Safe Algorithm for Firewall Policy Deployment

F. Bezzazi, A.Kartit, M. El Marraki, D. Aboutajdine
LRIT Unité associée au CNRST(URAC29)
Faculty of Sciences, CNRST. Morocco.
{fadwa, alikartit}@gmail.com, {marraki, aboutaj }@fsr.ac.ma

ABSTRACT: *The problem of modern network security becomes more and more difficult, and request security methods more efficient and adapted to current circumstances. Since the firewall is one of the most used components in any network architectures, policy deployment is a very important step to move from one policy to another without presenting any flaws. For this case, researchers have proposed several algorithms that solve this problem safely. In this paper we will see two types of policy deployment often used, study one of the type II language, and show that it can provide some security holes. We will propose some changes to transform this strategy to a new very efficient algorithm, which respect the safety criteria.*

Keywords: Firewall, Network security, Policy deployment, Strategy

Received: 10 August 2012, Revised 30 September 2012, Accepted 9 October 2012

1. Introduction

A firewall is basically the first line of defense for any network. The main aim of a firewall is to keep uninvited packets from browsing your network. It is placed at the borderline of the network to act as the Access Controller for all incoming and outgoing traffic. A firewall allows you to establish some rules to determine what traffic should be allowed in or out of your private network. Depending on the type of firewall implemented you could restrict access to only certain IP addresses or domain names, or you can block certain types of traffic by blocking the TCP/IP ports they use. Several types of firewalls exist: Packet filtering, Circuit-level gateway implementation, acting as a proxy server, Web application firewall. The large size and complexity of modern network topologies make the configuration of firewall policies more difficult, complicated and subject to errors. So that we could find firewall policies that contain 10K rules, as we could find others configured with over then 50K rules, what makes manual configuration like a mission impossible for any administrator. Those rules do essentially [6] permit the connection (enable), (ii) block the connection (deny). Many firewall management tools such as Cisco Security Manager[5], Juniper Networks' Netscreen Security Manager[7] , and Check Point SmartCenter[2] have been developed to make the work easy for the administrators.

A management tool aim is to achieve four goals by respecting following characteristics when deploying firewall policies: Correctness, Confidentiality, Safety, and Speed [1].

Correctness: A deployment is correcte if it successfully implements a policy on a firewall [3], ie 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 flowing between firewall and management tools because of the sensitive nature of these information transmitted during the deployment [3]. For this, the communication have to be secured by using encrypted communication protocols such as SSH and SSL [4].

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 language editing policy type II. We will demonstrate that the proposed algorithm called "*greedy2phase*" can't solve all cases, we will propose a correct algorithm which can replace a source policy by a target policy, and also examine efficiency of both algorithms by comparing there complexity to show how far the new solution is more efficient and give good results than the old one.

2. Deployment Policy

For several reasons an administrator needs to change the security policy adopted, and therefore it should replace the current policy with a new policy that is most appropriate to the new requirements. This operation is also called a deployment policy. So the deployment of a firewall policy is the process that allows us to move from a policy I initial to another target T. An administrator often needs to add or remove a network element, allowing an external network or blocked and therefore must often change policy with which he conducts what is called a deployment policy.

3. Deployment editing language

To make the policy understood 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. This language is composed of a number of commands but the majority of firewall use [1]:

• (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

Some firewalls support some of these commands, while others support another set of commands, Not all firewalls support all these commands. strating from this set of commands supported by the firewall we can define what we call firewall's policy editing language. the most representative language on the market are type I and type II policy editing languages. If a deployment uses only type I (resp. type II) commands, we call it a type I (resp. II) deployment. We classify the policy editing languages into two representative classes [8]: 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 [8], therefore it is complete. Most older firewalls and some recent firewalls,such as FWSM 2.x [4] and JUNOSe 7.x [8], 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 [8], 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.

4. Old version of the algorithm

Shortcoming: According to [8], TwoPhaseDeployment is a correct and safe algorithm. However we can easily show that is not correct even for a simple deployment. when we move a rule up and replace it before another rules already moved we can shift all the rules that came after and the order of rules in the final policy will not be the same as in the target policy Therefore the deployment will be unsafe and will not respect the caracteristics of an efficient deployment.

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: 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 deloyment is unsafe and not efficient.

Algorithm 1 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: /* Phase 1: insert and move */
5: inserts ← 0
6: for t ←1 to Size Of (T ) do
7:    if T [t] ∉ I then
8:       IssueCOMMAND(ins t T [t])
9:       inserts ← inserts + 1
10:   else
11:      IssueCOMMAND( mov Index Of (T[t], I) + inserts  t)
12:   end if
13: end for
14: /* Phase 2: backward delete */
15: for ← Size Of (I ) down to 1 do
16:   if I [i] ∉ T then
17:      IssueCOMMAND(del i + inserts)
18:      end if
19: end for
```

It is claimed in [8] 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).

5. Our Solution for this problem

We provide a new greedy two-phase algorithm, named NewTwoPhaseDeployment, to calculate a safe type II deployment for policies I and T. In phase 1, 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 phase 2, all rules that are in I but not in T are deleted starting we traverse the table H from the end if a rule does not figure in T we delete it from R. This is described in Algorithm 2 (new release). to make sure that is give a good result, we apply the

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

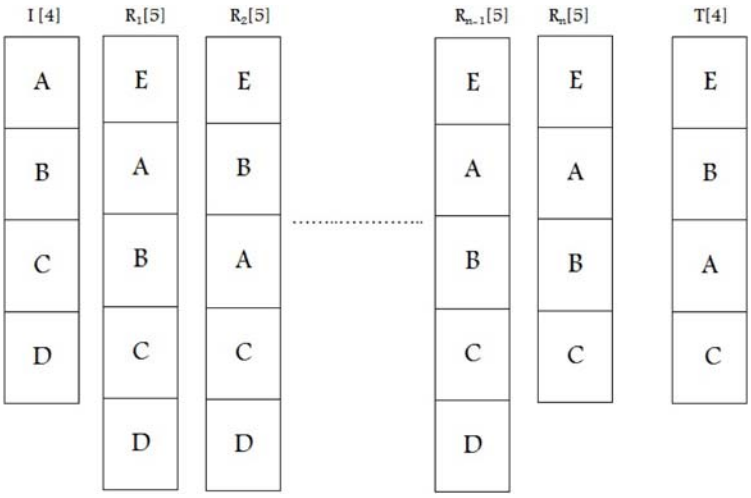

Figure 1. GreadyTwoPhase Running example

algorithm on the same example above. it's clear that the result given is correct, because the correctness is respected, so we can say that the algorithm is efficient.

However, we apply this algorithm on several cases, to test its efficiency, with different size of intial and target policy. We have

in figure 3, the size of I is bigger than the size of T, while T is a part of I. In figure 4 sizeof (I) < sizeof (T) In this example I have some rules in common with 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.

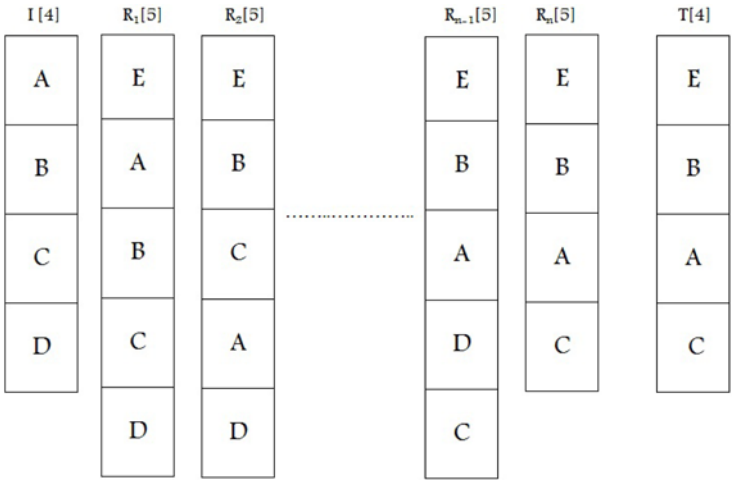

Figure 2. NewGreadyTwoPhase Running example

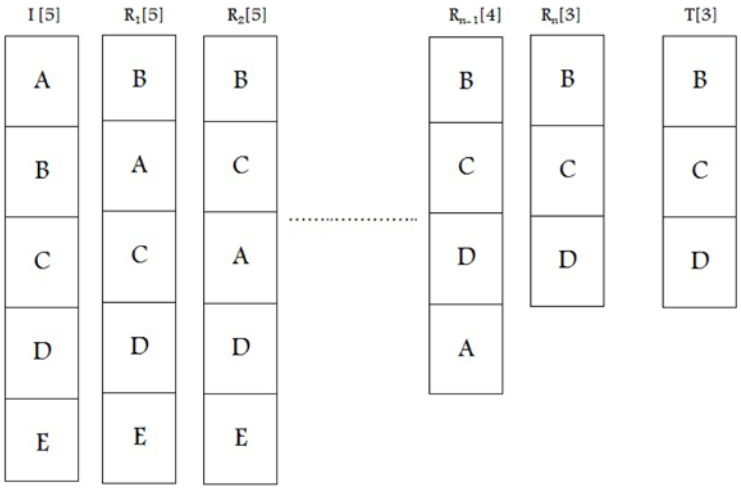

Figure 3. NewGreadyTwoPhase Running example

6. Performence Algorithms Comparison

The complexity of an algorithm is one of the utilized methods to judge if the algorithm is performanent or not according to its capacity to solve the problem in a short time by using a little number of commands ,or a smallest memory space. In this paper, we have seen two algorithms from type II of policy editing language. The main aim of these algorithms is to deploy a policy to pass from initial policy I to a target policy T while respecting all characteristics of efficient deployment. For this, we have to compare between both algorithms to decide which one is more efficient and performing, if the algorithm gives a good results and is executed in a short time using just a smallest memory space, we say that this algorithm is most efficient.

If we calculate the complexity of the old algorithm assuming that the execution time of the command ($\notin$) is done in a constant [8]. We can say that this algorithm is in order of n ie O (n). On the other hand the new algorithm, which gives better results, assuming also that the command ($\notin$) uses a constant time, is also in order of n, so that at the worst time of execution, the algorithm will be in order of n ie O (n).

So since both algorithms have the same complexity regarding time of execution, whereas the new gives better results so we can say that this new release is more efficient than the old version of algorithm.

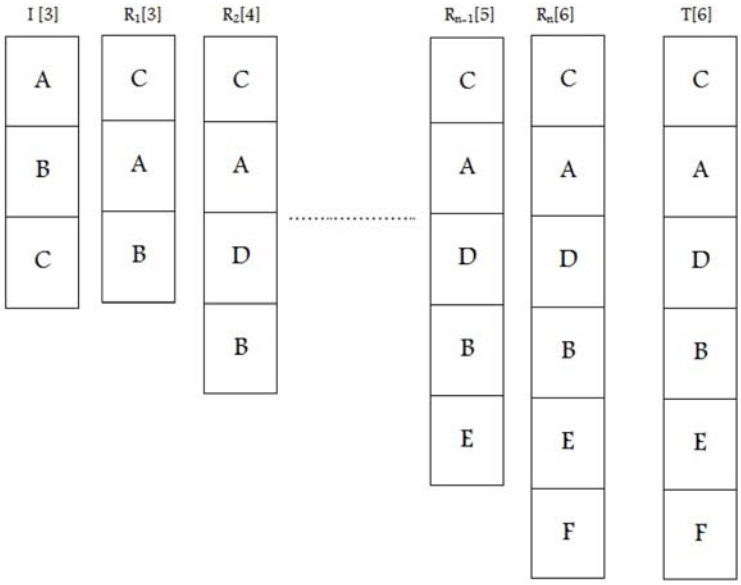

Figure 4. NewGreadyTwoPhase Running example

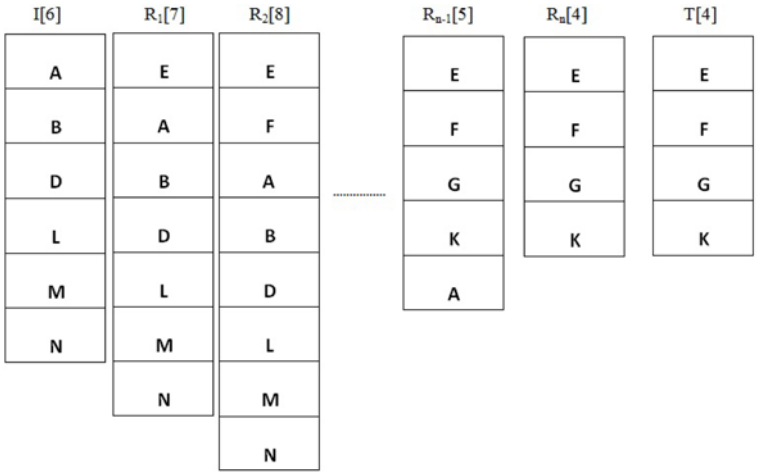

Figure 5. NewGreadyTwoPhase Running example

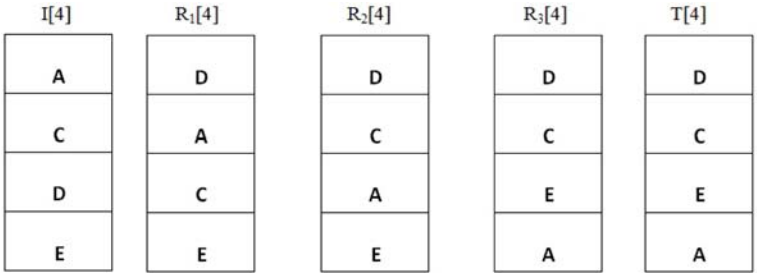

Figure 6. NewGreadyTwoPhase Running example

7. Conclusion

The deployment of a firewall policy is a new large subject and error-prone; 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 satisfactory, therefore our improvement gives better results. Finaly the comparaison of the complexity of the two algorithms show that they have the same complexity, while the new release give a better results than the old proposition. That lead us to say that this new proposition is optimal and could be applied to a much larger policies.

References

[1] Ahmed, Z., Imine, A., Rusinowitch, M. (2010). Safe and efficient strategies for updating firewall policies. Trust, Privacy and Security in Digital Business, p. 45–57.

[2] Chang, C., Zhang, B., Lao, Z. (2003). Method and apparatus for hybrid smart center loop for clock data recovery, February 25. US Patent 6, 526, 109.

[3] El Marraki, M., Kartit, A. (1817). On the correctness of firewall policy deployment. *Journal of Theoretical and Applied Information Technology*, ISSN, 3195, 22–27.

[4] Kartit, A., El Marraki, M. (2011). An enhanced algorithm for firewall policy deployment. *In*: Multimedia Computing and Systems (ICMCS), 2011 International Conference on, p. 1–4. IEEE.

[5] Remazeilles, V. (2009). La sécurité des réseaux avec Cisco. Editions ENI.

[6] Wack, J., Cutler, K., Pole, J. (2002). Guidelines on firewalls and firewall policy. Technical report, DTIC Document.

[7] Young, G., Pescatore, J. (2009). Magic quadrant for network intrusion prevention system appliances. Gartner Core RAS Research Note G, 167303, 1–12.

[8] Zhang, C. C., Winslett, M., Gunter, C. A. (2007). On the safety and efficiency of firewall policy deployment. *In*: Security and Privacy, 2007. SP'07. IEEE Symposium on, p. 33–50. IEEE.