PERFORMANCE EVALUATION OF ENHANCED-GREEDY-TWO-PHASE DEPLOYMENT ALGORITHM

Kartit Ali[1]

[1]Laboratory of Research in Informatics and Telecommunications (LRIT), Faculty of Science, University Mohammed V-Agdal, Rabat, Morocco

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 conflict detection of policies 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. In [1], we have proposed a correct algorithm for the deployment type I. But in this work we will study the performance evaluation of the new solution called "Enhanced-Two-Phase-Deployment". We show that the proposed solution is most efficient.

KEYWORDS

Firewall Policy, Network Security, Policy Deployment, Performance Evaluation.

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. 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). Its principle for operation is simple; it is a set of rules defined by an administrator based on the principle: everything that is not explicitly allowed is prohibited, which means that these rules are part of the configuration firewall must allow or dismiss an action or a data stream in order to establish or block a connection. Several firewalls deploying policies containing rules more than 20K are rare in the market, and yet we saw a firewall configured with rules for 50K. Manual configuration of these policies has clearly become an impossible task even for guru network administrators.

These rules in general [3] are: (i) accept a connection (enabled), (ii) blocks a connection (deny).
A firewall policy deployment should have following characteristics [2]: correctness, confidentiality, safety, and speed.

DOI : 10.5121/ijnsa.2013.5412

Correctness: A deployment is correct if it successfully implements the target policy on the firewall. After a correct deployment the target policy becomes the running policy. Correctness is an essential requirement for any deployment.
Confidentiality: Confidentiality refers to securing the communication between a management tool and a firewall. It's can be achieved by using encrypted communication protocols such as SSH [4] and SSL [5].

Safety: We can say that the deployment is safe if no illegal packet is accepted and no legal packet is rejected during the deployment. A naive deployment strategy may result in temporary security breaches and/or self-Denial of Service (self-DoS). Deployment safety is a challenging and new area of research.

Speed: A deployment should be done in the shortest time, so that the desired state of affairs is achieved as quickly as possible. A deployment algorithm should have a good running time, so that it is applicable even for large policies. A slow deployment is unpleasant for users and may partly defeat the purpose of deployment [2].

In this paper we focus on type II policy editing language .We will show how far the proposed algorithm called "Greedy-2-PhaseDeployment" 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 (see Figure 1). 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. It's is an ordered list of rules named ACL.

An ACL is an ordered set of rules, each rule is a statement concerning an action, which controls whether a firewall denies or allows the passage of packets based on criteria found in the header of a packet. ACLs are used to select the types of traffic to be analyzed, processed or transmitted by other means.

Packets pass through interfaces firewall or router associated with an ACL, the ACL is checked from top to bottom looking for a corresponding pattern of the incoming packet. The ACL applies one or more security policies using permit or deny an action to determine the fate of the packet. ACLs can be configured to control access to a network or subnet.

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.

The filtering decision is based on a firewall policy defined by network administrator.
It is possible to use any field of IP, ICMP, UDP, or TCP headers [2]. However, these fields are most commonly used: source port, destination port, protocol type, destination IP address and source IP address [6].

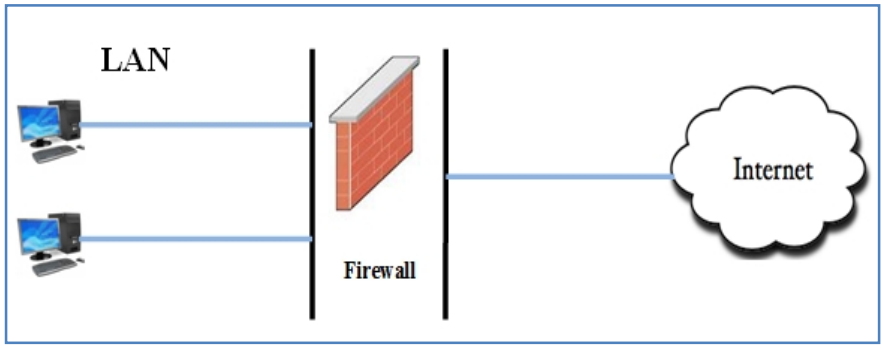

Figure1. Fire-wall architecture

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. Policy deployment is the process by which policy editing commands are issued on firewall, so that the target policy becomes the running policy. As discussed in the Introduction, a deployment must be correct and should satisfy the following three characteristics: confidentiality, safety, and speed.

3.1. Policy Editing Languages

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.

The set of commands that a firewall supports is called its policy editing language. Typically, a firewall uses a subset of the following editing commands [2]:

(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.

Policy editing languages can be classified into two representative classes [2]: Type I and Type II.

Type I Editing

Type I editing supports only two commands, append and delete. Command (app r) adds a rule r at the end of the political running R unless r is already in R, in this case, the command fails. Command (del r) removes r from R, if it’s present. As type I editing can convert any political running into any political target [2], so it is complete. Older firewalls and some recent firewalls, such as JUNOSe 7.x [8] and FWSM 2.x [7], 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 the rule r in the i^{th} rule in the running policy R, unless r is already present; (del i) removes the i^{th} rule from R; (mov i j) moves the i^{th} rule to the j^{th} position in R. Type II editing can convert any political running into any political target without rejecting legal packets or accepting illegal packets [2], therefore, it is complete and safe in the same time. It is obvious that for a given set of target and initial policies, a Type I deployment normally uses a lot of editing commands than an equivalent Type II deployment. There are some examples of Type II editing firewalls like Enterasys Matrix X [10] and SunScreen 3.1 Lite [9].

3.2. Deployment Efficiency

Deployment is more effective if it uses the minimum number of editing commands in a given language, to successfully deploy a policy target on a firewall. Therefore for a given deployment scenario, the most-efficient Type I deployment uses the minimum number of append and delete commands, similarly a most-efficient Type II deployment uses the minimum number of insert, delete and move commands. Therefore, the most-efficient deployment minimizes the overall deployment time. Deployment efficiency for Type II languages is discussed in more detail in Section 4.

3.3. Deployment Safety

A deployment is safe if no legal traffic is denied at any stage and no security hole is introduced during the deployment. A temporary security hole may permit malicious traffic to pass through the firewall that may cause serious damage to the network infrastructure.

4. TYPE II DEPLOYMENT

Type II deployment helps to modify random policy running. Therefore, for a given set of initial and target policy, a safe type II deployment uses less editing commands than a similar Type I deployment. If I and T have the same set of rules, then T can be regarded as a permutation of I. In the general case, where T has some rules that are not in I and I has some rules that are not in T, a command has to be generated to insert/delete each such rule.

4.1. Problems with Previous Algorithm

In [2], two algorithms for type II deployment are proposed. The first algorithm is a greedy two-phase Deployment called TWOPHASEDEPLOYMENT (see Algorithm 1), while the second algorithm is a most-efficient algorithm called SANITIZEIT. In this paper we interest to the first algorithm. It is claimed in [2] that TWOPHASEDEPLOYMENT is correct and safe. However, it can be shown that it is not correct even for very simple deployments. Consider the application of TWOPHASEDEPLOYMENT to I and T given in the cases bellow.

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.
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. }.
```

Example:
I = A-M-C-L-K-E
T = L-C-E-M-B-D-F-K
R = K-F-D-B-M-L-C-E

Proof:

(a) t=1 ; indexof(T(t)=L,I)=4 ; move(4,1) ; R0= L-M-C-K-E
(b) t=2 ; indexof(T(t)=C,I)=3 ; move(3,2) ; R1=L-C-K-E
(c) t=3 ; indexof(T(t)=E,I)=4 ; move(4,3) ; R2= L-C-E
(d) t=4 ; T(t)=M ins ; R3= M-L-C-E
(e) t=5 ; T(t)=B ins ; R4= B-M-L-C-E
(f) t=6 ; T(t)=D ins ; R5=D-B-M-L-C-E
(g) t=7 ; T(t)=F ins ; R6=F-D-B-M-L-C-E
(h) t=8 ; T(t)=K ins ; R7=K-F-D-B-M-L-C-E

We can clearly observe that the order of the rules is not respected, the respect of order is very important, so deployment does not meet the safety criterion. When you move a rule to a higher position that causes a shift in the positions of other rules and then at the end you get a different result from the policy target T. So deployment is not correct and does not meet the characteristics already mentioned for the effective deployment.

4.2. Our Solution for Type II Deployment

The above problems motivate us to provide a correct, safe and efficient algorithm, called ENHANCED-TWOPHASEDEPLOYMENT (see Algorithm 2).

Algorithm 2: ENHANCED-Greedy-2-Phase Deployment

```
1. ENHANCEDTwoPhaseDeployment (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 */
7. for t←1 to SizeOf(T) do
8. if T[t] ∉ I then
9. IssueCommand(ins t T[t])
11. else
12. IssueCommand( mov IndexOf(T[t] , I) 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 )
18. }.
```

The previous example can be reused to prove the truth of the change because they have two inserts at the end and since they are the last operations, they should normally lead to good positioning or insertion in the lead will disordering target.

Example:
I = A-M-C-L-K-E
T = L-C-E-M-B-D-F-K
R = L-C-E-M-B-D-F-K

Proof:

t=1 ; indexof(T(t)=L,I)=4 ; move(4,1) ; R0= L-M-C-K-E
t=2 ; indexof(T(t)=C,I)=3 ; move(3,2) ; R1=L-C-K-E
t=3 ; indexof(T(t)=E,I)=4 ; move(4,3) ; R2= L-C-E
t=4 ; T(t)=M ins(M,4) ; R3= L-C-E-M
t=5 ; T(t)=B ins(B,5) ; R4= L-C-E-M-B
t=6 ; T(t)=D ins(D,6) ; R5= L-C-E-M-B-D
t=7 ; T(t)=F ins(F,7) ; R6= L-C-E-M-B-D-F
t=8 ; T(t)=K ins (K,8) ; R7= L-C-E-M-B-D-F-K

5. PERFORMANCE EVALUATION OF THE NEW ALGORITHM

We try to follow the identical set of test cases as in [2] to evaluate the performance of Enhanced-Greedy-2-PhaseDeployment. Thus, we use four firewall policies with 2000, 5000, 10000, and 25000 rules. We perform five different tests for each policy. We implemented the new algorithm in C++ in order to test and evaluate the performance of it. All tests are performed on HP with Intel(R) Core(TM) 4 DUO CPU 3.00Ghz (2 CPUs) processor and 6GB of RAM. We use a firewall simulator that is configured to match the performance of a ASA 525 firewall and connect to it over a 100Mb Ethernet link. We run each test case 10 times and then record the average for Enhanced-Greedy-2-PhaseDeployment and SANITIZEIT algorithm combined with diff. The results of each test on policies 1-4 are shown in the table below (see Table 1). While the column SI specifies the total time taken by diff and SANITIZEIT algorithm given in [2] for computing a safe deployment, the time taken by Enhanced-Greedy-2-PhaseDeployment is specified in the column EG2PD.

Table 1. Results of Experiments (in seconds).

Tests	**Policy1 (size=2000)**		**Policy2 (size=5000)**		**Policy3 (size=10000)**		**Policy4 (size=25000)**	
	EG2PD	SI	EG2PD	SI	EG2PD	SI	EG2PD	SI
Test 1	0,0054	0,0110	0,0140	0,0216	0,0213	0,0360	0,0622	0,1750
Test 2	0,0051	0,0110	0,0169	0,0266	0,0152	0,0390	0,0630	0,1290
Test 3	0,0046	0,0360	0,0162	0,0450	0,0142	0,0533	0,0620	0,3310

Test 4	0,0045	0,0380	0,0165	0,2300	0,0135	1,1330	0,0623	9,6450
Test 5	0,00471	0,0687	0,0142	0,3280	0,1323	3,2440	0,0642	15,0660

It is obvious that Enhanced-Greedy-2-PhaseDeployment takes a fraction of second to calculate most efficient and safe deployment for policies as long as Policy 4. In addition, Enhanced-Greedy-2-PhaseDeployment generates a safe and most efficient deployment much quicker than the SANITIZEIT algorithm combined with diff. However, it might not be appropriate to directly draw conclusion for tests 2-5 as no details are given about nature of changes in [2]. For instance, consider Test 5 on Policy 4, 90% edit distance means 22600 commands need to be issued to turn initial policy to target policy. If 22,600 insert commands are required that means T has 47,600 rules, while if 22,600 delete commands are required then T has only 2600 rules. Therefore, reliable comparison can only be done if size of initial policy and target policy used in [2] is known, so that policies of same size could be used for testing Enhanced-Greedy-2-PhaseDeployment. However, Test 1 consists only 10 changes and it can be used to compare the two algorithms.

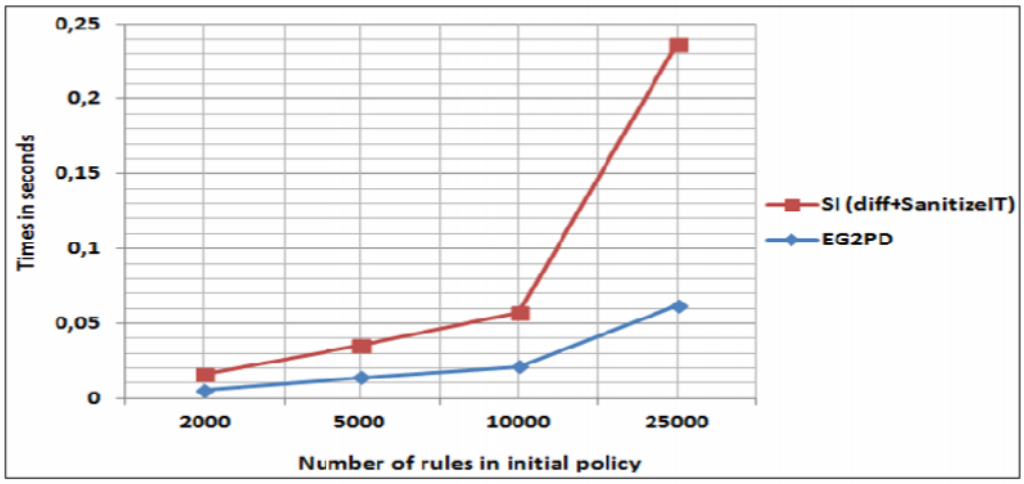

Figure 1. Comparison of Enhanced-Greedy-2-PhaseDeployment and SanitizeIT for Test 1

From the curve illustrated in Figure 2, it can be concluded that Enhanced-Greedy-2-PhaseDeployment is more efficient than SANITIZEIT 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 15 seconds to compute a deployment sequence.

6. CONCLUSION

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

In this paper, we have shown that recent approaches [2] to firewall policy deployment contain critical errors. Indeed, these approaches can introduce temporary security holes that permit illegal traffic and/or interrupt network services by blocking legal traffic during a deployment. We have proposed for type II policy editing languages the efficient and safe algorithm called Enhanced-

Greedy-2-PhaseDeployment. This algorithm is approximatively linear, most-efficient and safe. Our experimental results showed that this algorithm does not add any overhead and it is practical even for very large policies. We will work on the second algorithm called SANITIZEIT to improve it.

REFERENCES

[1] A. Kartit and M. El Marraki "On the Correctness of Firewall Policy Deployment", Journal of Theoretical and Applied Information Technology, ISSN: 1992-8645, Volume 19, n°1, pages 22 – 27, 15th September 2010.

[2] C. C. Zhang, M. Winslett, and C. A. Gunter. On the Safety and Efficiency of Firewall Policy Deployment. In SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pages 33–50,Washington, DC, USA, 2007. IEEE Computer Society.

[3] S. Karen and H. Paul, "Guidelines on Firewalls and Firewall Policy", NIST Recommendations, SP 800-41, July, 2008.

[4] T. Yl¨onen. SSH: secure login connections over the internet. In SSYM'96: Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography, pages 4–4, Berkeley, CA, USA, 1996. USENIX Association.

[5] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce, pages 4–4, Berkeley, CA, USA, 1996. USENIX Association.

[6] S. Cobb. ICSA Firewall Policy Guide v2.0. Technical report, NCSA Security White Paper Series, 1997.

[7] Cisco Security Manager. http://www.cisco.com/en/US/products/ps6498/index.html.

[8] JuniperNetwork and SecurityManager. http://www.juniper.net/us/en/local/pdf/datasheets/1100018 en.pdf.

[9] M. Englund. Securing systems with host-based firewalls. In Sun BluePrints Online, September 2001.

[10] Entrasys Matrix X Core Router. http://www.entrasys.com/products/routing/x/.

Authors

Ali KARTIT is a doctor in computer network security at the Faculty of Sciences of Rabat. His research area covers security policies of firewalls, the Intrusion detection systems (IDS / IPS) and security in the cloud environment.