Specification-Based Testing of Firewalls*

Jan Jürjens[1] and Guido Wimmel[2]

[1] Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, Great Britain
tel. +44 1865 284104, fax +44 1865 273839, jan@comlab.ox.ac.uk
[2] Department of Computer Science, Munich University of Technology
TU München, 80290 München, Germany
tel. +49 89 289 28362, fax +49 89 289 25310,
wimmel@informatik.tu-muenchen.de

Abstract. Firewalls protect hosts in a corporate network from attacks. Together with the surrounding network infrastructure, they form a complex system, the security of which relies crucially on the correctness of the firewalls. We propose a method for specification-based testing of firewalls. It enables to formally model the firewalls and the surrounding network and to mechanically derive test-cases checking the firewalls for vulnerabilities. We use a general CASE-tool which makes our method flexible and easy to use.

1 Introduction

The increasing connection of businesses and other organisations to the Internet poses significant risks: Attackers from the Internet may exploit vulnerabilities in the internal hosts connected to the Internet to gain unauthorised access to the corporate network. Due to the complexity of computer systems, it is impossible to protect an internal host just by making sure that it has no vulnerabilities.

This motivates the use of firewalls [4] to protect a network from the Internet, or subnetworks from each other. Incoming and outgoing traffic is filtered and possibly dangerous services are blocked. Firewalls are complex systems composed of several hard- and software components the correct design of which is difficult, in particular for networks that use more than one firewall (e. g. larger companies). Here, the interplay between the firewalls could introduce vulnerabilities. Absolute correctness of the firewall design and implementation is vital since a single weakness allowing unauthorised access makes it fail. However, testing firewalls is usually confined to applying simple check lists (e. g. [5]), possibly using specialised tools (such as [6]); the reliability of the process depends on the skill of the person in charge.

We propose an alternative approach: we formally model a firewall system, and derive test sequences automatically from the formal specification — following the approach to specification-based testing of [17,18,13]. Testing the firewall with

* This work was partially supported by the Studienstiftung des deutschen Volkes, and by the German Ministry of Economics within the FairPay project.

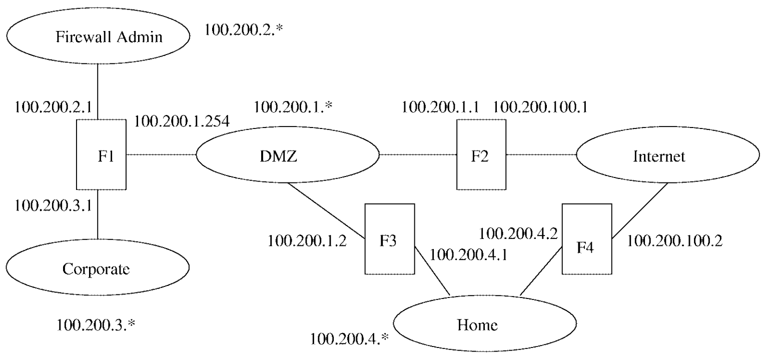

Fig. 1. The Network

these test sequences provides more confidence that the firewall implementation actually provides the desired protection, than ad-hoc testing, especially since the test-sequences are derived with respect to the actual network topology. Our approach is embedded in an easy-to-use CASE framework [9]. Because of its generality, there are few restrictions on the model: Firewall rules need not be of a special form, *stateful* firewalls can be modelled etc. The network model is also flexible, allowing to model possible faults or Trojan horses (malicious code injected by attackers) at the hosts. Various scenarios, such as stress test, spoofing (source address forging), and policy violations, can be tested. Additionally, one can check the firewall specification with a model-checker.

In the next section, we explain our approach using an example network. We then point to related work and conclude.

2 Testing Firewalls

In our approach, we give a (possibly partial) description of a network behaviour that presents a potential threat. From this, a test-sequence is derived automatically which indicates how the system should react to this threat according to the specification. This test-sequence can then be used for actual testing of the firewall.

A further advantage of specification-based testing is that, given a sufficiently detailed specification, one can also determine which values internal variables need to have at certain points of the execution and can use this for more detailed debugging.

2.1 Example Network

We consider an example (in the following called the Network) similar to the one given in [3] (see Fig. 1).