Formal Verification of Firewall Rules

February 17, 2010

1 Introduction

Firewall rules are executed in a sequential order. The action to be taken on an incoming packet is determined by the first rule that matches the packet. There may be cases where certain rules are never executed, since they are preceded by some other rules which match all the packets that are matched by them. This occurs when some specific rule is preceded by some general rule. Rule-i is said to be more specific than rule-j if all packets satisfied by rule-i is also satisfied by rule-j. A firewall rule can be said to be correct if the specific rules always precedes general rules. The main aim of the project is to find whether any general rule is preceding a specific one.

In order to verify the correctness of firewall based on ordering of its rules, we define a partial order relation for the rules. The partial order relation is defined based on the generalisation of the rules.

2 Representation of firewall rules

Let R denote the set of firewall rules.
Let port = { 0,1,2,...65535, $*_{port}$ },
IP $=\{address/prefix - length|$ address is a 32 bit string and $0 \leq prefix - length \leq 32\} \cup \{*_{ip}\}$,
protocol = { tcp, udp, icmp, $*_{protocol}$ },
action = {DENY,ACCEPT}.
The set of firewall rules, R $\subset$ *port* $\times$ IP $\times$ *port* $\times$ IP $\times$ *protocol* $\times$ *action*. A firewall rule can be represented by a tuple r $\in$ R.

3 Definition of partial order on firewall rules

The partial order relation that we are defining is "is specific than" $\sqsubseteq$. Inorder to define the partial order on R, we first define partial order on the component sets of R.

The "is specific than" relation on port is defined as follows. Let $p_1, p_2 \in$ port.

$$p_1 \sqsubseteq p_2 \Leftrightarrow p_1 = p_2 \text{ or } p_2 = *_{port}$$

Similarly we now define the relation on IP. Let $a1, a2 \in IP.a1, a2 \neq *_{ip}$. For x $\in$ IP, let x.address(n) return the first n bits of address represented by x. If a1.address(min(a1.prefix-length,a2.prefix-length)) = a2.address(min(a1.prefix-length,a2.prefix-length)) then

1. a1.prefix-length $\leq$ a2.prefix-length $\Rightarrow$ a2 $\sqsubseteq$ a1
2. a2.prefix-length $\leq$ a1.prefix-length $\Rightarrow$ a1 $\sqsubseteq$ a2

Also,

$$\forall x \in IP, x \neq *_{ip} \Rightarrow x \sqsubseteq *_{ip}$$

The "is specific than" relation on protocol can be defined as following.Let $p_1, p_2 \in$ protocol.

$$p_1 \sqsubseteq p_2 \Leftrightarrow p_1 = p_2 \text{ or } p_2 = *_{protocol}$$

Two elements x_1 and x_2 is said to be related, that is

$$x_1 \bowtie x_2 \Leftrightarrow x_1 \sqsubseteq x_2 \vee x_2 \sqsubseteq x_1$$

else x_1 and x_2 is said to be not related.

$$x_1 \not\bowtie x_2 \Leftrightarrow \neg(x_1 \sqsubseteq x_2) \wedge \neg(x_2 \sqsubseteq x_1)$$

Now we can define the "is specific than" relation on a set of firewall rules R. Let $r_1, r_2 \in$ R and let $r_1 = (sp_1, sip_1, dp_1, dip_1, prot_1, a_1)$ and $r_2 = (sp_2, sip_2, dp_2, dip_2, prot_2, a_2)$.

$$\begin{gathered} r_1 \sqsubseteq r_2 \Leftrightarrow (sp_1 \sqsubseteq sp_2) \wedge \\ (sip_1 \sqsubseteq sip_2) \wedge \\ (dp_1 \sqsubseteq dp_2) \wedge \\ (dip_1 \sqsubseteq dip_2) \wedge \\ (prot_1 \sqsubseteq prot_2) \end{gathered}$$

We define partial order relation only on ordered set of firewall rules. Two rules r_1, r_2 are said to be in DISORDER if there exists some fields in r_1 which are $\sqsubseteq$ the corresponding fields in r_2 and all the other fields in r_1 $\sqsupseteq$ corresponding fields in r_2.

In case of disordered set of rules, we first convert them into a set with no disordered pair of rules. We introduce the function ORDER which takes the disordered tuples as input and generates an ordered set of tuples as output.

```
ORDER (R)
{
    if R has no disordered pairs then
       done
    else
       select pair of disordered rules from R
    Y = split(pairs)
    ORDER(R - pairs ∪ Y)
}
```

The following algorithm checks whether the given set of firewall rules with out any disorder pairs is in correct order. A set of firewall rules R is said to be in correct order if no general rule precedes a specific rule. That is if

$$\forall r_i, r_j \in R, i < j, \neg(r_j \sqsubseteq r_i)$$

Algorithm to check whether a set of firewall rules is correctly ordered

We propose an algorithm using graphs for verifying whether the above condition is satisfied. The nodes of the graphs are used to represent the firewall rules. Let $(r_1, r_2 ..., r_n)$ be the set of firewall rules on which partial ordering is defined. The rules are scanned in a a sequential manner.
Let G be the graph.
We define the following :
`addComponent(Component c)`
    Adds a new component c to G.
`addNode(Component c,Node n)`
    Adds node n to the component c.
`c.end`
    Denotes the node corresponding to the most general rule in component c.

ALGORITHM

1. Add a component c to G and add rule r_1 as node r_1 to c.
2. Scan the rules sequentially.
3. When a rule r_i is reached
   (a) for all nodes r_j where $r_j = c_i.end$ for any c_i in G, if($r_i \not\bowtie r_j$) then
      - `addComponent(`c_k`)`
      - `addNode(`c_k`,`r_i` )`

   (b) for all components c_i of G, find $r_j \bowtie r_i$ where $r_j = c_i.end$
   for any r_j, if $r_i \sqsubseteq r_j$ then
     There is an incorrect ordering
   else
     for all r_j,
      - add edge r_j –> r_i
      - cnew = Merge all c_i to one component.
      - cnew.end = r_i

Advantages of our system

- We define safe operations on the partial order relations like safe-insert and safe-delete.
- Firewall rules follow graph structure.

4 Related Works

In [1], [2] and [3], the different kinds of firewall policy anomalies are discussed. The different anomalies identified are:

1. Shadowing
2. Correlation
3. Generalization
4. Redundancy

The papers [1], [2] and [3] introduces a technique for the detection of these anomalies in the firewall rule set. They have implemented it using tree. The nodes represent the tuples and the edges represent the values of the tuple in the rule. When a rule is detected, a new branch is inserted in the tree and if there is a chance of the branch being overlapping, then they conclude that there is a conflict in the rules. Hence all the conflicting rule pairs are identified. But, in [5], it is mentioned that this technique is of exponential complexity.

In [4], two algorithms for resolving the anomalies identified in [1], [2] and [3] are proposed. The main aim of the first algorithm is to produce completely disjoint rules. Shadowing and correlation are resolved by splitting the rules and deleting the rules which are not required. The disadvantage is that each rule might be split to a large number of rules. They haven't mentioned properly how these would be done. In the second algorithm, shadowing and post redundancy are resolved by making the specific rule to precede the general rule. Exact match is resolved by deleting one of the rules. They define a relation R_{IM} on the rule set and prove that R_{IM} is a partial order relation. They have defined an ordering function where the specific rule precedes the general rule. The ordering functions says that if $R_x R_{IM} R_y$ then index of $R_x <$ index ofR_y. Their algorithm takes a set of disordered pair of rules as input and reorder the rules to resolve the conflict. They also show that the complexity of this algorithm is O(n^2).

Our paper proposes a mechanism to detect shadowing and post redundancy in firewall rule set. The technique can be used only on ordered set of firewall rules. By ordered set of firewall rules, we mean the set without any correlation anomaly. Hence, our aim is :

- To resolve correlation
- To detect and resolve shadowing and post redundancy

Correlation anomaly can be resolved by splitting the rules to become disjoint. For each firewall rule, its correlation with all preceding firewall rules must be detected and corresponding splitting should be done. But the disadvantage is that the number of resultant rules will be very large.

The main motivation behind resolving correlation and then detect the other anomalies is that an iterative solution to solve both of them is difficult. It would always be good if both can be performed together. That is for each rule, if correlation anomaly can be resolved and the anomalies resolved then itself. That would reduce the complexity to a great extent.

References

[1] H Al-shaer, E Hamed. Modeling and management of firewall policies. *In IEEE eTransactions on Network and Service Management.*

[2] Hazem Boutaba Raouf Hasan Masum Al-shaer, Ehab Hamed. Conflict classification and analysis of distributed firewall policies.

[3] Hazem H Al-shaer, Ehab S Hamed. Firewall policy advisor for anomaly discovery and rule editing. *Information Systems.*

[4] Alex X Liu. Formal verification of firewall policies. 2008.

[5] S Pozo, R Ceballos, and R. M. Gasca. Fast algorithms for consistency-based diagnosis of firewall rule sets.