Journal of Theoretical and Applied Information Technology
20th January 2014. Vol. 59 No.2

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

TOWARDS A NEW ALGORITHM MORE EFFICIENT FOR UPDATING FIREWALL POLICIES

[1]A. KARTIT, [2]Z. KARTIT, [3]M. EL MARRAKI

[1,2,3] Laboratoire de Recherche en Informatique et Télécommunications, Faculty of Sciences, University of Mohammed V-Agdal, Rabat, Morocco
E-mail: [1]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 policy requires automated tools in order to create a suitable environment for configuring or deploying safely a policy target. Most researchers are interested in detection of conflicts or the problem of optimization of policies firewall while few of them have proposed deployment strategies for two important types of edition policies. We have already proposed a correct algorithm for the deployment type I [0]. In this work we study one of these strategies that falls within the type II edition policies. We show that the proposed solution of type II edition [1] is not fully correct and lead to security vulnerabilities, and then we offer a few corrections and improvements for this type of editing.

Keywords: *Security Policy (SP), Firewall Policy Management (FPM), Security Vulnerabilities (SV), Network Security (NS)*

1. INTRODUCTION

A firewall is a physical device or logic for system protection for home computers. Due to insecurity of Internet and intranet, the use of firewalls is still a very important step. It can also serve as an interface between one or more enterprise networks to monitor and possibly block the flow of data by analyzing the information contained in the data stream. On one hand, it allows you to block and trace attacks or suspicious connections may originate from viruses or others. On the other hand, a firewall is used in many cases also to prevent the uncontrolled leakage of information to outside. 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 deployed Policies containing 10K rules are not uncommon in commercially deployed firewalls, and we have seen a firewall configured with 50K rules. Manually configuring such policies has clearly become mission impossible even for guru network administrators. These rules in general all incoming and outgoing traffic (see Figure 1).

[2] are: (i) accept a connection (enabled), (ii) blocks a connection (deny).

A firewall policy deployment should have following characteristics [1]: correctness, confidentiality, safety, and speed.

Different firewalls support different policy editing commands. The set of policy editing commands that a firewall supports is called its policy editing language. In [1], the authors classify policy editing languages into two representative classes, Type I and Type II, and provide deployment algorithms for both types of languages.

In this paper, we analyse the algorithm “Greedy - two-Phase Deployment” provided in [1] and show that this algorithm have serious flaws. We present an improved safety formalization that can be used as a basis for formulating safe deployment strategies. We provide most-efficient and safe algorithm for Type II languages.

2. FIREWALL BACKGROUND

A firewall is generally placed at the borderline of the network to act as the Access Controller for

It's basically the first line of defense for any

Journal of Theoretical and Applied Information Technology
20th January 2014. Vol. 59 No.2

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

network.

While firewalls are often discussed in the context of Internet connectivity, they may also have applicability in other network environments. For example, many enterprise networks employ firewalls to restrict connectivity to and from the internal networks used to service more sensitive functions, such as accounting or personnel. By employing firewalls to control connectivity to these areas, an organization can prevent unauthorized access to its systems and resources.

The filtering decision is based on a firewall policy defined by network administrator. A firewall policy is an ordered list of rules. A firewall rule r defines an action, typically accept or reject, for the set of packets matching its criteria. It is possible to use any field of IP, UDP, or TCP headers [1]. However, the following five fields are most commonly used: protocol type, source IP address, source port, destination IP address and destination port [5].

Any field in packet's header can be used for the matching process. However, the same five fields are most commonly used. In a packet, each of these fields has an atomic value. If all the fields of a packet p match with the corresponding fields of a rule r, then p is accepted or rejected according to the decision field of r. If p does not match to any rule in policy, then the default match-all rule is applied.

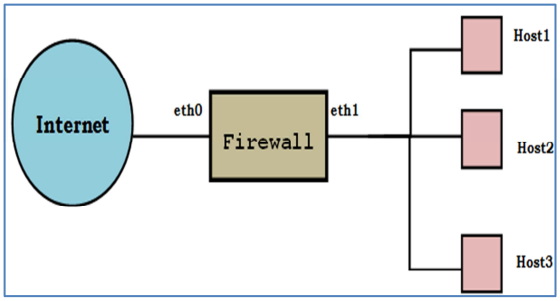

Fig. 1: Firewall Architecture

3. 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

A network administrator or a management tool issues commands on firewall to transform the running policy R into the target policy T. 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 [1]:

(app r), (del r), (del i), (ins i r) and (mov i j).

Policy editing languages can be classified into two representative classes [1]: Type I and Type II.

3.1.1 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 [1], therefore it is complete. Older firewalls and some recent firewalls, such as FWSM 2.x [6] and JUNOSe 7.x [7], only support Type I editing.

3.1.2 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 [1], 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. Examples of Type II editing firewalls include SunScreen 3.1 Lite [8] and Enterasys Matrix X [9].

3.2 Deployment Efficiency

A deployment is most-efficient if it utilizes the minimum number of editing commands in a given language, to correctly deploy a target policy 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. Usually a policy editing command takes constant time, and the variation in deployment time is negligible

Journal of Theoretical and Applied Information Technology
20th January 2014. Vol. 59 No.2

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

for different types of 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.

4. TYPE II DEPLOYMENT

The Type II deployment allows for random modification of a running policy. Therefore, for a given set of I and T, a safe Type II deployment usually utilizes less editing commands than an equivalent Type I deployment.

4.1 Problems With Previous Algorithm

In [1], 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 [1] 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 figure 2 and figure 3.

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. }.
```

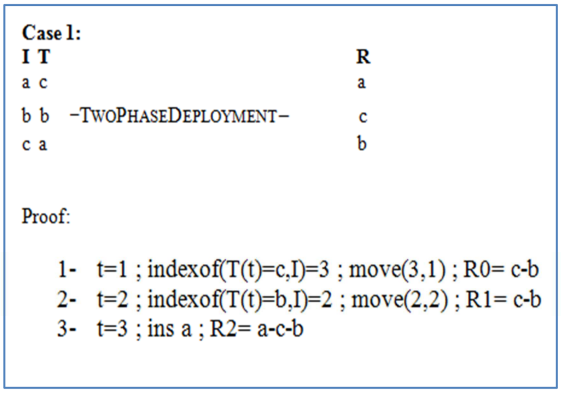

Fig. 2: Twophasedeployment Running For Case 1

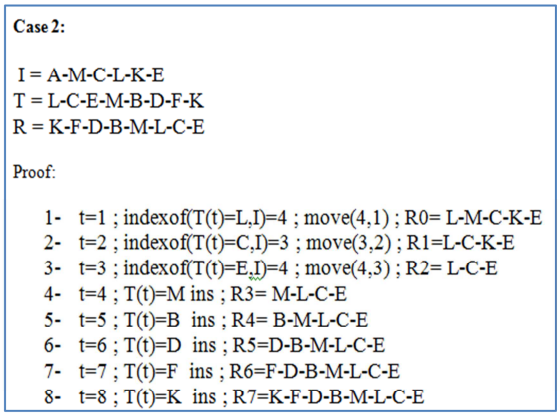

Fig. 3: Twophasedeployment Running For Case 2

We can clearly observe that the order of the rules is not respected in the both cases, 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 Algorithm 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
```

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

```
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.  }.
```

This algorithm includes the following changes:

1 - Remove the variable "inserts": Initially, she had served in the positioning of elements within the "I". This problem does not arise since the "I" is scalable so it is supports the new elements added. This is confirmed by the operation and functions "indexOf" and "del" working on "I" intermediaries.

2 - The introduction of the command ins(t T [t]) that is supported by the firewall type II and overcomes the problem of positioning the new rules. Indeed, the insertion made after such change is made in a wrong position as new elements are added in the lead.

The first two examples can be reused in the figures 4 and 5 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.

Case 1:

I = a-b-c
T = c-b-a
R = c-b-a

Proof:

1- t=1 ; indexof(T(t)=c,I)=3 ; move(3,1) ; R0= c-b
2- t=2 ; indexof(T(t)=b,I)=2 ; move(2,2) ; R1= c-b
3- t=3 ; ins(T(t),3) ; R2= c-b-a

Fig. 4: ENHANCED-Twophasedeployment Running For Case 1

Case 2:

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:

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

Fig. 5: ENHANCED-Twophasedeployment Running For Case 2

5. CONCLUSION

We have shown in this paper, that recent approaches [1] 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 algorithm called ENHANCED- TWOPHASEDEPLOYMENT. We will work on the second algorithm called SANITIZEIT to improve it.

REFRENCES:

[0] 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.

[1] 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.

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

[3] T. Yl¨onen. SSH: secure login connections over the internet. In SSYM'96: Proceedings of the 6th conference on

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

USENIX Security Symposium, Focusing on Applications of Cryptography, pages 4–4, Berkeley, CA, USA,
1996. USENIX Association.

[4] 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.

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

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

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

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

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