Firewall Policies Management: A Survey Paper

Hootan Rashtian
M.Sc. Student
Canada
University of British Columbia
rhootan@ece.ubc.ca

ABSTRACT

Firewalls are critical components of network security and have been widely deployed for protecting private networks. A firewall determines whether to accept or discard a packet that passes through it based on its policy. However, most real-life firewalls have been plagued with policy faults, which either allow malicious traffic or block legitimate traffic. In this paper we had an overview on some of the top recently published papers in this topic. We also suggested some points as the future works in this important field.

Keywords

Firewall policies, Policy faults, Fault localization.

1. INTRODUCTION

Nowadays, with the global Internet connection, one of the most important aspects of networks is their security both in research and industrial communities. Due to the increasing threat of network attacks, firewalls have become more important elements than ever not only in enterprise and large scale networks but also in small-size networks with any kind of applications such as business, institutions, and home networks.

Firewalls have been the very first defense for secure networks against attacks and unauthorized traffic. Its task is ideally to filter out unwanted network traffic coming from or going to the secured network. The filtering decision is based on the firewall policy which is a set of ordered filtering rules defined according to predefined security policy requirements. This sequence of rules follow the first-match semantics where the decision for a packet is the decision of the first rule that the packet matches A firewall is often placed at the entrance between a private network and the outside Internet so that it can check all incoming and outgoing packets and decide whether to accept or discard a packet based on its policy. However, most real-life firewall policies are poorly configured and contain faults (i.e., misconfigurations) [1]. A policy fault either creates security holes that allow malicious traffic to sneak into a private network or blocks legitimate traffic and disrupts normal business processes. In other words, a faulty firewall policy evaluates some packets to unexpected decisions. These kind of packets are called misclassified packets of a faulty firewall policy. Therefore, it is important to find ways that can assist firewall administrators to automatically correct firewall faults.

There are many different hot issues about the firewalls since their role is really important in terms of security of the networks. In this section, we are going to clarify motivation and also the scope of this survey paper by introducing the subjects clearly.

One of the most important issues is automated correction of firewall policies faults. It has been always a critical point because these policies which are modeled as a sequence of rules can have huge effect on creating security holes. On the other hand, it can also block the allowed traffic which can be also assumed as a weak feature of the firewalls. Both of the mentioned problems are originated from a cause which is technically called firewall misconfigurations. For solving this problem, other experts in field of network security have suggested different solutions such as semi-automatic and automatic correction of firewall [2]. There are different challenges in solving this problem such as categorizing, locating and correcting the faults, and we covered their ideas and contributions by analyzing the works and defining their weaknesses and strengths.

Fault localization is also one of the issues that are covered in this paper [3]. It is a very important branch of software testing which worth to be analyzed as a part of our study. Some new approaches have been introduced that have the ability to reduce needed effort and time for fault localizing of firewall policies. They can be effective for optimization of different kinds of tools developed for management of firewall policies since all of them frankly deal with detection of fault locations.

Another alternative for management of centralized and distributed firewalls policies is to develop new tools which assist administrators to purify the firewall policy from rule anomalies [4]. Another expected advantage of such tools is to help the administrator to manage legacy firewall policies. This should be actually done without any need for prior analysis of rules. This idea can be more effective when it is combined with usable security concepts. It causes to develop tools that are so user-friendly that even not

professional end-users can use them and perform some levels of firewall management. Because of the importance of usable security, we cover some of the latest works in this really interesting field.

One important related aspect to optimization of firewall policies is process of testing. To help ensure the correctness of firewall policies, researchers have developed various firewall analyses and testing tools [6]. The main function of these firewall analysis tools is to detect anomalies in firewall policies based on common patterns of firewall configuration mistakes.

Although such firewall analysis tools are useful, the weakness of such tools is that the "anomalies" may not be mistakes and also the number of "anomalies" could be too large to be practically useful [7],[8]. Several firewall policy testing techniques have been proposed [9]. However, these techniques for firewall policies testing are not based on well-defined testing techniques in software engineering [10].

For example, these techniques do not consider coverage criteria for firewall policy testing. We will cover this aspect. It is a very critical part of the work since the reliable evaluation of all the techniques designed to improve the security of firewalls will not happen unless some stable testing tools for this special purpose have been developed.

From another point of view, firewall policies faults can be prevented by using new design methods [11]. In new design methods, common weaknesses of firewalls should be considered [1]. Usability aspect of firewall tools also can give an insight in terms of creating new design methods [5]. Thus, this topic itself has enough importance to be covered in our work since it reduces the overall needed effort of securing networks.

All the above topics will be discussed in this survey project both in terms of qualitative and quantitative analysis. The results of the work hopefully can give the readers an insight about the literature and also introduce the open questions and future works. Table 1 shows the titles of selected topics.

The following parts of the paper will be as follows:
Section 2 is mainly about the works that have been done so far about automated correction of firewall policies faults and analysis of different recent methods available in this field and their features and weaknesses. In section 3, the idea of fault localization will be covered based on top recent works. In section 4 is basically about the tools for administrators assistant and advantages and disadvantages of some of the recently developed tools for this aim. Testing of firewall policies will be more discussed in section 5. As the last field that is discussed in this paper, new design methods of firewalls will be covered in section 6. Finally, we will have a conclusion in section 7 where some general conclusion and future works are discussed.

2. Firewall Policy Model

Firewall policy has a common model [3] which consists of a set of rules. Each of the rules has a format as follows:

⟨predicate⟩→⟨decision⟩(1)

A ⟨predicate⟩ defines a collection of packets over a definite number of *fields. Fields* are showed in the format of F1, …, Fn. On the other hand, ⟨decision⟩ of a rule is related to the evaluation of predicate and it can be true or false. If predicate is evaluated to true, then the decision will be appeared.

A packet is basically a tuple (fv1, ..., fvn). As it was mentioned earlier, it is defined over a finite number of *fields* F1... Fn. fvi is a variable whose values of fvi variable are within a D(Fi)domain. It is common to mention values in *fields* to their integer values to make representation format as simple as possible. The predicate itself can be represented:

(F1∈S1)∧...∧(Fn∈Sn)(2)

In this representation, Si denotes a part of domain D(Fi). Each of (Fi ∈ Si) is called a ⟨clause⟩, which should be evaluated to correct or incorrect. It is very important that a firewall policy uses a standard semantic for its functionality which is called the first-match semantic. In this semantic, there is an iteration which continues until the time it reaches the end of the rules. This iteration starts from the first rule by looking for its predicate to see whether it has been evaluated to true or not. If this condition is satisfied then the decision that corresponds to this rule is derived and returned; otherwise it goes to the next rule in the set of policies.
There are some other expressions that are technically used in this field. For example, conflict or anomaly, overlap, shadowing, generalization, correlation and policy conflicts. Conflict means as follows [4]. When we talk about policy conflict, we are talking about an entity that is associated with a collection of rules. These rules have the ability to derive a packet space which is common among them. The point here is that rules in this collection match all the packets and at least two of the rules have different decisions.
Matching of a packet with different rules is called overlap. Shadowing happens when a rule cannot effect on the decision for passing or failing the packet because of the preceding rules that match the packet. Generalization means when a subset of matched packets for this rule is also matched by preceding rules but with different decisions. Correlation happens when there is an intersection between a rule and some other rules but the matched packets by this intersection are not assigned the same decision.

Redundancy occurs when there are more than one rules in the policy with same effect.

3. Automated Correction of Policy Faults

Although automatic correction of firewall policy sounds very useful, it has its own difficulties to be correctly applied on the firewall. These difficulties can be categorized in separate groups as follows [2]:

- Counting the number of faults and defining types of faults
- Locating the origin of the fault among torrents of rules in the firewall
- Correcting the faults without making any side-effects which affects other rules and their functionalities

To resolve the problems of these difficulties, many researchers have introduced different techniques and approaches.

In [2], the authors tries to provide a solution that correct the faulty firewall policies in an optimized way which means in a way that needs minimum modifications. For this aim, they provide a model which covers all common policy faults. The model contains five fault types: wrong order, missing rules, wrong decisions, wrong predicates and wrong extra rules. The interesting point of their work is that they propose a correction technique for each of the categories in the model. The techniques are basically proposed based on the test of firewall policy. Test cases are made by generating packets to examine firewalls. There are different ways that have been suggested by researchers for packet generation. They use the test packet generation approach of [2] which consists of three different techniques. The provided approach in [6] for this aim tries to achieve the highest possible structural coverage. First technique is random packet generation. For using this technique, domains of each of policies should be defined and packets will be generated randomly based on domain scopes without using the policies. The benefit of this technique is the high speed of packet generation. On the other hand, the lack of achieving high coverage is notable because of its random inherent. The second technique generates packets based on local constraint solving. Unlike the previous one, it analyzes each rule separately and checks conditions of that rule without considering probable effects of other rules. Although it has been considered to be far from randomness, but there are still some problems with this technique. First, because of overlaps of predicates, some entities may not be covered. Second, it is not clear that which entities will be ignored and which will not. Therefore, it is not possible to decide for generating other packets to cover those entities. The third technique is packet generation based on global constraint solving. In this technique, overlaps of predicates are considered by analyzing the policy and also the constraints of the policy. It has better efficiency in terms of covering the target entities, but the limitation is the large amount of needed time for analysis. In [3], authors provided another automatic way for packet generation by introducing rule representation and traffic space segmentation and a segmentation algorithm for firewall policies which make it possible to generate effective test packets for the aim of optimization but the point here is that the proposed approach was used for testing the implementation of firewall and it was not necessarily for automatic correction of firewall policies. As another way of test generation, [10] used a specification centered approach. In this case, definitions of test case specifications are required and these could be consequences a packet might be dropped or other potential scenarios. After that test cases are generated to cover transitions of states in firewalls and connected networks. In general their approach is capable of supporting different scenarios that are designed using the execution history of the system. It is not clear in their work that how they can handle cases that the number of test cases with special specification get very large. This weakness is somehow common as we can see in the following part.

The next challenge in test generations is the huge number of generated packets. Since the task of classifying the test packets are done manually, too huge number of generated tests will make it too time consuming and somehow impossible to check all of them. They also provided two algorithms. For this reason, [6] used a technique which was firstly proposed by [12]. In this technique, the main goal is to decrease the number of generated packets without affecting the quality of them. In other words, this reduction should not lead to a notable loss in terms of the structural coverage. What is done in this test reduction phase is basically to remove those packets that do not increase the coverage based on previously defined metrics. This is done by evaluating packets one by one. The point here is that, since this algorithm works greedily, the result will not be necessarily optimized. By optimized result we mean the lowest number of packets which meets the same level of coverage as the given set.

As the next step, it is time to correct the faulty firewall policy using least possible number of modification. Basically it is a difficult task as we discussed earlier about the nature of firewall policies. In [2], authors suggests to algorithms for this purpose. First one is a greedy one which works in an iterative way to use the best correction technique among the previously mentioned five techniques for maximizing the number of passed tests for each fault. As it is shown in Figure 1, all five techniques are used at first. Then it is calculated that how many of test packets are passed or failed. The one with a higher efficiency is chosen. This process is continued until the time that there is no more failed packet. It worth to mention that it does not

provide the optimized solution always. Especially when it adds new rules to the policy rules, it approaches to the

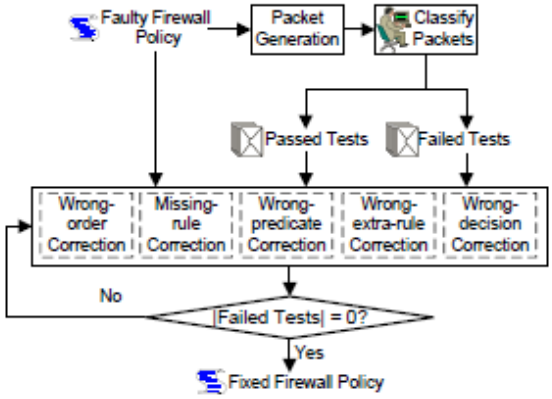

Figure 1: Greedy Algorithm, Adopted from [2]

worst case by maximizing the length of needed modifications.

For solving this weakness, they provide an improved version of the algorithm. In the improved algorithm they split the techniques into two levels. The reason is that in this way it is possible to make sure that increasing the number of modification is necessary. As it is seen in Figure 2, in this algorithm, classified test packets are passed to wrong order correction, wrong decision correction and wrong extra-rule correction techniques to see if they can decrease the failed packets or not. If the answer is no then they are passed to the other two techniques which are missing rule correction and wrong predicate correction. Again it continues until the time that there is no failed test. Another notable feature in their work is that administrators are allowed to manually change the flow of the algorithm. For example they can first try their preferred techniques. It is very good since make the approach more usable to be embedded in firewall administrative tools.

They also applied their approach on real life firewalls and showed that for three categories of fault model, their approach was highly successful in terms of detecting. There is another interesting research in this area [12]. In this work, the rules are analyzed and then structural coverage metrics are used to find faulty rules in the policy. Although the results show high efficiency, it seems that there some problems with their work. First is that just wrong decisions and predicates are covered in it. As the second one, it has been assumed that firewall policy has only one fault which is far from reality. The last weakness is the most important and highly related to this section. It does not provide the user any solution for detected problems in the firewall policy and this cannot be claimed as a full automatic approach.

As we discussed in this section about top approaches in automation of firewall policy correction, still some parts of the works such as classifying the test packets are done manually which means that the whole process is not %100 automatic.

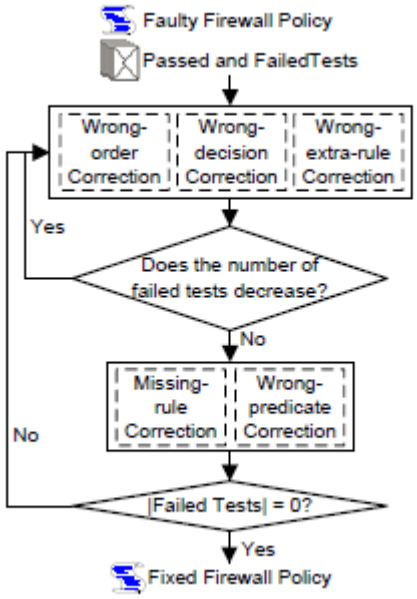

Figure 2: Improved algorithm, Adopted from [2]

4. Fault Localization

A very important part in management of firewall policies is to fix the faults. For this goal, first the root of the fault should be defined in the debugging phase of testing. The act of finding the root of the fault is technically called fault localization[13]. If testers and debuggers want to perform it manually, it will take a very long time because of the inherent complexity of fault localization and the huge amount of rules. Thus, efficiency of localizing the faults should be considered as a factor of quality. In [3], Hwang proposes an approach for this task in order to decrease the cost of debugging and fault localization. The approach is based on a fault model which is suggested by the authors consisting Rule Decision Change (RDC) and Rule Field interval Change (RFC) type faults. Each of them shows a kind of incorrectness. RDC shows that the decision is correct or incorrect about a special rule. RFC also shows a false definition of an interval in a rule. It worth to mention that in this approach, it has been assumed that policy has just one fault which is not a realistic assumption. The overall flow of the approach starts by inspection of those test cases that have not made the correct decision. If they do not include those types specified in the fault model, some other rules will be named as the potential tricky rules. As the main idea, the number of rules to be inspected is decreased by analyzing the characteristics come from faults. Still this is not the end of the game. After the previous step, the remaining rules are ranked based on previously defined metrics such as structural coverage. A more detailed review

on [3] shows some more interesting aspects of the proposed idea. The process of reducing the number of rules to be inspected and ranking them has been categorized in three separate techniques. These are basically come from some heuristics or better to say some rules of thumb. The first technique is called Covered-Rule-Fault Localization [3]. The idea is useful when there are so many rules that are covered by failed test cases. In this case, it has been suggested to policy testers to give the highest priority to the earliest rule that has been covered. As a very simple example, consider set of rules in a firewall policy. Imagine that m>n and n*th* rule is covered by n*th* failed test case and m*th* rule is covered by m*th* failed test case. On the other hand modification of n*th* rule does not have effect on m*th* rule to be covered by m*th* failed test case or not. Thus, the n*th* rule is placed in a higher priority than the m*th* rule. The second technique introduce a way for the case that the earliest-placed rule that is addressed by failed test cases has no fault. In this special situation, the earliest-placed rule is in a higher priority and therefore regardless of the modifications on the faulty rule which has currently a lower priority, it has still false result as the decision. Thus, the faulty rule should be assigned a higher priority than the earliest-placed rule so that it will be observed during the process of evaluation. After selecting rules by the previous technique, the third technique will select from them those with more probability to be faulty one. In this technique, the idea is that the decisions of faulty rules and other rules should be different from each other. Imagine that dec1 is the decision for the clause faulty rule with higher priority and the dec2 is the decision for the rule with lower priority. After modifying of the clause of faulty rule, dec1 should be different from dec2 since dec2 is unexpected for that failed packet.

In related to the previous technique, the authors introduced a way for giving priority to rules based on analyses on the clause coverage and the probability to be faulty regarding those analyses. The idea here is based on a fact which is inferred from the analysis on the clause coverage for the rules that are faulty according to failed packets. This fact says that at least equal or smaller number of clauses is evaluated to false in faulty rules. Based on this observation, they have suggested a list of rules from the lowest value of clause coverage to the highest one for the task of inspection. We use their example to show the idea more clear. As it is shown in figure 3, 9 out of 12 test cases are passed and 3 out of 12 are failed. Since fourth and fifth rules are covered by failed tests, the first step is to look for fault(s) in the fourth rule because it is the earliest-placed rule covered by faulty tests. By knowing that there is no fault in this rule, the next step is to apply rule reduction. Because first three rules have higher priority than R4, we consider them as candidates. Reason is that any problem with these rules may cause an incorrect result by R4. In this case R1 and R3 are chosen since they have different decisions than R4. In last step, we should see which one has a higher rank to be chosen for inspection.

Another work that is close to the aforementioned approach is Marmorstein et al.'s work [3]. In this paper, the authors provided two techniques not only to detect the firewall policy errors, but also repair them. In other words, the proposed techniques facilitate the act of tracing errors and finding the roots of errors. In order to use these techniques, the user specifies the desired behavior of the firewall using logical assertions. The syntax for assertions is derived from the query language explained in [14]. One useful advantage of assertion analysis is that it allows generation of relevant counterexamples. These counterexamples provide a context for the error which can often help the administrator discover why a failure has occurred.

As the first technique, they provided a way which generates some packets as examples to show weaknesses and requirements of the firewall policies. Their second technique makes a history log of rules which helps the administrators and policy testers to find the root problem. For this aim, more accurate information are collected about the particular rules that cause a problem or an error by creating a log consists of history of the rules. They named it history map. This map has a duty to match each failed packet to the set of rules that potentially pass or fail it. By using the history map, it is possible to associate packets in an assertion's fail collection with a fewer amount of rules for filtering. Most probably the number of rules is smaller in comparison with the total number of rules in the whole policy. It means that workload for administrators and policy testers will be decreased in terms of inspection and they will not need to inspect the whole set of rules exhaustively anymore. Instead of that, they will just focus on the special areas that are suggested by the history map and they will omit the unrelated rules.

In comparison with the previous technique, this one has some disadvantages. For instance, it just considers rules that are related to failed packets while the previous one considers broader range of rules in RFC fault type. The output of this one is more like a general list of rules that are potentially faulty while the previous one decreases the number of probably rules and gives them priority values. In [2], which is the next work of the authors of [12], weak assumption of having only one fault in firewall policy is not used anymore.

One point that seems to need more work is the improved algorithm in [2] does not have a notably better efficiency than the greedy one (%5 better). This exactly points to the fact that still more works needs to be done about the optimization of correction.

5. Firewall Tools

Another point of view in managing firewall policies is developing tools which help administrators in dealing with this complex and time consuming task. It is aggravated by

change of the environment which can be change of the entities or topologies or standards in the network. This calls the urgent need for some special tools and systems to play a role in managing the firewall policies. In this section we

be some rules that have fewer conflicts. In addition to these kinds of problems which are related to the nature of the firewall policy and its concept, there is another difficulty which arise from the nature of the way that firewall policies are maintained. It is very common for firewall policies to be

	# Rule Cov		# Clause Cov by failed tets			Selected	Rank
	#Failed	#Passed	#True	#False	Susp		
R_1: $F_1 \in [0,10] \wedge F_2 \in [3, 5] \wedge F_3 \in [3, 3] \rightarrow$ *accept*	0	2	6	3	0.66	●	2
R_2: $F_1 \in [5, 7] \wedge F_2 \in [0, 10] \wedge F_3 \in [3, 5] \rightarrow$ *discard*	0	2					
R_3: $F_1 \in [5, 7] \wedge F_2 \in [0, 10] \wedge F_3 \in [6, 7] \rightarrow$ *accept*	0	2	4	5	0.44	●	3
R_4: $F_1 \in [2,10] \wedge F_2 \in [0, 10] \wedge F_3 \in [5,10] \rightarrow$ *discard*	2	1				●	1
R_5: $F_1 \in [0,10] \wedge F_2 \in [0, 10] \wedge F_3 \in [0,10] \rightarrow$ *discard*	1	2					

Figure 3: Example of rule reduction and rule ranking,

Adopted from [3]

had a look on some of the recent works in this area that have received the attention of the community. Also an overview on usability aspect of firewall systems was studied as an important feature should be considered for design of any kind of systems.

Among the recently developed tools, policy anomaly detectors have been very popular [15, 16, 17, 18, 19]. For example, FIREMAN [19] and FPA [11, 12] have been designed to address this problem. But they have still some limitations that sometimes make trouble for policy anomaly detection. For instance, just being capable of doing pairwise anomalies in the long list of firewall rules is not enough. This limitation is seen in FPA. For FIREMAN, situation is better since it has been designed in a way that analyzes the rule that is under test to all preceding rules. Although it works better than FPA but still it is not the best. The reason is that it does not consider the subsequent rules, which mean that this approach does not cover all the rules so that it is not complete. When an anomaly is detected by FIREMAN, it is not clear that which of the rules is involved in this anomaly. All the provided information is about one of the rules and previous rules before that. On the other hand, for resolving anomalies in policy, it is necessary to deal with policy conflicts. The fact here is that it is very difficult to consider all aspects for removing or modifying the conflicting rules to make the policy free of conflicts. Sometimes the degree of difficulty is changed from very difficult to impossible. If we want to mention more details about difficulties, we can name the huge number of conflicts as the first. This is mainly because of the very large number of rules in firewall policies and also the fact that they are usually highly related to each other. This fact brings the second item of difficulty. In some cases some rules have conflict with many other rules, while there may

maintained by different persons as administrators. It itself is the root of many problems since it is expected for a firewall policy to have many legacy rules that are created and used by different persons. It is basically a bad feature since it affects both sides of the problem. In other words, it causes new conflicts because other administrators may not know exact assumptions of their colleagues. On the other hand, it needs some knowledge about the assumptions and intentions of different administrators in order to omit the conflicts correctly otherwise it might have bad effects on semantics and as the result it will not resolve the conflicts correctly. Sometimes also administrators use techniques to decrease the number of rules such as introducing some overlaps in the rules. This also increases the complexity of analyzing firewall policies.

In [20], authors introduced a framework for anomaly management of firewall policies. It works based on a segmentation technique which is rule-based. It has a higher efficiency in terms of anomaly detection in comparison with aforementioned tools and also facilitates effective resolutions for anomalies. This framework which is called FAME is also capable of generating the outputs using a technique for information visualization. This feature is a big difference regarding the fact that previous tools could not support this feature and their outputs were just limited to a general list of potential anomalies. But with visualization, it is much more convenient for the user to analyze and explain by using their visual cognition. A grid-based visualization is provided which facilitate management for administrators since information are presented in grid format. It is one of the works in terms of usability since the interface has been designed regarding the common requirements for

performing this task. Figure 4 shows snapshots of the interface of FAME.

In [4], another tool has been developed which gives a very good insight to administrators about the effects of a specific change in a firewall policy. In this tool, the input has been defined to be the configuration of the firewall and the proposed changed. The output will be the impact of that

(a) Conflict Viewer for all conflicts

(b) Conflict Viewer for conflicts in one group

(c) Information of one conflict

Figure 4: Snapshots of usable interface of FAME,
Adopted from [4]

proposed change. It seems to be useful since it gives insight from both sides of the problem. For example, it can show that by applying a change some packets that were previously allowed to flow are not assumed to be legitimate anymore. On the other hand, it can show that also previously illegitimate packets will be allowed to flow in the future if the new change is taken place. The foundation of this tool consists of theorems which define whether the decision for each packet should be changed or not. A categorization for potential changes has been used. This categorization includes four common changes usually happen in firewall policies: rule deletion, rule insertion, rule modification and rule swap. Each of aforementioned theorems corresponds to each type of change. Then, based

on the theorems, some algorithms have been introduced for analysis of impact of change. It also provides methods for the cases that impact of the proposed change is not useful and has undesirable side-effect. There are some other works similar to this but the difference is that they have talked about the change-impact in context of general programs and software engineering [21] which are completely different from firewall policy. The authors of [12] have another paper which is close to this work. In [2], their proposed algorithm could compare semantics of two firewalls and could provide the differences between them. The main difference of their newer work is in efficiency of the algorithms. One may say that in [2] their algorithm could analyze the case that administrators apply more than one change at a time while the newer one cannot handle this case. Actually it seems that it is not a notable and important feature since in reality administrators apply changes one by one at a time. As we discussed earlier in section 3 in more details, there are other works on firewall tool development. Most of those approaches and corresponding tools are capable of doing firewall testing and analysis based on the known attacks. They cannot usually find faults that do not let legitimate packets to flow.

Firewall tools are also important from the usability point of view. As we mentioned earlier in section 1, some parts of the process of firewall policy management are still performed manually. On the other hand, it is usually better to give the administrators the option to stop some automatic part of the work to do it manually if it is needed. From another point of view, having a well-organized interface for flow of the work will help users have a better understanding of work flow. All of these reasons make it vital for a firewall tool to have a usable interface. In [5], the author mentions that the problem of experience of working with such tools comes from different things. First, it is not a high-level task as common tasks like programming, system design and project management. Another one is the high speed of getting more complex. It is difficult for the user to understand and then catch up with problems. Although most of administrators use command line interface as the main interface, some visualizations are described in this paper as suggestions to be used. These visualizations can make configurations more simple. These suggested visualizations are based on design rules so that they are most probably usable. Some other works have been done for visualization. For instance in [22], the proposed tool make it possible for administrators to see a visualization for some special packets by just asking about specification of packets. Also in [23], the low-level task of configuration has been changed into a really high-level task. Users need to ask their questions like their daily conversations. It works quite like an expert system.

As the future work, it seems to be quite a good job if researchers focus more on qualitative study for usability of the tools that they have already developed. Having a look on available papers in this area shows there is not enough focus on the importance of the fact that how usable these tools are.

6. Wrap-up:

In this paper we tried to cover important issues that are related to the firewall policy management. We first mentioned the importance of the topics as the introduction. Then, a quick overview on the most common firewall policy was provided in section 2. After that we reviewed recent works on automatic correction and fault localization in two separate sections. It seems that more works is needed in terms of optimization of algorithms for fault localization since the improved versions of algorithms do not seem to be notably better than initial algorithms that they provided earlier. For automation, still classification of test cases is usually done by administrators. In firewall tools section, we tried to have a broader look on the works that have been done for developing tools. In this section, we also focused on usability aspect of the developed tools. Although different works are available for visualization, they are not evaluated in terms of usability. It might be a good suggestion to set up some qualitative studies for this aim.

7. REFERENCES

[1] WOOL, A. A quantitative study of firewall configuration errors. IEEE Computer 37, 6 (2004), pp. 62–67.

[2] Fei, C., Liu, A.X., Hwang, H., Xie, T. 2010. First Step Towards Automatic Correction of Firewall Policy Faults. In Proceedings of the 24th USENIX Large Installation System Administration Conference(LISA 2010), San Jose, CA, November 2010

[3] Hwang, J., Xie, T., Chen, F., and Liu, A. X. 2009. Fault localization for firewall policies.In Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS).

[4] Hu, H. Ahn, G. AND Kulkarni, K. FAME: A firewall anomaly management environment. SafeConfig '10 Proceedings of the 3rd ACM workshop on Assurable and usable security configuration (New York, NY, USA, 2010), ACM, p. 4.

[5] Wong, T. On the usability of firewall configuration. Submitted to Workshop on Usable IT Security Management, part of Symposium On Usable Privacy and Security (SOUPS) 2008, July 23-25, 2008, Pittsburgh, PA, USA.

[6] HWANG, J., XIE, T., CHEN, F., AND LIU, A. X. Systematic structural testing of firewall policies. In Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS) (2008), pp. 105–114.

[7] AL-SHAER, E., AND HAMED, H. Discovery of policy anomalies in distributed firewalls. In Proceedings of IEEE Conference on Computer Communications (INFOCOM) (2004), pp. 2605–2616.

[8] LIU, A. X. Change-impact analysis of firewall policies. In Proceedings of European Symposium Research Computer Security (ESORICS) (2007), pp. 155–170.

[9] LIU, A. X., GOUDA, M. G., MA, H. H., AND NGU, A. H. Non-intrusive testing of firewalls. In Proceedings of International Computer Engineering Conference (ICENCO) (2004), pp. 196–201.

[10] JURJEN, J., AND WIMMEL, G. Specification-based testing of firewalls. In Proceedings of International Conference Perspectives of System Informatics (PSI) (2001), pp. 308–316.

[11] LIU, A. X., AND GOUDA, M. G. Diverse firewall design. IEEE Transactions on Parallel and Distributed Systems (TPDS) 19, 8 (2008), pp. 1237–1251.

[12] E. Martin, T. Xie, and T. Yu. Defining and measuring policy coverage in testing access control policies. In *Proc. 8th*International Conference on Information and Communica*tions Security*, pages 139–158, 2006.

[13] J.A. Jones and M. J. Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, Pages 273-282, 2005.

[14] R. Marmorstein and P. Kearns. Assisted firewall policy repair using examples and history. In Proc. of the 21st Conference on Large Installation System Administration Conference,pages1–11,2007.

[15] Marmorstein, Robert and Phil Kearns, ‘‘A Tool for Automated iptables Firewall Analysis,’’ FREENIXTr a c k , 2005 USENIX Annual Technical Conference, pp. 71-82, April,2005.

[16] E. Al-Shaer and H. Hamed. Firewall Policy Advisor for anomaly discovery and rule editing. In Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on, pages 17–30, 2003.

[17] E. Al-Shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In IEEE INFOCOM, volume 4, pages 2605–2616, 2004.

[18] J. Alfaro, N. Boulahia-Cuppens, and F. Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. International Journal of Information Security, 7(2):103–122, 200.

[19] F. Baboescu and G. Varghese. Fast and scalable conflict detection for packet classifiers. Computer Networks, 42(6):717–735, 2003.

[20] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, P. Mohapatra, and C. Davis. Fireman: A toolkit for firewall modeling and analysis. In 2006 IEEE Symposium on Security and Privacy, page 15, 2006.

[21] Ren, X., Chesley, O.C., Ryder, B.G. Using a concept lattice of decomposition slices for program understanding and impact analysis. IEEE Transactions on Software Engineering 32(9), 718–732 (2006).

[22] T. Tran, E.Al-Shaer, and R. Boutaba. PolicyVis: Firewall Security Policy Visualization and Inspection. In poceedings of 21st Large Installation Sstem Administration Conference. Nov, 2007.

[23] A. Mayer, A. Wool and E. Ziskind. Fang: A Firewall Analysis Engine. In Proceedings of IEEE Security and Privacy, May 2000.

.