Model-Based Mutation Testing of Firewalls

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

Fevzi Belli
Department of Computer Science, Electrical Engineering and Mathematics,
University of Paderborn, Germany
belli@upb.de

Abstract

We propose a combined approach for test case generation to uncover errors both in firewall software and in its configuration. A case study validates the approach.

1. Introduction

As being most important security defense of a network, firewalls have to be tested to validate that they work as specified. Most of the work known from security literature focuses on testing of firewall rules where firewall implementation is assumed error-free. However, a firewall can be hacked and programmed to behave differently from its specification. In that case, the hacked firewall is a *mutant* of its original.

The firewall specification is mainly composed of intended *security policy* and allowed *network protocols*, which are the main focus of a hacker, or an attacker. The intended security policy consists of firewall rules which configure the firewall behavior and the allowed network protocols constitute the important part of firewall's internal working which can be described as packet capture, decision making on the packet under consideration, and packet release. Decision making operation is carried out with respect to firewall policy and network protocols. The security policy is external to the firewall like a configuration file, whereas packet checking with respect to network protocols is implemented in the firewall software.

Since the firewall policy is considered as a specification, equivalence class testing is suggested. The network protocol checking can be represented by a model. Therefore, we use here model-based mutation testing. This is the novelty of the approach.

In the next section, our approach to firewall testing is explained. Section 3 presents a case study; Section 4 concludes with a brief summary and future work.

2. Approach

In the equivalence class testing part of the approach, sequence of firewall rules is converted to a firewall policy tree as described in [1], which is then used as a test tree. Each node in the policy tree represents a field of a rule. A firewall rule is abstracted as "IF (<protocol>, <src_ip>, <src_port>, <dst_ip>, <dst_port>) THEN <action>", where protocol is a network protocol, such as TCP or UDP, and action is either ALLOW or DENY. The root node of a policy tree represents the protocol field, and the leaf nodes represent the action field, intermediate nodes represent other fields in order. Every tree path starting at the root and ending at a leaf represents a rule in the policy and vice versa. The equivalence class partitioning divides the input domain of the software under test (SUT) into a finite number of partitions or equivalence classes. The equivalence classes are identified by taking each input condition – in our approach each field of the rule or each node of the policy tree – and partitioning it into two classes [2]. When generating test cases, values that a hacker might choose are considered in addition to the boundary values in equivalence classes.

Assuming that the given finite state automata (FSA) correctly specifies the expected, desirable behavior of SUT, manipulation of either the state transitions or the states can be used to generate mutants of the system, i.e., to specify erroneous, *undesirable* situations [3]. Using this model-based approach, a mutant can be generated and it will behave in a way that the system is not supposed to behave. Basically, we can generate mutants of an FSA by inserting an extra state transition in any direction, omitting an existing arc, inserting an extra state, or omitting an existing state.

Although firewalls are implemented as software, their method of input and output is network I/O. Therefore, network packets should be produced,

injected, and collected in order to test a firewall. Test packets will be derived from generated test cases and those packets will be sent or injected to the firewall to analyze its behavior. To be able to analyze and evaluate behavior firewall under test (FUT) with respect to test cases, we developed a special architecture as illustrated in Figure 1.

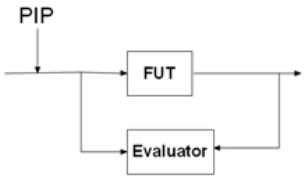

Figure 1. Firewall evaluator architecture

The test packets will be released from packet injection point (PIP). All the traffic entering and leaving the firewall will be recorded and collected data will be analyzed to obtain test outputs, which will be compared with expected outputs to determine test result. We expect to see allowed packets at the packet leaving point but not the denied packets. This architecture can also be used to monitor a firewall constantly to check whether it operates in accordance with its specification and implementation.

3. Case Study

Some of the test cases generated using specification based equivalence class testing approach for the firewall rule <tcp, 193.140.248.*, any, *.*.*.*, 25, accept> are presented in Table 1. Test cases numbered as 2, 3, and 4 have the given expected output unless there are rules concerning the values in the test input.

Table 1. A set of generated test cases

No	Test Input	Expected Test Output
1	tcp,193.140.248.*,any,*.*.*.*,25	allow
2	udp,193.140.248.*,any,*.*.*.*,25	deny
3	tcp,193.140.248.1,any,*.*.*.*,25	allow
4	tcp,193.140.248.*,80,*.*.*.*,25	allow

If the firewall under test has a statefull packet filter, as almost all firewalls have, that means that related automaton for each statefull protocol is implemented in the firewall software. If TCP protocol is supported by the firewall, it should contain necessary code that executes the FSA for the TCP protocol, of which graphical representation is given in Figure 2.

We can create now a mutant for TCP automaton by omitting state transition(s). Since TCP is a reliable protocol, it has a high level redundancy. Therefore, a mutant created by omitting only a single transition will not be able to behave altogether differently. However, if a hacker omits all the transitions going out from ESTABLISHED state, the connection will be virtually open forever. That means blocking a port of a host, which may be considered as a denial of service attack. Moreover, if a firewall rule allows this connection, the implementation of that rule will be correct but useless.

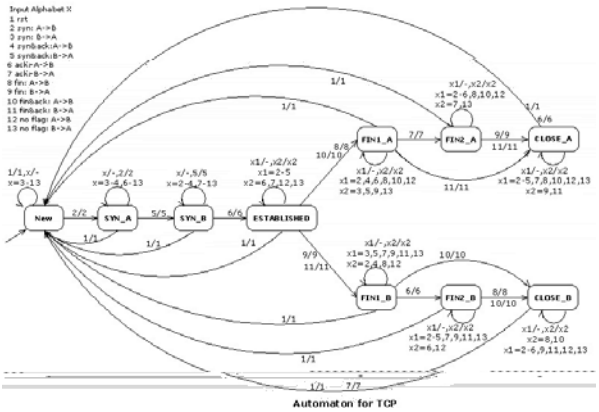

Figure 2. Automaton for TCP [4]

4. Conclusion

A combined approach for testing firewalls is introduced. At the moment we apply this approach to some well-known open source firewalls. The next step will integrate the knowledge of protected network by considering penetration testing goals. We plan to develop also protocol-specific test suites.

References

[1] E. Al-Shaer, and H. Hamed, "Discovery of policy anomalies in distributed firewalls", *Proc. INFOCOM*, IEEE, 2004, vol.4, pp. 2605-2616.

[2] T. Tuglular, "Test Case Generation for Firewall Implementation Testing using Software Testing Techniques", *Proc. 1st Internat'l Conf. on Security of Inform. & Networks*, Trafford Publ., N. Cyprus, 2007, pp. 196-203.

[3] F. Belli, C.J. Budnik, and W.E. Wong, "Basic Operations for Generating Behavioral Mutants", *Proc. 2nd ISSRE Workshop on Mutation Analysis*, IEEE, 2006.

[4] D. Senn, D. Basin, and G. Caronni, "Firewall Conformance Testing", *Proc. 17th TestCom*, LNCS 3502, Springer, 2005, pp. 226-241.