range specified in algorithm, using ceiling function that approximate to the largest integer and

comparing result to our limit 254, Second conversion function (Algorithm 2), who's input is the initial set of filtering rules, and output is conversion R by extract last byte from source and destination, convert decision to be 0 and 1 Third division function (Algorithm 3), who's input is extracted source and destination of R, in this algorithm source and destination of each rule will be compared to divided IPv4 result from algorithm 1 and put result in division table .Fourth, detection function(Algorithm 4), is recursive whose input is R from division table and take each rule and compare to other rules , that if there is intersection between source and also destination between rules with same decision then extract consistent rules by comparing if there is intersection between source and also destination between rules with different decisions then extract consistent and inconsistent between rules by comparing , else add two rules directly to consistent , because no intersection between source or destination that means two rules applied to different destination or two different sources , the output of the main detection function is the set which results as a transformation of the initial set R. This new set is equivalent to the initial one, R, and all its rules are completely disjoint. Therefore, the resulting set is free of both redundancy and shadowing of rules, as well as any other possible configuration error Fifth, (Algorithm 5), test completeness by drawing result of consistent rules and if drawing is independent so, achieving to best case of independency,

3.1 Division Process

Algorithm 1: Divided ip (Range)

```
Begin
    CountRange ← ceiling (254/Range)
    Assign value one to variable j
    Assign value zero to variable div 2
    For I ← 1 to CountRange
    If (div2 < 254)
       do
          Divide (div1, div2) ←add (j, j+Range)
          j =j+ Range+1
       End
    Else div2 equal 254
End
```

Algorithm 2: Conversion(R)

```
Begin
    For i←1to (count(R))
    do
       Ri [Source] ← extract range of last byte of source
       Ri [Destination] ← extract range of last byte of destination
       Ri [decision] ← extract decision part and convert
       Deny←0, Accept←1
       Division (Ri [Source], Ri [Destination])
    End
End
```

Algorithm 3: Division (A, B)

```
Begin
    For i←1 to count(R)
    do
       Divided(R) ←divide R by comparing A to divided ip
       Divided(R) ←divide R by comparing B to divided ip
    End
End
```

3.2 Detection Process

Algorithm 4: Detection(R)

```
Begin
    Repeat
       do
          For i←1 to (count(R) -1)
             do
                For j←i+1to (count(R))
                   do
                      If ((Ri [ source ]∩ Rj[ source] ≠ Ø) ∨(Ri [destination ]∩ Rj[ destination ] ≠ Ø))∧(Ri [decision] = Rj [decision] )
                      Then compare Ri and Rj
                      Consistent[Rcons]←extract consistent part
                      Else If ((Ri [ source ]∩ Rj[ source] ≠ Ø)∧(Ri[destination]∩Rj[destination] ≠ Ø)∧(Ri [decision] ≠ Rj [decision] )
                      Then compare Ri and Rj
                      Consistent[Rcons]←extract consistent part
                      Inconsistent[Rincons]←extract inconsistent part
                   Else
                      Consistent [Rcons] ← add (Ri, Rj)
                   End
             End
          End
```

```
        End
          Until no inconsistency detect
          If (Inconsistent [Rincons] ≠ Ø)
          Then "warning inconsistency occurred "
          Testcompleteness (Consistent [Rcons])
  End
```

3.3 Test Completeness & correctness

Algorithm 5: Testcompleteness(C)

```
    Begin
        For i←1 to (count(C))
            do
                Drawcons←Draw Ci
            End
        If (∩ Drawcons = Ø)
        Then "consistent reached
        Else "warning still inconsistent"
        End
    End
```

3.4 Complete Algorithm

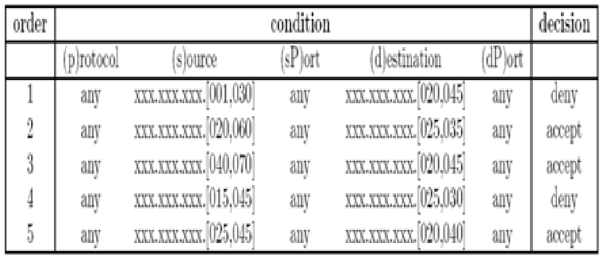
```
CompleteDetection(R)
    Begin
        Srange=Drange=Range
        Divided ip (Range)
        Foreach set of firewall rules
      do
        Conversion(R)
        Detection(R)
      End
    End
```

3.5 Applying the Algorithms

3.5.1 Applying division process

Example:-

Table 1 : Example of a set of filtering rules with five condition attributes



order	condition					decision
	(p)rotocol	(s)ource	(sP)ort	(d)estination	(dP)ort	
1	any	xxx.xxx.xxx.[001,030]	any	xxx.xxx.xxx.[020,045]	any	deny
2	any	xxx.xxx.xxx.[020,060]	any	xxx.xxx.xxx.[025,035]	any	accept
3	any	xxx.xxx.xxx.[040,070]	any	xxx.xxx.xxx.[020,045]	any	accept
4	any	xxx.xxx.xxx.[015,045]	any	xxx.xxx.xxx.[025,030]	any	deny
5	any	xxx.xxx.xxx.[025,045]	any	xxx.xxx.xxx.[020,040]	any	accept

In Table 2 result of applying algorithm 1 for range specified 15 so divided IPV4 on this base is shown in this Table. by applying algorithm 2 as shown in Table 3, taking rules and rebuild rules in table that contains source [Si, Sj], destination [Di, Dj] and decision [Aij] and called initial table.

Table 2: result of applying algorithm 1 with Range =15

1	15
16	30
31	45
46	60
61	75
76	90
91	105
106	120
121	135
136	150
151	165
166	180
181	195
196	210
211	225
226	240
241	254

Table 3: result of applying algorithm 2

/*construct initial table */				
Si	Sj	Di	Dj	Aij
1	30	20	45	0
20	60	25	35	1
40	70	20	45	1
15	45	25	30	0
25	45	20	40	1

By applying algorithm 3 as shown in Table 4, taking Table 3 and divide source [Si, Sj] on basis of range and divide destination [Di, Dj] on basis of range, in this example range defined is 15.

Table 4: result of applying algorithm 3

Define Srange=Drange=Range = 15

Si	Sj	Di	Dj	Aij
1	15	20	30	0
1	15	31	45	0
16	30	20	30	0
16	30	31	45	0
20	30	25	30	1
20	30	31	35	1
31	45	25	30	1
31	45	31	35	1
46	60	25	30	1
46	60	31	35	1
40	45	20	30	1
40	45	31	45	1
46	60	20	30	1
46	60	31	45	1
61	70	20	30	1
61	70	31	45	1
15	30	25	30	0
31	45	25	30	0
25	30	20	30	1
20	30	31	40	1
31	45	20	30	1
31	45	31	40	1

3.5.2 Detection process

By applying algorithm 4 as shown In Table 5, taking Table 4 and apply hybrid intersection and comparisons sequence modules recursively that result is consistent table that contains only consistent rules that free of any conflict or inconsistency

Table 5: result of applying algorithm 4

/*construct final consistent table that contains only consistent rules */

Si	Sj	Di	Dj	Aij
1	19	20	45	0
20	24	20	24	0
20	24	36	45	0
20	30	40	45	0
31	45	20	24	1
31	45	31	40	1
40	45	41	45	1
46	70	25	45	1

In Table 6, taking Table 4 and apply hybrid intersection and comparisons sequence modules recursively that result is inconsistent table that contains only inconsistent rules that cause inconsistency problems.

Table 6: result of applying algorithm 5

/*construct final inconsistent table that contains only inconsistent rules that cause conflict and overlap */

Si	Sj	Di	Dj	Aij
20	45	25	30	0
20	45	25	30	1
25	30	20	40	0
25	30	20	40	1
31	45	25	30	0
31	45	25	30	1

/ * resulting rules * /
R1 :(s ∈[1, 19] ∧d ∈ [20, 45]) → deny
R2 :(s ∈ [20, 24] ∧d ∈ [20, 24]) → deny
R3 :(s ∈ [20, 24] ∧d ∈[36, 45]) → deny
R4 :(s ∈ [20, 30] ∧d ∈[40, 45]) → deny
R5 :(s ∈[31, 45] ∧d ∈[20, 24]) → accept
R6 :(s ∈ [31, 45] ∧d ∈[31, 40]) → accept
R7 :(s ∈ [40, 45] ∧d ∈[41, 45]) → accept
R8 :(s ∈ [46, 70] ∧d ∈[25, 45]) → accept

Figure 3: consistent resulting rules after applying range algorithm in Figure3, rebuild table 5 in form of rule structure that contains only consistent rules.

3.5.3 Test completeness

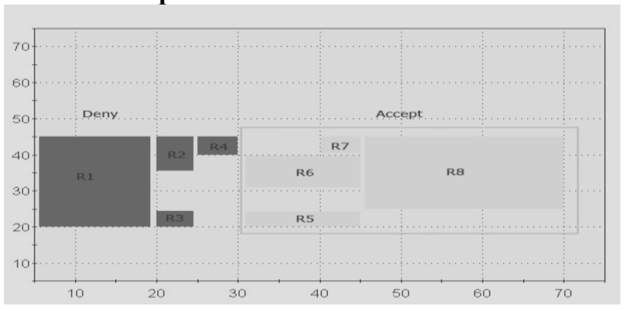


Figure 4: result of Table 5 construct final consistent table

By applying algorithm 5 as shown in Figure 4 of resulting rules that indicate achieving best case for inconsistency detection, that no overlap each rule is separated from other rules that means shadowing and redundancy our goal problems solved using this new technique

4. Conclusions and Future Work

A firewall is a system or group of systems that enforces an access control policy between two networks. The actual means by which this is accomplished varies widely, but in principle, the firewall can be thought of as a pair of mechanisms: one that exists to block traffic, and the other, which exists to permit traffic. Probably the most important thing to recognize about a firewall is that it implements an access control policy. If you don't have a good idea of what kind of access you want to allow or to deny, a firewall really won't help you. It's also important to recognize that the firewall's configuration, because it is a mechanism for enforcing policy, imposes its policy on everything behind it. Administrators for firewalls managing the connectivity for a large number of hosts therefore have a heavy responsibility and many problems occurred when configuration of firewall systems so in this paper new algorithm introduced that can guide to construct firewall systems free of inconsistency that remove inconsistent rules. In this paper range algorithm solve many problems in latest approach are the most important Problem Inaccurate results that solved and reach the best case " independency case ", so more accurate results can achieve using range algorithm Some other advantages of our approach are the following. First of all, our transformation process verifies that the resulting rules are completely independent between them. Otherwise, each redundant or shadowed rule considered as useless during the process is removed from the configuration. On the other hand, the discovering process provides an evidence of error to the administration console. This way, the security officer can check whether the security policy is consistent, in order to verify the correctness of the process. The complete independence between rules, moreover, enables the possibility to perform a second rewriting of rules in a positive manner " final consistent table " or in a negative manner " final inconsistent table " After performing this second transformation, the security officer will have a clear view of the accepted traffic or the rejected traffic. Our future research plan includes detecting inconsistent firewall rules importance using hybrid rough sets and range algorithm using importance rule to detect importance of each inconsistent rule that higher importance rule probability cause first to execute, evaluation of range algorithm related to relation between range and processing time, and extending the proposed techniques to handle class B and class A.

References

[1] A Mayer, A Wool, E Ziskind, 2000. "Fang: A Firewall Analysis Engine." IEEE SYMPOSIUM ON SECURITY AND PRIVACY

[2] B. Hari, S. Suri and G. Parulkar. , March 2000 "Detecting and Resolving Packet Filter Conflicts." Proceedings of IEEE INFOCOM'00

[3] Cuppens, F., Cuppens-Boulahia, N., and J. Garcia-Alfaro. November 2005, Misconfiguration Management of Network Security Components. In Proceedings of the 7th International Symposium on System and Information Security, Sao Paulo, Brazil

[4] Frédéric Cuppens, Nora Cuppens-Boulahia and Alexandre Miège, 2004 and formal approach to specify and deploy a network security policy. In Second Workshop on Formal Aspects in Security and Trust, pages 203-218

[5] E. Al-Shaer and H. Hamed. March 2003 "Firewall Policy Advisor for Anomaly Detection and Rule Editing." Proceedings of IEEE/IFIP Integrated Management Conference (IM'2003)

[6] Fr ́ed ́eric Cuppens, Nora Cuppens-Boulahia, and Joaqu ́ınGarc ́ıa-Alfaro, December 2007 Detection of Network Security Component Misconfiguration by Rewriting and Correlation, Universitat Autonoma de Barcelona, page 1-3

[7] http://en.wikipedia.org/wiki/IP_address

[8] Kent Hundley and Gil Held. Cisco Access Lists Field Guide. McGraw-Hill, March 2000

[9] Liu, A. X., Gouda, M. G., Ma, H. H., and Ngu, A. H. (2004). Firewall Queries. In Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS-04), pages 197-212

[10] P. Eronen and J. Zitting, November 2001 "An Expert System for Analyzing Firewall Rules." Proceedings of 6th Nordic Workshop on Secure IT-Systems (NordSec 2001),

[11] Pozo4, S., Ceballos, R., Gasca, R.M, 2008 "Polynomial Heuristic Algorithms for Inconsistency Characterization in Firewall Rule Sets". 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE). Cap Esterel, France. IEEE Computer Society Press.

[12] Tung Tran, Ehab Al-Shaer, and Raouf Boutaba, November 11-16, 2007 Firewall Security Policy Visualization and Inspection Proceedings of the 21st Large Installation System Administration Conference (LISA '07) page 1-16

Hamdy N.Agiza Obtained his B. Sc. 1976, in Mathematics "Excellent". Faculty of Science, Mansoura University, Egypt. M. Sc. 1982, in Pure Mathematics, Faculty of Science, Mansoura University, Egypt. Entitled "Study of Some Tauberian Conditions for Some Methods of summability". Ph. D. 1987, in applied Mathematics, Heriot Watt University, Edinburgh, Scotland U.K. entitled "A Numerical and Theoretical Study of Solutions to a Damped Nonlinear Wave Equation". He worked as Demonstrator in Mathematics Department, Mansoura University, 1976- 1981. Assistant Lecturer in the same Faculty.1981 - 1983. Post Graduate stude Heriot-Watt University (UK) 1983 - 1987. Lecturer in Mansoura University1987-1999. Assistant Professor in Mansoura University 1999-2005. Professor in Mansoura University 2005-Now.

Elsayed Radwan B. Sc. Student, 1992, Mathematics department (Statistics and Computational Sciences), Faculty of Science, Mansoura University with excellent degree M.Sc. 1999 "On Recent Technique for Solving Stockastic Multi-Objective Programming Problem" Egypt, University of Mansoura, Faculty of Science, Math. Department (Statistics and Ph.D. 2006 "Increasing Cellular Neural Template Robustness by New Artificial Intelligence Techniques" Japan, Toin University of Yokohama, Faculty of Engineering, Tazaki lab future research plans are to develop a new hybrid model of Fuzzy Cellular Neural Networks and Rough Sets in Image enhancement. The future work should study the sensitivity analysis of the new fuzzy template

Ahmed Farouk Obtained his B. Sc. 2006 in computer sciences "very good". Faculty of Computer Sciences and Information System, Mansoura University, Egypt. Premaster in Computer Sciences 2007, Faculty of Computer Sciences and Information System, Mansoura University, Egypt. Prepare master now in Computer Sciences specialized in network security related to firewall configuration. Now working as network engineer in Etisalat Company as I have Cisco Certified Network Professional(CCNP). My future research will focus on network security, rough sets, and artificial intelligent, networking information security.