ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCED STRATEGY OF DEPLOYMENT FIREWALL POLICIES

[1]F.Bezzazi, [2]A.KARTIT, [3]M. EL MARRAKI, [4]D.ABOUTAJDINE
LRIT Unité associé au CNRST (URAC29)
Faculty of Sciences
University Mohammed V-Agdal
B.P.1014 RP, Rabat, Morocco
E-mail: [1]bezzazi.fadwa@gmail.com, [2]alikartit@gmail.com

ABSTRACT

Firewall is one of the most widely utilized component on any network architecture, since that a deployment is a very important step to turn the initial policy to a target policy. This operation must be done without presenting any risks or flaws. Much research has already addressed the specification of policies, conflict detection and optimization, but in our paper we will focus on researches that talk about strategies for the security of policy deployment, some researchers have proposed a number of algorithms to solve this problem, we will discuss one of these algorithm then we propose an amelioration of this strategy. Our experimental results show that the new version of algorithm is very fast and can be used safely even for deploying very large policies.

Keywords: *Policy Deployment (PD), Firewall Policy (FP), Network Security (NS).*

1. INTRODUCTION

Firewall are devices or programs controlling the flow of network circulating between hosts or networks that use different security postures, most firewalls were deployed at network perimeters. This does not provide sufficient protection, because it could not detect all cases and types of attacks as well as attacks sent by an internal host to another are often not pass through network firewalls. Because of these and other factors, network designers now often include firewall functionality at places other than the network perimeter to provide an additional layer [1]. One of the functions of the firewall is to allow the establishment of some rules to determine which traffic should be allowed or blocked on your private network. Those rules do essentially (i) permit the connection (enable), (ii) block the connection (deny) [1]. Many firewall management tools such as Cisco Security Manager\cite{remazeilles2009securite}, Juniper Networks' Netscreen Security Manager[4] , and Check Point SmartCenter [8] have been developed to make the work easy for the administrators.

When deploying firewall policies, four goals must be achieved, for this, a management tool has to respect following characteristics [2]: Correctness, Confidentiality, Safety, and Speed [5].

Correctness: A deployment is correct if it successfully implements a policy on a firewall [7], i.e. If it can transform any initial policy to a target policy. This characteristic is very important for any deployment.

Confidentiality: A deployment must respect the confidentiality of information owing between firewall and management tools because of the sensitive nature of this information transmitted during the deployment [7]. For this, the communication has to be secured by using encrypted communication protocols such as SSH and SSL [9].

Safety: A deployment is safe if no legal packet is refused and no illegal packet is allowed during the deployment.

Speed: It's very important that the policy of the firewall is deployed in very short time, to avoid any suspicious traffic. So deployment must be done in the shortest time in order to be applied on very important policies. That's why the use of a little number of commands is very required to reduce the complexity and so the running time of the algorithm.

In this paper we focus on type II policy editing language .We will show how far the proposed algorithm called "greedy2phase" can't solve all cases, then propose a correct algorithm which can replace any initial policy by a target one, and also examine efficiency of both algorithms by evaluating their performances to show how far the new solution is more efficient and gives good results than the old one.

2. FIREWALL BACKGROUND

A firewall is generally placed at the borderline of the network to act as the Access Controller for all incoming and outgoing traffic. It's basically the first line of defense for any network. The main aim of this component is to keep unwanted packets from browsing your network [10]. It threats have gradually moved from being most prevalent in lower layers of network traffic to the application layer, which has reduced the general effectiveness of firewalls in stopping threats carried through network communications. However, firewalls are still needed to stop the significant threats that continue to work at lower layers of network traffic. Firewalls can also provide some protection at the application layer, supplementing the capabilities of other network security technologies.

Analysis of network traffic differs depending on the type of firewall, as well authorization or block specific instances is done by comparing the characteristics to existing policies.
Each type of firewall must essentially understand the capabilities of this latter, policy design and firewall technology acquisition that effectively meet the needs of an organization, and in order to protect the flow network traffic.

3. POLICY DEPLOYMENT

To keep the network in a high level of security, administrators or management tools must change the security policy adopted in order to replace the current policy with a new one that meets the new requirements. That is what we also called a policy deployment. Deployment of a firewall policy is the process by which we move from initial policy I to another target policy T. The changes of a firewall policy are done according to the changes that the administrator needs to do like add or remove a network element, also allowing an external network to connect or block it.

4. DEPLOYMENT EDITING LANGUAGE

To deploy a user's target policy, a management tool sends editing commands to transform the firewall's current policy and make it understandable by the firewall. The administrator will need a language to be able to build a firewall and then run effectively in accordance with the characteristics mentioned previously.

Numbers of commands compose this language but the majority of firewall use [5]:

(app r) appends rule r at the end of R
(del r) deletes r from R
(del i) deletes the rule at position i from R
(ins i r) inserts r at position i
(mov i j) moves the ith rule to the jth position in R

Where r stands for a rule, and i and j are position numbers. Some firewalls use a set of these commands while others use a different set of commands. This set of commands is called firewall's policy editing language. Several type of editing language exists on the market but the most representative is type I and type II policy editing languages.

Type I (resp. II) deployment is a deployment that use only type I (resp. type II) command. We classify the policy editing languages into two representative classes [6]: Type I and Type II.

Type I Editing: Type I editing supports only two commands, append and delete. Command (app r) appends a rule r at the end of the running policy R, unless r is already in R, in which case the command fails. Command (del r) deletes r from R, if it is present. As Type I editing can transform any running policy into any target policy [6], therefore it is complete. Older firewalls and some recent firewalls, such as FWSM 2.x [9] and JUNOSe7.x [6], only support Type I editing.

Type II Editing: Type II languages allow random editing of firewall policy. It supports three operations: (ins i r) inserts rule r as the ith rule in running policy R, unless r is already present; (del i) deletes ith rule from R; (mov i j) moves the ith rule to the jth in R position. Type II editing can transform any running policy into any target policy without accepting illegal packets or rejecting legal packets, therefore it is both complete and safe. It is obvious that for a given set of initial and target policies, a Type II deployment normally uses fewer editing commands than an equivalent Type I deployment.

5. OLD VERSION OF THE ALGORITHM

Algorithm 2 Greedy 2-Phase Deployment

```
1. TwoPhaseDeployment (I, T) {
2. /* algorithm to calculate a safe type II
deployment */
3. /* to transform firewall policy I into T */
4.
5. /* Phase 1: insert and move */
6. inserts ← 0
7. for t← 1 to SizeOf(T) do
8. if T[t] ⊄ I then
9. IssueCommand(ins t T[t])
10. inserts ← inserts + 1
11. else
12. IssueCommand( mov IndexOf(T[t] , I)+inserts t)
13.
14. /* Phase 2: backward delete */
15. for i ←SizeOf(I) down to 1 do
16. if I[i] ⊄ T then
17. IssueCommand( del i + inserts)
18. }.
```

Defect of the algorithm: Based to [6], TwoPhaseDeployment is a correct and safe algorithm. However t it does not give correct result even for a simple deployment. Like when we move a rule up and displace it before another rules already moved we can shift all the rules that came after this rule and the order of rules in the final policy will not be the same as in the target policy T. Therefore, the deployment will not respect the characteristics of an efficient deployment; we say in this case that the deployment is unsafe.

The move of rules of R in this algorithm may change the order of those rules and then, produce a policy different then the target policy.in the first phase of the algorithm we traverse the policy T starting from the beginning and we compare each rule with rules in policy I, if it does not exist already in I then, we insert it in the correct position in the running policy R according to its position in T, if it already appears in I so we have to move it at the right place in R, in this case we have two possibilities[10]: move the rule r down or move it up. In the first case we have no problem we move the rule r normally at the right position in R, but in the second one, it can cause a shift for the rules that were already moved and have the position over the rule r and so that for all the rules that comes after, thus, the order of rules in the final policy may be different than the order in the target policy. Because of the incorrect order of rules in the result of the first phase, in the second phase some rules that exit in T are deleted, or some rules that does not exist in T still exist in the final policy. So we say that this deployment is unsafe and not efficient.

It is claimed in \cite{zhang2007safety} that GreadyTwoPhaseDeployment is correct and safe. However, it can be shown that it is not correct even for very simple deployments. Consider the application of GreadyTwoPhaseDeployment to I and T given in Figure 1.(a):
To make sure that is give a good result, we apply the algorithm on the same example above.

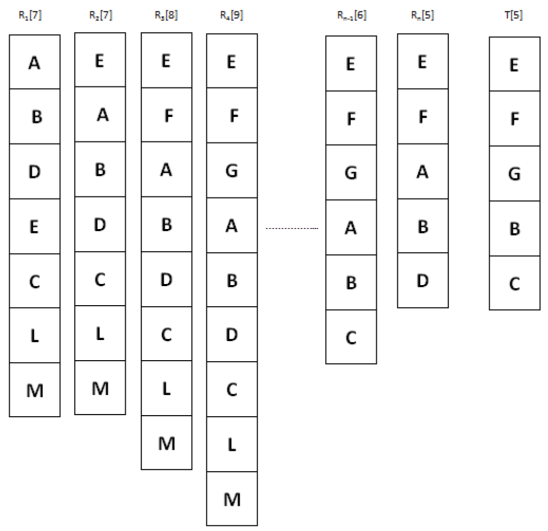


Figure 1: NewGreadyTwoPhase Running example

6. OUR CONTRIBUTION FOR THIS PROBLEM

We provide a new greedy two-phase algorithm, named EnhancedTwoPhaseDeployment, to calculate a safe type II deployment for policies I and T .This algorithm got also two phases; In the first phase, the algorithm inserts the rules of T at the beginning of the running policy R. When a rule to be inserted is already in R, it gets moved up to the right position instead according to its position in the hash table H. In the second phase, all rules that are in I but not in T are deleted starting from the last rule in table H, if a rule does not figure in T we delete it from R. This is described in Algorithm 2 (new release).

Algorithm 2 Greedy 2-Phase Deployment (New Release)

```
1: TwoPhaseDeployment (I,T){
2: /* algorithm to calculate a safe type II
deployment */
3: /* to transform firewall policy I into T */
4: /* Phase 1: insert and move */
5: H ← I
6: inserts ← 0   pos ← 0
7: for t ← 1 to SizeOf(T) do
8: if T[t] ⊄ I then
```

```
   H (ins t T[t])
9: IssueCOMMAND(ins t T[t])
10: inserts ← inserts + 1
11: else
12: pos =IndexOf(T[t],H)
13: H( mov pos t)
14: IssueCOMMAND( mov pos t)
15: end if
16: end for
 /* Phase 2: backward delete */
17: for i = SizeOf(H) down to 1 do
18: if H[i] ⊄ T then
19: IssueCOMMAND(del i)
20: end if
21: end for }.
```

To make sure that is give a good result; we apply the algorithm on the same example above.

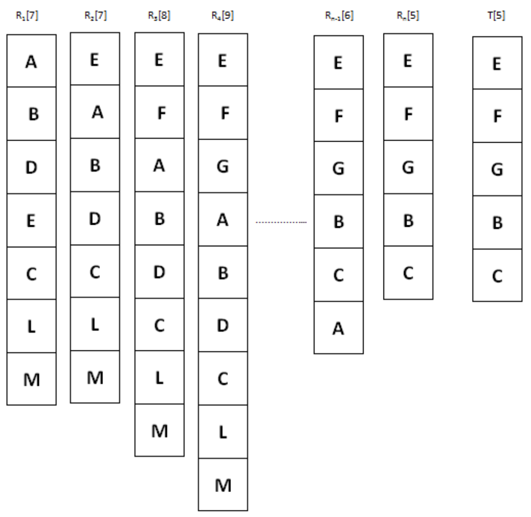


Figure 2: EnhancedGreadyTwoPhase Running example

When we apply the new version of the algorithm it's clear that the result given is correct, since the correctness is respected, so we can say that the algorithm is most efficient than the old one.

In order to make sure that this algorithm gives perfect results, we apply this algorithm on several cases, to test its efficiency, for this, we work with different size of initial and target policy.

We have in figure 3, the size of I is smaller than the size of T, while T and I have a rule in common. In figure 4 policy I and policy T have the same size, in this example policy T is a shuffle of policy T , on the other side figure 5 I and T does not have any rule in common. And the figure 6 we have sizeof(I)=sizeof(T) while I is a shuffle of T.

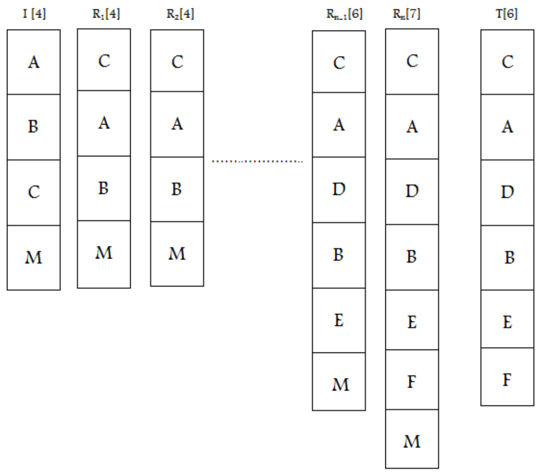


Figure 3: EnhancedGreadyTwoPhase Running example

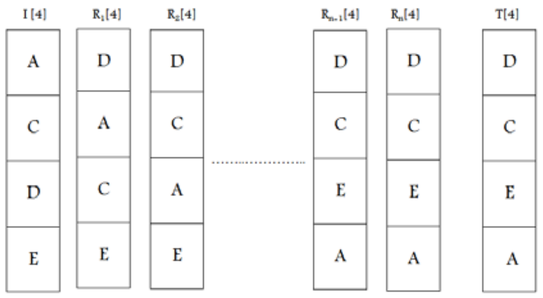


Figure 4: EnhancedGreadyTwoPhase Running example

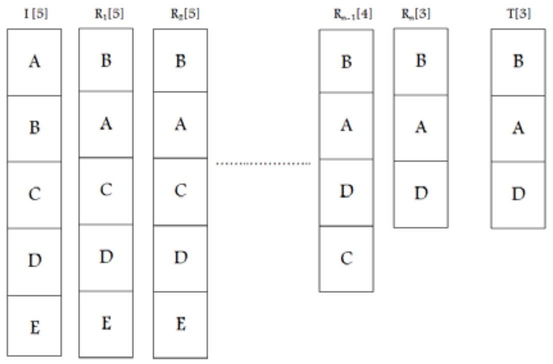


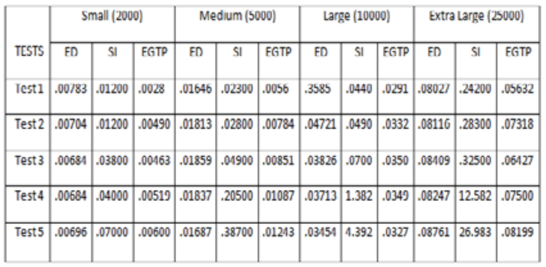
Figure 5: EnhancedGreadyTwoPhase Running example

To evaluate the performance of ENHANCEDTWOPHASEDEPLOYMENT, we try to follow the same set of test cases as in [6] and [5] Therefore we use four firewall policies with 2000, 5000, 10000, and 25000 rules. We perform five different tests For each policy.The aim of the deployment is to convert initial policy to the target policy, these tests requires respectively 10, 500, 1000, 60%, 90% commands for test 1,test 2, test 3, test 4 and test 5. Note that these percentages are taken from the initial policy. The algorithm is implemented in C++, and all tests are performed on

Sony VAIO E Series VPC-EB1E1E/WI - Core i3 330M -2.13 GHz processor - and 4 Go of RAM.

The results of each test on policies 1-4 are given in the table below. The time taken by ENHANCEDTWOPHASEDEPLOYMENT is specified in the column GTP, while the column SI, ED specifies the total time taken respectively by EFFICIENTDEPLOYMENT and SANITIZEIT algorithm given in [6] and [5] for computing a safe deployment. All times are represented in seconds.

Table 1: Results of Experiments (in seconds)



TESTS	Small (2000)			Medium (5000)			Large (10000)			Extra Large (25000)		
	ED	SI	EGTP	ED	SI	EGTP	ED	SI	EGTP	ED	SI	EGTP
Test1	.00783	.01200	.0028	.01646	.02300	.0056	.3585	.0440	.0291	.08027	.24200	.05632
Test2	.00704	.01200	.00490	.01813	.02800	.00784	.04721	.0490	.0332	.08116	.28300	.07318
Test3	.00684	.03800	.00463	.01859	.04900	.00851	.03826	.0700	.0350	.08409	.32500	.06427
Test4	.00684	.04000	.00519	.01837	.20500	.01087	.03713	1.382	.0349	.08247	12.582	.07500
Test5	.00696	.07000	.00600	.01687	.38700	.01243	.03454	4.392	.0327	.08761	26.983	.08199

Table 1 shows that the algorithm is performed EGTP in a fraction of second to move from one policy to another, and done so faster and safer than other algorithms shown in the table. As can be seen from the results for the tests (1-5), the new algorithm is run in a minimal time comparing with other algorithms such ED and SI which also belong to type II and have the same EGTP aim.

During all tests we see that the execution time change but is always optimal if we compared to other results. Regardless the importance of adopted firewall policy.

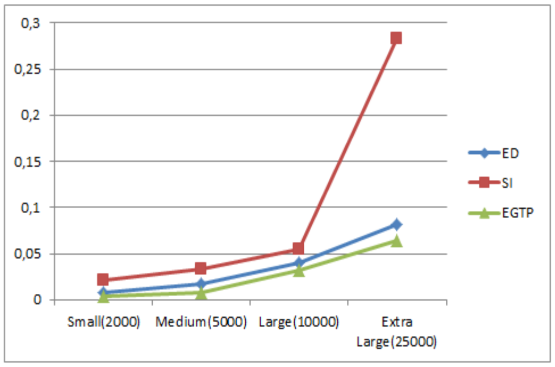


FIGURE 6: COMPARISON ED, SI, AND EGTP FOR THE AVERAGE OF THE FIVE TESTS

From the curve illustrated in Figure 6, it can be concluded that ENHANCEDTWOPHASEDEPLOYMENT is more efficient than SANITIZEIT and EFFICIENTDEPLOYMENT and the running time is close to linear. Furthermore, SANITIZEIT appears to have a polynomial running time. This effect is more notable in case of test 5 and Policy 4, where SI takes almost 27 secs to compute a deployment sequence.

7. CONCLUSION

Firewall policy deployment is a new large subject and error-prone task [10], several researchers have proposed strategies in order to update a policy while respecting the safety and efficiency criteria, but still doesn't propose an efficient one, which gives a good results in all cases.

In this paper, we discussed one of these existing strategies that contains security flaws and may therefore allow illegal packet as it can block legal one. We will study the principle of one of these strategies, for a type II language, that we applied to a simple example and show that it is unable to give a good result, but after making some changes, the result was satisfying, therefore our improvement gives better results. Finally from the table 1 and the curve we can say that our contribution is a better solution for this the deployment of any firewall policy regardless it size. That led us to say that this new proposition is optimal and could be applied to a much larger policies.

REFRENCES:

[1] Wack, J. and Cutler, K. and Pole, J., "Guidelines on firewalls and firewall policy", *DTIC Document-2002.*

[2] Kartit, A. and El Marraki, M.,"An enhanced algorithm for Firewall Policy Deployment", *International Conference on Multimedia Computing and Systems (ICMCS), IEEE 2011, pp.1-4.*

[3] Young, G. and Pescatore, J., "Magic quadrant for network intrusion prevention system appliances" *Gartner Core RAS Research Note G, 167303:1–12, 2009.*

[4] V. Remazeilles. "La sécurité des réseaux avec Cisco". *Editions ENI, 2009.*

[5] Z. Ahmed, A. Imine, and M. Rusinowitch. "Safe and efficient strategies for updating firewall policies". *Trust, Privacy and Security in Digital Business, 2010,pp. 45-57.*

[6] C.C. Zhang, M. Winslett, and C.A. Gunter., "On the safety and efficiency of firewall policy deployment". *In Security and Privacy, 2007. SP'07. IEEE Symposium on, IEEE, 2007. pp. 33-50.*

[7] El Marraki, M. and Kartit, A.," On the Correctness of Firewall Policy Deployment". *Journal of Theoretical and Applied Information Technology, ISSN vol. 3195, 2010, pp. 22-27.*

[8] Chang, C. and Zhang, B. and Lao, Z., "Method and apparatus for hybrid smart center loop for clock data recovery", *US Patent 6,526,109. 2003*

[9] Bezzazi, F and Kartit, A and Marraki, M El and Aboutajdine, D, "Optimized strategy of deployment firewall policies", *Second International Conference on Innovative Computing Technology (INTECH), IEEE 2012, pp. 46-50.*