Specification-Based Testing of Firewalls*

Jan Jürjens[1] and Guido Wimmel[2]

[1] Computing Laboratory, University of Oxford***
[2] Department of Computer Science, Munich University of Technology†

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.

This is the web-version 13/8/01 of a paper at Perspectives of System Informatics 2001, LNCS, Springer Verlag. Please refer to www.jurjens.de/jan for the current version and other material.

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 [CB94] 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. [ES99]), possibly using specialised tools (such as [Fre98]); 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 [Wim00, WLPS00, LP00]. Testing the firewall with 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.

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

*** jan@comlab.ox.ac.uk, tel. +44 1865 284104, fax +44 1865 273839 - Wolfson Building, Parks Road, Oxford OX1 3QD, Great Britain

† wimmel@informatik.tu-muenchen.de, tel. +49 89 289 28362, fax +49 89 289 25310 - TU München, 80290 München, Germany

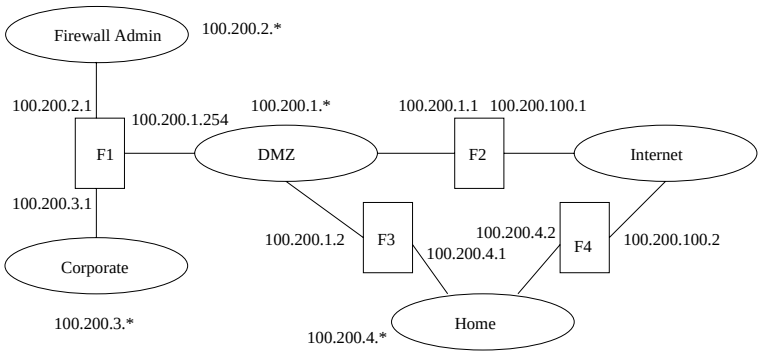

Fig. 1. The Network

Our approach is embedded in an easy-to-use CASE framework [HMR+98]. 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 [BMNW99] (see Fig. 1).

Each interface has its own IP-address. The DMZ (*demilitarised zone*) contains a web-server, a DNS-server and a mail-server.

Here we consider IP-based packet-filtering firewalls. The firewalls should implement the *rule-bases* that are specified below. Each rule specifies a *source*, a *destination*, a *service-group*, the direction of the packet, and the *action* to be taken. The source and destination are host-groups (sets of IP addresses), the service-group specifies a set of services (such as http/https, smtp (e-mail), dns etc.), and the actions

we consider here are to *drop* the packets of the corresponding session, or to let them *pass* (for space limitations we do not consider other actions that may be possible, such as writing a log). The service corresponding to the packet can generally be inferred from the number of the source or destination port given in the packet. Here we can only give a few examples of such rules:

- let packets pass only if the direction of the packet complies with the topology of the network wrt. its source and destination (what this means should be obvious in our simple example, e. g. F3 should let a packet out to the Home subnet via the connection 100.200.4.1 only if its source is in the Firewall Admin, the DMZ or the Corporate subnets and the destination is in the Home subnet, F4 should let packets from the Internet into the Home network only if the source address is in the Internet and the destination address in the Home network, etc.)
- let packets with source in the Corporate subnet and destination in the Internet through
- let packets between the mail-server and the Internet, or the Corporate or Home subnets pass if the service is smtp
- let packets between the DNS-server and the Internet, or the Corporate or Home subnets pass if the service is dns
- let packets from the Internet, or the Corporate or Home subnets to the web-server pass if the service is http or https

For example, a packet with source address in the DMZ and destination address in the Internet will be dropped by F3 (by the first rule, since its destination address is not in the Home subnet) and can only go directly via F2 to the Internet (and similarly for the reverse direction).

2.2 Formal Model

We modelled the network containing the firewalls with help of the CASE tool AutoFocus/Quest. AutoFocus[HMR+98, HMS+98] is a tool for graphically specifying distributed systems. It is based on the formal method Focus, therefore the models have a simple and clear formally defined semantics. AutoFocus supports different views on the system model, describing structure, data types, behaviour and interactions. These views are related to UML-RT diagrams. In addition to modelling, AutoFocus offers simulation, code generation, test sequence generation and formal verification of the modelled systems.

The **structural view** on our example network system is depicted in Figure 2, as an AutoFocus system structure diagram (SSD). Each network component (subnets and firewalls) is an AutoFocus system component, drawn as a rectangle. These components can exchange data via named channels, which connect output and input ports (filled and empty circles). In addition, there is a special component `PacketGenerator` that produces a random packet and sends it to one of the other components, so that it can start travelling through the network.

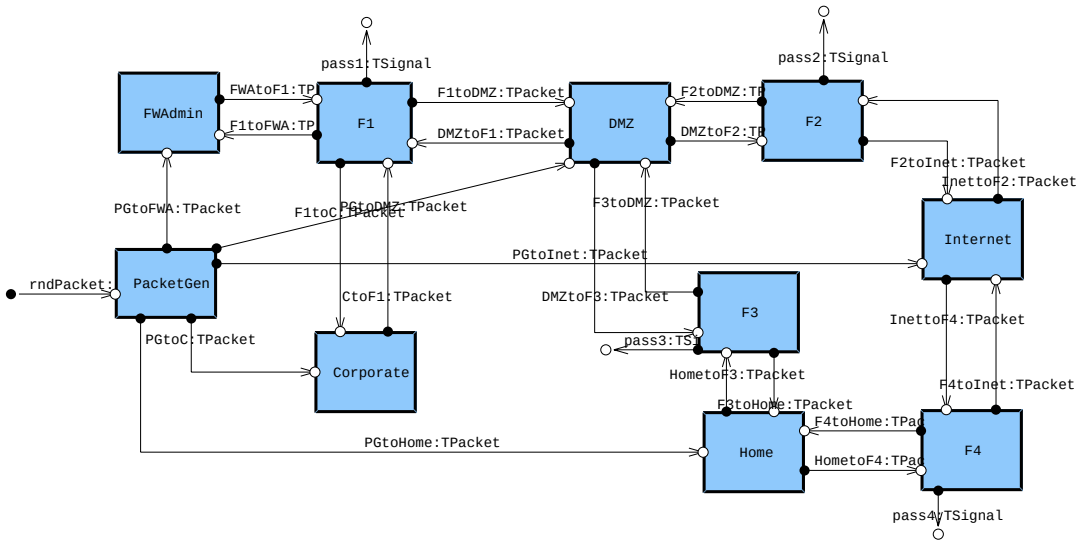

Fig. 2. SSD for firewall system

The **data type definition** in the model describes network packets, and a function for a consistency condition on packets. As other data types in AutoFocus, they are defined in a functional style, as follows:

```
data TAddress = FWAdmin | Corporate | DMZ | Internet | Home |
                F1 | F2 | F3 | F4 | Mail | DnsSrv | WWW;
data TService = Http | Https | Smtp | Dns | Ping | Ssh;
data TPacket  = Packet(source: TAddress, dest: TAddress,
                service: TService);
data TSignal  = Present;
fun srvKons(Mail,Smtp) = True   |  srvKons(DnsSrv,Dns) = True
  | srvKons(WWW,Http)  = True   |  srvKons(WWW,Https)  = True;
```

Finally, each component in the network model is assigned a specified behaviour, using **state transition diagrams** (STDs). STDs correspond to extended finite state machines (meaning they can have a data state as well as a control state, and communicate with the other state machines). The state transition diagrams for the packet generator and the subnets are straightforward — in the subnets of our current network model, the packets are just relayed at random to other output ports. However, the model is very flexible in this respect, so the functionality could also be changed such that the packets may be manipulated.

For the behavioural model of the firewalls, as exemplified by the firewall F1, see Figure 3. Corresponding to each input port of the firewall (which models one of the interfaces of the firewall), there are a number of state transitions, corresponding to forwarding rules of the firewall.[1] As an example, the highlighted transition `DRule2` in Figure 3 can fire if a packet arrives at the interface `fromDMZ` connected to `DMZ`,

[1] Note that often rules in firewalls are presented as sequences, and not as sets like here, but this makes no difference provided the rules are consistent.

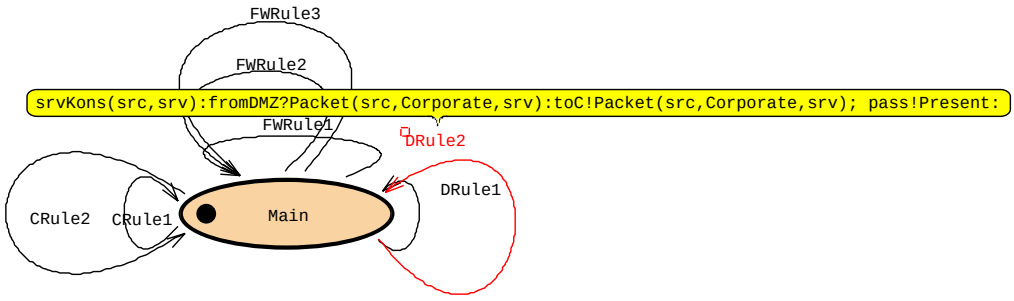

Fig. 3. STD for firewall F1

with destination address `Corporate`, and with a service consistent with its source address. The correspondence is modelled by the function `srvKons` defined above. In case the transition fires, the packet is forwarded to the port `toC`, and the signal `Present` at the output port `pass` gives an indication that it was passed. The other transitions model other forwarding/dropping rules in an analogous way (`FWRule1,2,3` for packets arriving from the `FWAdmin` subnet and `CRule1,2` for packets arriving from the `Corporate` subnet).

AutoFocus is based on a time-synchronous communication scheme, so an execution consists of a sequence of clock cycles.[2] Thus, the forwarded packet can be read by the connected components in the following clock cycle. In all, the highlighted rule corresponds to part of the firewall behaviour as specified in the previous section: packets from the Mail-, DNS-, or Web-Server to the `Corporate` subnet are only passed if the corresponding service entry is correct.

2.3 Testing

In our approach for firewall testing, we use the formal AutoFocus model as a specification to generate test cases. We call this *specification-based test sequence generation* (see e.g. [WLPS00]). For this purpose, first test case specifications based on the system model have to be formulated. Test case specifications would be, for example, that we look for executions where a packet arrives at a certain interface of a component, or executions where a packet is dropped by a firewall. These are translated into logic and solved. The solutions are all test cases of a given maximum length satisfying the test case specification. These test cases represent concrete system executions (which exact packets with which data originate at which component, the way they travel etc.), can be depicted as message sequence charts and fed into the actual implementation of the firewall system for testing.

[WLPS00] describes a test sequence generation approach based on a propositional solver (SATO), whereas in [LP00], a variant (also for AutoFocus) is presented that is based on constraint logic programming (CLP). We used the CLP variant for our purposes, as the CLP-solver proved to be more flexible with respect to specification and generation of the solutions.

[2] Note that this is not a restriction (even though we cannot assume a global clock in a network) because the actions specified here do not depend on global timing.

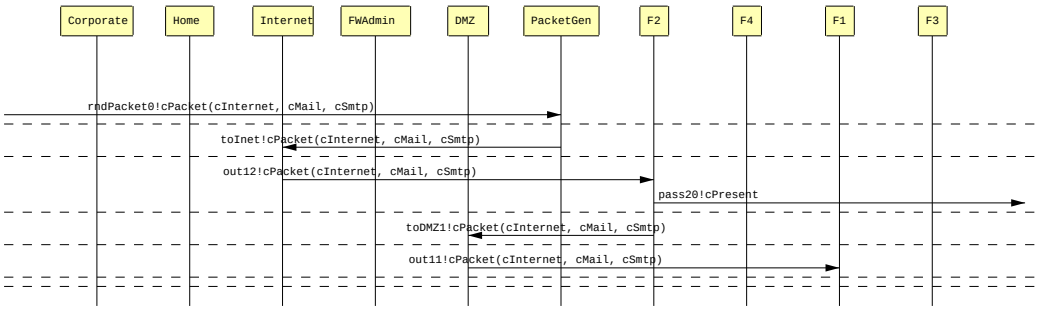

Fig. 4. Test Case for firewall FW1

Thus, with our approach we can systematically generate many (or even all) test cases of a given maximum length to verify chosen security aspects of the firewall implementation. This leads to an improved reliability of the system resulting from the test, as opposed to ad-hoc testing.

In general, our approach supports all kinds of test scenarios that can be specified based on the execution history of the system. Important test scenarios for threats against the firewall example system we tested include the following:

- **Stress test.** For a chosen firewall component, generate all test cases from the specification, where this firewall drops an arriving packet. Thus, the firewall system to be tested can systematically be flooded with packets it should drop. As an example, for the firewall component `F1` we have to specify that a packet arrives at one of the input ports, but no `pass` indication is given in the next execution step. See Figure 4 for a corresponding MSC. This MSC shows the route of a packet coming from the Internet that arrives at the DMZ with destination address being the Mail server (so it is correctly passed on by F2), but is then incorrectly relayed to F1 inside the DMZ — for example by a Trojan horse.
- **Spoofing.** In this scenario, packets with forged source addresses are exposed to the system. For example, an attacker on the Internet may try to send packets to the Network that appear to have originated from internal hosts in the Network in order to defeat security mechanisms (such as the inner fire-walls) that rely on the source information of packets. These test cases can be generated in a similar way as above, by specifying that the packets generated by the `PacketGen` component should have forged source addresses. Spoofing is described in more detail in [Bel89].
- **Policy violations.** As explained in [Gut01], firewall systems have to be based on a security policy. In [Gut01], this policy is given in the form that, if a packet *was in* a certain subnet, and *reaches* another subnet, a certain condition on its source and destination address and service has to be fulfilled. These policy statements can also be used for test sequence generation. Two variants are possible — first test case specifications which fulfill the policy rules and thus check if the firewall system correctly lets packets pass, and second test case specifications that violate the policy. The latter can be derived by negating the condition on the packet

data and lead to test cases that check if the firewall correctly drops packets. The execution in Fig. 4 actually represents both a test case for fulfilment of the policy given in section 2.1 (smtp packets from the Internet to the mail server can be passed to the DMZ) and a violation (those packets must not be relayed on to the `Corporate` or `FWAdmin`) subnets.

Systematic Selection of Test Sequences. For a given test case specification, the number of test cases can get fairly large — especially with more complex data types than in our example. In our case, we can use domain-specific knowledge to improve coverage. Firstly, the maximum length of the test sequences can be restricted to the diameter of the network, provided that only one packet at a time is considered and the components do not maintain states. In addition, we can separately generate a number of test cases with fixed packet origin, source address, destination address or service specified in addition to the original test case specification, to ensure these possibilities are tested. In a further stage, this is also possible for certain combinations of these parameters (e.g., tests where packets originating from the Internet with service "Smtp" are involved etc.).

Performance. Computing test cases of the above kind for our example network takes about .1s per test case, measured on a SUN UltraSparc with 1GB of main memory running at 400MHz.

3 Related Work

[Gut97, Gut01] introduces a language for expressing global network access control policies and algorithms to compute filters for such policies and to check filters against policy violations. [BMNW99, MWZ00] present a firewall management toolkit. Contrary to ours, the intention of this work is not firewall testing, but managing the configuration, and it is assumed that all filtering devices work properly [MWZ00, p.2]. [RA00] uses model-checking to analyse network vulnerabilities.

A Petri-net model of firewalls is given in [Sch97]. References to work on specification-based testing are in [WLPS00]. The Focus method underlying AutoFocus has previously been used for systems security, e. g. in [Jür01b, Jür01a, AJ01].

4 Conclusion and Future Work

We proposed a method for specification-based testing of firewalls, enabling one to formally model the firewall and the surrounding network and to mechanically derive test-cases checking the firewall for vulnerabilities. We used a general CASE-tool which makes our method flexible and easy to use. We demonstrated our approach with an example firewall.

In future work we will consider advanced network and firewall designs, such as authentication headers (using cryptography), virtual private networks, and distributed firewalls. It would be desirable to have a higher-level language for the security policies that is automatically translated into rules (following [Gut01]).

Also, our approach opens up the possibility to go beyond test-sequence generation and perform the actual testing automatically, on an actual firewall system.

5 Acknowledgement

Helpful comments by Joshua D. Guttman on a draft are gratefully acknowledged. Many thanks also go to Alexander Pretschner for helpful suggestions on constraint-based test sequence generation and comments on this paper, and to Heiko Lötzbeyer for the implementation work.

References

[AJ01] M. Abadi and Jan Jürjens. Formal eavesdropping and its computational interpretation. In *Theoretical Aspects of Computer Software (TACS '01)*, LNCS. Springer, 2001.

[Bel89] S. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communication Review*, 19(2):32–48, 1989.

[BMNW99] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. In *Security and Privacy*, 1999.

[CB94] W. Cheswick and S. Bellovin. *Firewalls and Internet Security: repelling the wily hacker.* Addison-Wesley, 1994.

[ES99] UK IT Security Evaluation and Certification Scheme. UK ITSEC Certification Report No. P117 – CyberGuard Firewall for UnixWare, 1999.

[Fre98] M. Freiss. *Protecting Networks with SATAN.* O'Reilly, 1998.

[Gut97] J. Guttman. Filtering postures: Local enforcement for global policies. In *IEEE Symposium on Security and Privacy*, 1997.

[Gut01] J. Guttman. Security goals: Packet trajectories and strand spaces. In R. Gorrieri and R. Focardi, editors, *Foundations of Security Analysis and Design*, LNCS. Springer, 2001. Forthcoming.

[HMR+98] F. Huber, S. Molterer, A. Rausch, B. Schätz, M. Sihling, and O. Slotosch. Tool supported Specification and Simulation of Distributed Systems. In *International Symposium on Software Engineering for Parallel and Distributed Systems*, pages 155–164, 1998.

[HMS+98] F. Huber, S. Molterer, B. Schätz, O. Slotosch, and A. Vilbig. Traffic Lights – An AutoFocus Case Study. In *1998 International Conference on Application of Concurrency to System Design*, pages 282–294. IEEE Computer Society, 1998.

[Jür01a] Jan Jürjens. Composability of secrecy. In *International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS 2001)*, volume 2052 of *LNCS*, pages 28–38. Springer, 2001.

[Jür01b] Jan Jürjens. Secrecy-preserving refinement. In *Formal Methods Europe (International Symposium)*, volume 2021 of *LNCS*, pages 135–152. Springer, 2001.

[LP00] H. Lötzbeyer and A. Pretschner. Testing concurrent reactive systems with constraint logic programming. In *2nd Workshop on Rule-Based Constraint Reasoning and Programming*, Singapore, 2000.

[MWZ00] A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In *IEEE Symposium on Security and Privacy*, 2000.

[RA00] R. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *IEEE Symposium on Security and Privacy*, 2000.

[Sch97] C. Schuba. *On the Modeling, Design, and Implementation of Firewall Technology.* PhD thesis, CERIAS, Purdue, 1997.

[Wim00] G. Wimmel. Specification Based Determination of Test Sequences in Embedded Systems. Master's thesis, Technische Universität München, 2000.

[WLPS00] G. Wimmel, H. Lötzbeyer, A. Pretschner, and O. Slotosch. Specification Based Test Sequence Generation with Propositional Logic. *Journal on Software Testing Verification and Reliability*, 10, 2000.