Test Case Generation for Firewall Testing

Tugkan Tuglular
Dept. of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey
tugkantuglular@iyte.edu.tr

Firewall tests have to be performed to verify that the firewall works as specified. In this work, a test case generation approach is developed, which defines test cases based on the firewall rule sequence and uses real traffic database to prepare test packets. Test packets can be used or injected to check if the firewall implementation is erroneous, i.e. the rules do not correspond to the actions of the firewall. Although injection based firewall testing is accepted as an inefficient way of testing firewall implementations in the literature [1], there has been no alternative method developed yet. Most of the academic work focuses on testing of firewall rules where firewall implementation is assumed error-free. Even if firewall implementation is error-free, a firewall can be hacked and programmed to behave differently from the intended security policy. In that case, real time injection based testing is one of the ways to reveal the security breach.

There are three general approaches to firewall testing; penetration testing, testing of the firewall implementation, and testing of the firewall rules [2]. Penetration testing is performed to check the firewall for potential breaches of security that can be exploited. The firewall implementation testing approach evaluates the correspondence of firewall rules with respect to the actions the firewall performs (e.g. if a rule indicates to block a packet but the firewall forwards the packet, that means a firewall implementation error exists) [2]. Testing of the firewall rules verifies whether the security policy is correctly enforced by a sequence of firewall rules or not.

The firewall implementation testing is achieved through defining test cases, deriving test packets from these test cases and sending or injecting the packets to the firewall to analyze its behavior [2]. *Heidi* states that there are two strategies to inject packets; injecting bogus packets pretending to be originated from all the hosts outside the private network and injecting bogus packets pretending to be originated from some of the hosts outside the private network, where the first strategy is extremely inefficient, and the second strategy cannot cover all possible host IP addresses, which make the testing incomplete [1]. This is very similar to white-box software testing, where software cannot be tested for all possible values and execution paths [3]. However, a limited number of important values and paths can be selected and exercised. This is the main idea behind the test case generation approach developed in this work.

Pressman states that test cases that (1) guarantee that all independent paths within a module have been exercised at least once, (2) exercise all logical decisions on their true and false sides, (3) execute all loops at their boundaries and within their operational bounds, and (4) exercise internal data structures to ensure their validity, can be defined using white-box testing methods [3]. Same principles are applied in the test case generation approach developed in this work. The common format of packet filtering rules, represented as (<order>,<protocol>,<src_ip>,<src_port>,<dst_ip>,<dst_port>,<action>) [4], is used throughout the application.

First, sequence of firewall rules is converted to a firewall policy tree developed by *Al-Shaer and Hamed*, where each node in the tree represents a field of the filtering rule, and each branch at this node represents a possible value of the associated field. The root node of a policy tree represents the protocol field, and the leaf nodes represent the action field, intermediate nodes represent other 5-tuple filter fields in order. Every tree path starting at the root and ending at a leaf represents a rule in the policy and vice versa. Rules that have the same field value at a specific node, will share the same branch representing that value [4]. This tree enables us (*i*) to traverse all independent paths, (*ii*) to exercise ACCEPT and DENY decisions on true and false sides of each leaf, (*iii*) to execute each path for the regions immediately adjacent to its boundaries, and (*iv*) to ensure that each rule is represented by a valid data structure. Item (*ii*) is explored by *Jurjens and Wimmel* as fulfilling and violating test cases in specification-based testing of firewalls [5]. Item (*iii*) is used by *Zaugg* as boundary test [2]. Items (*i*) and (*iv*) are trivial because of the tree structure.

Second, fulfilling and violating test values are generated for each node of each path. Boundary test values are included within the test values for addresses and ports as well. Test cases for each rule are composed of the combination of test values of each node. They are stored in a database table with a unique index, the corresponding rule number and the expected decision which is either ACCEPT and DENY. Then for each test case, a real network packet is created using real network traffic database. Chosen as a test case, a network packet is searched with the same protocol and if possible with the same port numbers. The best-fit network packet is then rewritten with the address test values specified for that test case and this rewritten packet is stored in another database table with the test case index.

In the next version, firewall policy tree will be replaced by anomaly free firewall policy tree. With this replacement, the change in the test cases will be observed and investigated. In the future, test case generation approach will be extended by including the knowledge of protected network, by integration of penetration testing goals (e.g. checking for vulnerabilities, spoofing, etc.), and by considering stateful connections.

References

[1] H. Heidi. Specification Based Firewall Testing, Master of Arts Thesis, Texas State University-San Marcos, May 2004.
[2] G. Zaugg. Firewall Testing, Diploma Thesis, ETH Zürich, 2005.
[3] R. S. Pressman. Software Engineering: A Practitioner's Approach, European 3Rev.ed., 1994, UK.
[4] E. Al-Shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In IEEE INFOCOM'04, March 2004.
[5] J. Jurjens and G. Wimmel. Specification-based testing of firewalls. In A. Ershov, editor, Proc. of the 4th International Conference Perspectives of System Informatics (PSI'01), LNCS. Springer.