Artif Intell Rev (2008) 29:223–248
DOI 10.1007/s10462-009-9147-0

A formal logic approach to firewall packet filtering analysis and generation

John Govaerts · Arosha Bandara · Kevin Curran

Published online: 5 November 2009

Abstract Recent years have seen a significant increase in the usage of computers and their capabilities to communicate with each other. With this has come the need for more security and firewalls have proved themselves an important piece of the overall architecture, as the body of rules they implement actually realises the security policy of their owners. Unfortunately, there is little help for their administrators to understand the actual meaning of the firewall rules. This work shows that formal logic is an important tool in this respect, because it is particularly apt at modelling real-world situations and its formalism is conductive to reason about such a model. As a consequence, logic may be used to prove the properties of the models it represents and is a sensible way to go in order to create those models on computers to automate such activities. We describe here a prototype which includes a description of a network and the body of firewall rules applied to its components. We were able to detect a number of anomalies within the rule-set: inexistent elements (e.g. hosts or services on destination components), redundancies in rules defining the same action for a network and hosts belonging to it, irrelevance as rules would involve traffic that would not pass through a filtering device, and contradiction in actions applied to elements or to a network and its hosts. The prototype produces actual firewall rules as well, generated from the model and expressed in the syntax of IPChains and Cisco's PIX.

Keywords Formal logic · Internet security · Fire walls · Logic programming · Network modelling

J. Govaerts · A. Bandara (✉)
The Open University, Walton Hall Campus, Milton Keynes, MK7 6AA, UK
e-mail: a.k.bandara@open.ac.uk

K. Curran
School of Computing and Intelligent Systems, Faculty of Computing and Engineering, University of Ulster, Londonderry, Northern Ireland, UK
e-mail: kj.curran@ulster.ac.uk

1 Introduction

In today's world, information has become all pervasive. The global media business sells masses of information everyday and Screen Digest (2007) in its January 2007 report about the broadcast and media technology business quotes its value at USD 11 billion, forecast to grow at a rate of 12% every year until 2010. This has been made possible, in part, through the development of the Internet. The Internet Systems Consortium, Inc (2007) shows the number of hosts on the Internet to be in excess of 433 million as of January 2007. The Internet has grown incredibly, as an increasing number of organisations have relied upon this medium to facilitate or improve their operations. Some use it to advertise their presence and their capabilities, others to gather information and find new opportunities, others still incorporate the so-called e-commerce to increase their revenue by widening their customer base or retaining existing relationships (Desmet et al. 2006). Unfortunately, those capabilities rest on a number of principles which expose those who are connected to vulnerabilities and an organisation can become a victim of various classes of attacks. Table 1 sums up the frequency of attacks reported, classed by types, with an estimation of the losses incurred following those attacks.

After having experienced computer intrusions, 70% of the respondents patched the holes that allowed the intrusions to take place, 30% did not report the attacks, 25% reported them to law enforcement authorities and 15% reported them to legal counsel. Those who did not report the intrusions, chose not to because they feared the negative publicity would hurt them (reported by 78%), because they feared competitors would use this knowledge to their advantage (reported by 36%), because civil remedy seemed the best course of action (reported by 27%) or because they were unaware of the law enforcement interest in such cases (reported by 22%). This highlights the importance of prevention. In spite of those risks, as it has become inconceivable to conduct business without widespread communication, the enterprises have to manage their security with increased efficiency. Therefore, a number of tools have reached the market and, among them, firewalls have become a centrepiece. Security is an important matter and that a lot of effort has been put in the development of theories, concepts and tools, all forming the foundation for organisations to find a compromise between the protection of their resources and the need to access information or share it with others. To date however, there seems to be comparatively little work based on logic programming techniques.

Organisations should be concerned about the security of their information systems. There are vast financial resources at stake. On the one hand, taking advantage of the global

Table 1 Types of attacks with their frequency and financial impacts (Gordon et al. 2006)

Type of attack	Experienced by %	Losses (thousands USD)
Insider abuse of net access	42	1,850
Unauthorised access to information	32	10,617
Denial of service	25	2,992
System penetration	15	758
Theft of proprietary information	9	6,034
Financial fraud	9	2,557
Misuse of public web application	6	270
Web site defacement	6	163
Sabotage	3	260

communication environment may boost the economic situation of enterprises. On the other hand, the means to guarantee an efficient security may involve substantial investments that weight heavily on their financial health. Past decades have seen a significant increase in the usage of computers and, more recently, the emphasis has been on the capabilities of systems to communicate with each others. This trend has put a heavier weight on security to help organisations protect their electronic properties. In this domain, firewalls have proved themselves an important piece of the overall architecture, as the body of rules they implement actually realises the security policy of their owners. Unfortunately, there is little help for their administrators to understand the actual meaning of the firewall's rules and this becomes even more difficult as the body of rules is usually created in an ad-hoc manner with little opportunity for in-depth analysis. This work shows that formal logic is an important tool in this respect. In an ideal logic programming environment, the programmer makes a series of true assertions and the interpreter, using some logical inference process, run those as a program to solve a problem. The declarative nature, the potential power of expression and the flexibility make the difference when compared with other strategies. This approach is particularly apt at modelling real-world situations and its formalism is most relevant to reason about such models. Logic may be used to prove the properties of the elements it represents and is a sensible way to go in order to create the models on computers to automate relevant activities. Prolog was chosen to create a prototype to present a practical implementation of the concepts described. Its name stands for PROgramming in LOGic and it is based on the mathematical notions of relations and logical inference. It differs from other languages in that, rather than running a program to obtain a solution, it prompts users to ask questions. The databases created for the prototype consist of facts presenting network models and the body of firewalls' rules applied to their components, along with a number of logic rules to reason about the facts, made available for firewalls' administrators to query in order to detect anomalies and reason about the security policy.

2 The need for network security

Attacks on the Internet are a frequent occurrence, most of them using software available on the Internet, such as probes (Guster and Hall 2001). The Internet is an influence on criminal, as well as legitimate, activity and that the growth of Internet access (28.5 million adults on line in 2006, up from 3.4 million in 1996) has spurred the number of illegal acts (Garlik 2007). In 2006 there were 92,000 reported cases of identity theft and fraud, of which 40% were facilitated online, and expected to increase because the fraudsters increase their technical sophistication; 207,000 cases of financial fraud, up 32% from 2005; 1,944,000 cases of online harassment, of which at least 90% remain unreported; 6 million virus incidents, leading to 100 prosecutions. For integrated value chains to reach their potential, security must be in place to give trading partners and customers' confidence that their transactions are handled safely by the network. Therefore, network security services make one of the major elements of interoperability between business partners and firewalls represent one of the primary mechanisms for protecting an organisation's internal network and computers from the outside (Yang and Papazoglou 2000).

Security requirements can be defined in a policy hierarchy as objectives and their corresponding implementation policies as concrete plans to realise them. Hence, access control, security coverage, content access and security association requirements define classes of security requirements, the conditions that are to be met to satisfy them lending themselves to a formal, logic-based description (Fu et al. 2001. A corporate security policy must be expressed in order to deal with the needs of the enterprise and with the risks it is ready to accept in mind. There are two types of policies: one is authorisation-based to specify activities permitted or forbidden, the other is obligation-based to specify activities a subject must or must not perform. Policy inconsistencies may arise from omissions, errors or conflicting requirements (Lupu and Sloman 1997). Conflict analysis deals with the following subjects: modality conflicts as authorisation and obligation policies, and application-specific conflicts that are related to the required consistency of the policies with external criteria. There is a number of concepts to deal with conflict detection (e.g. identification of overlaps between policies), with special emphasis on the role of policy precedence relationships. There is a need to bridge the gap between policy statements and their enforcing configurations. For instance, a policy condition can be tested against a number of properties in order to enforce the constraint when the condition is matched and given some information about the topology of the environment, the mapping from global intent to local mechanism is well suited to translation automation (Hinrichs 1999).

2.1 Network security

Firewalls have become relevant tools to help organisations realise their intended level of security. They may be compared to a router but additionally they are a protection from danger based on the principle that some packets are not forwarded from one interface to the other on a multi-homed host (Hunt 1992). However, they have evolved to become quite sophisticated systems. Firewall techniques can be classified in two categories: packet filtering and application-level gateways (Srinivasan et al. 1999). Firewalls have seen technological advances, but the tools involved in their management and the means available to their administrators have not kept pace with that evolution. One tool is the Firmato management toolkit (Bartal et al. 1999). It comprises an Entity-Relationship model to represent the roles involved in the security policy and the network topology, a Model Definition Language to define an instance of the Entity-Relationship model through parsing, a Model Compiler to translate the model into firewall configuration files, and a Rule Illustrator to provide the administrators with a graphical representation of the configuration files (Bartal et al. 1999). Direction-based filtering is an approach to help administrators achieve anti-spoofing, egress-filtering and zone-spanning. Anti-spoofing is addressed with the notion of direction to an interface, in order to drop a packet whose source address had been spoofed on the Internet (Wool 2004b). Graph theory is used in the RADIS algorithm of Firmato to check whether the claimed path of a packet is consistent with the network topology (Bartal et al. 1999). Firewall Analysis and Configuration Engine (FACE) deals with configurations of firewalls to prevent spoofing, a problem described as originating from compromised hosts (Verma and Prakash 2005). Such a mechanism is critical because those attacks increase in number and because the Internet Protocol (IP) lacks the property to provide any assurance about the true source of a packet. Therefore, FACE uses the notions of trust assumption, trusted path, implementable policy, identification of unimplementable policies and correctness of generated rule-sets.

A network can configure itself automatically while maintaining global security properties (Burns et al. 2001). This is based on the notion of invariant, a network property that has to be maintained even when the network state changes, for example in response to an attack. The ultimate goal would be a self-configuring network driven by changes in the network itself. The security policy must not be inferred from the network state and the task of policy administration is defined as finding the 'right' configuration for all network elements relative to the given security policy and topology. The policy specification must be separated from the network model to let the network evolve without disturbing the security model and let the administrators express their intents. Multiplying defence devices to increase robustness of the network through redundancy makes the protection task less straightforward as adding interfaces also means adding attack points (Uribe and Cheung 2004). Packet transformation can occur due to issues such as Network Address Translation, Port Mapping or IPSec that transform fields in the packets as they move across the network. One possibility to counter this difficulty is to include a history of the packets as part of their trajectory (Uribe and Cheung 2004). Detection and containment of compromised hosts can be used to deceive traditional protections between the intranet and the Internet, such as firewalls. Self-securing network interfaces provide a scalable extension to intrusion detection and containment facility. One of the ways this can be achieved is by looking inward at a host and watching for misbehaviour. Therefore there is a need for cooperation among filtering devices in order to enforce the security policy (Guttman 1997). This is required whenever several filtering devices and the traversal of a succession of network areas are involved. It involves the formalisation of security goals as policy statements, a network model, an abstraction of addresses and packets, and the assignment of inbound and outbound filtering constraints to each communication interface, referred to as ‘filtering postures’. A specification language allows representation of rules, networks and services, and sets of hosts and services. This formalism leads to reasoning about policies and ‘postures’. Operational Research (O.R.) techniques can optimise the placement of firewalls in large networks (Smith and Bhattacharya 1997). To achieve this, the network must be represented in a way that allows O.R. to deal with the optimisation of cost and delay. Firewalls can help focus the attention to key regions of the network to protect them from the likeliest attacks (Guttman 1997) and there should be cooperation among the filtering devices.

There are those who are more critical regarding the role of firewalls. Firewalls represent a single point where security functions may be aggregated and that they may be used beyond TCP/IP and the Internet. There is a difficulty in using them and their interference with the usability and vitality of the Internet as disadvantages. A lax security within the perimeter, resulting from a false sense of confidence, can be considered more worrying since many attacks are perpetrated from the inside of the network (Oppliger 1997). Optimisation techniques can remain static and do not adapt to the varying dynamics of the network (Acharya et al. 2006). Internet firewalls can be evaluated with security analysis tools such as SATAN which is aimed at the verification of firewall hosts and protected networks, combined with the observation of the possible interactions with the firewalls both from inside and outside the private network (Al-Tawil and Al-Kaltham 1999). Team Cymru (2007) have classified packets with addresses that should never appear in Internet routing tables and have found that 60% of naughty packets belong to either unallocated network blocks or private and reserved addresses as defined in RFC 1918 and RFC 3330, and should therefore be easy to tag. Web Application Firewalls (WAFs) may counter shortcomings of traditional network firewalls, particularly in dealing with application layer attacks (Desmet et al. 2006). Web applications are widespread, but are still error-prone and hosting bugs offering welcome targets for attackers due to their high accessibility and possible profit gain. The subject of rules ordering appears in a number of contributions. Rule ordering creates interdependencies that were neither intended, nor controlled and that it makes understanding the rules more difficult (Burns et al. 2001). However correcting shadowing and redundancy errors by rewriting the rules into a set equivalent to the original one can be done, but where the rules are disjoint, this makes their order irrelevant (Cuppens et al. 2005). There is also a potential untoward effect on performance. Some advocate reordering the rules, but the behaviour of the routers may become difficult to figure out and the actual actions of the set of filters may be changed (Chapman and Zwicky, 1995). Networks that were originally created to share resources and enhance cooperation among their end-users, and were later opened to the outside world, must be designed and managed with security in mind. The issue has made life more complex for the owners of those networks.

2.2 Relevance of performance

The cost of lookup in a filtering strategy may add some latency in operations. Binary Decision Diagrams (BDDs) may represent rules lists and hence facilitate quicker lookup, analysis and hardware support (Hazelhurst et al. 2000). As performance improves, more complex rules become feasible and better security may be guaranteed. The BDDs also constitute the base of change analysis. The Boolean formulas representing the rule-set before and after modification are compared for equivalence, logical implication or other relationships. The need for thorough tests of a set of complex rules is considered as well and this is achieved by replaying actual traffic captured earlier, combined with a production grammar to generate additional traffic. BDDs form the representation of rules in Hazelhurst (1999) as well. This representation lends itself to analysis in order to deal with modifications and to verify the rule-set automatically. A set of 'what-if' questions helps validate the meaning of the rule-set. Highlighting the topic of efficiency, the cost of performing look-ups in a complex list of rules may significantly decrease the performance of the system, in addition to the inherent difficulty in understanding their purpose. Indeed, the ability to classify packets rapidly has seen a number of contributions. A filter of highest priority that applies to the packet can be represented geometrically as a multi-dimensional hyper-rectangle and conflict detection is added by locating overlapping regions where the corresponding actions are in conflict (Eppstein and Muthukrishnan 2001). Another approach is through a combination of filter reordering, priority assignment to individual filters' and fields' priority assignment where conflicts are solved implicitly through filter ordering (Hari et al. 2000). Data structure size reduction plays an important role in allowing the filter base to fit into the high-speed memory of its host (Qiu et al. 2001). Thus, it is important to analyse trading storage for time as well as backtracking searches, set pruning tries structures, compression algorithms and pipelining backtracking searches. Regularities in the rule-set improve the search for the tuple corresponding to a given packet (Srinivasan et al. 1999). This rests on speeding up tuple search with a combination of hashing and additional heuristics involving a tuple pruning algorithm, precomputation and rectangle search.

The majority of flows consists in a small number of large flows (skewed flow size) or in long-lived flows (skewed flow duration) therefore a dynamic firewall rules reordering based on online traffic statistics can be used to optimise packet filtering in order to cope with improvement in network performance (Hamed and Al-Shaer 2006). Aggregated Bit Vector (ABV) is a rule aggregation scheme to enhance scalability (Baboescu and Varghese 2005) based on the observation that existing solutions based on two-field classifications, usually source and destination addresses, do not scale well in one of time or storage on more general cases, where more fields are involved. BPF+ is a framework (Begel et al. 1999) to explore the trade-off between flexibility and performance in packet filtering.

Starting from the assumption that packet classification can become a complex operation that can overwhelm filtering devices, efficient classification schemes that parses a large portion of the packets' headers to define precise decision can be applied (Lakshman and Stiliadis 1998). Ordered Binary Decision Diagrams (OBDDs) are shown to improve performance of firewalls analysis tools in order to process larger bodies of rules (Uribe and Cheung 2004). Security is often thought of as a methodology but the importance of performance, either in terms of processing required or in terms of data structure size is an issue that should not be underestimated. As performance increases, more exhaustive actions may be considered, resulting in better protection (Gupta and Keown 2001).

2.3 Firewall policy modelling

The domain of security is vast and, among its various components, the rules representing the firewall's configuration appear to be critical because they embody the actual security policy of the organisation administrating them. Still, one common difficulty is identified as understanding the purpose of those rules. Therefore, modelling them is a way to improve this understanding. Rules can be defined as sets of filtering fields bound to one action field each. Such a body of rules is complex to manage and systems' administrators require tools in order to perform an efficient job. Policy modelling can be done by a formal definition of rules relations and policy representation (Al-Shaer and Hamed 2003) so then the issue can be treated as a policy anomaly through classification and through a discovery algorithm. Thus there is a need for strategies for rule insertion, removal and modification. Inter-firewall anomalies in distributed environments complete previous considerations on intra-firewall anomalies (Al-Shaer and Hamed 2004), where multiple firewalls are used to specialise the controls as they may be required for given domains behind the firewalls. As it is critical to find out what a set of rules actually does. Geometry can be relied on to formalise each rule as a multi-dimensional, axis-parallel hyper-rectangle (Eronen and Zitting 2001). Thus, packet classification is reduced to locating a point in a partition of space formed by overlaying the rectangles and conflict detection becomes a search to find overlapping regions among the rectangles.

The difficulties in understanding firewall rules may stem from arcane languages, less than ideal user interfaces, order sensitivity and the complexity of interactions between rules (Wool 2001). Naming objects to make them more explicit may help and a simulation of the firewall's behaviour against all possible incoming packets may play an important role in understanding the security policy actually enforced by the rules. The shortcomings of another tool, the Fang system, are addressed in the Lumeta Firewall Analyzer, which describes the firewall connectivity, generates typical queries automatically and uses various sources to name objects. It supports multiple vendors and deals with some specifics from Check Point products. The 'Rigorous Automated Security Management' is a four-step procedure to achieve the intended security (Guttman and Herzog 2003). It consists in modelling all components involved, expressing security goals as properties of trajectories based on the actual path and the claimed path of the packets, deriving algorithms to match the policy and implementing the derived configurations. Network configuration can be represented as a sequence of Horn clauses that must be generated automatically as an abstraction of the effects of the configurations of firewalls, routers, switches (Ou 2005). Those clauses form the basis upon which reasoning tools may conduct 'what-if' analysis. An argumentation framework which is a representation of security requirements and firewall rules can be used in a formal, logic-based fashion (Bandara et al. 2006). Here the tasks of firewall rules analysis should consist in errors detection, properties verification and resolution of anomalies. The Ponder language addresses the requirements for a policy specification language, along with access control policies, constraints on policies and composition of policy specifications (Damianou et al. 2001). Automated rule generation and analysis tools can be used to simplify the configuration of firewalls especially in the cases of distributed firewall models whose inclusion of all components makes their management even more complex (Verma and Prakash 2005). Understanding what a security policy really achieves is critical to manage an IT environment and one way to help administrators understand this is by modelling the rule-sets of firewalls (Gouda and Liu 2004).

2.4 Anomaly detection

Administrating firewalls and their rule-sets is a complex task. Inevitably, errors creep into the configurations and, thus, the detection of anomalies has seen a number of texts covering this topic. There is a clear correlation in firewalls between the number of errors and the complexity of the rule-set as a function of the number of rules, the number of objects (e.g. hosts and networks) and the number of communication interfaces involved (Wool 2004a). New firewalls should be added instead of adding new networks to existing ones in order to keep the rule-set complexity low. Distributed firewalls ensure filtering is executed at endpoints, allowing more detailed information to be processed and opening the door for specialised control (Wool 2001). The unlikelihood of bandwidth bottleneck at the perimeter is seen as another advantage however there does exist increased complexity arising from the need for a central policy to control the filtering actions in a coherent fashion and the need to ensure that every device is protected. Firewall rules can be multiplied to resolve conflicts but this increases the complexity of the rule-set and, thus, the likelihood of errors (Hari et al. 2000). Static analysis tools offer the advantage of full coverage, a characteristic facilities that answer queries do lack (Yuan et al. 2006). Firewall analysis should be fully automated as most security administrators would not know what queries to submit and have difficulties understanding query answers when they are too general therefore tools that are as exhaustive as possible should be used (Mayer et al. 2006). Parsers are required to bridge various semantic discrepancies in order to achieve multi-vendor support but generating automatically a network topology file from the firewall's routing table will save complex coding at a later date. Shadowing is the term used when a rule's packets have already been matched by a previous rule, but that their actions are different. The term correlation defines rules matching some common packets, while their actions are different (Yuan et al. 2006). Generalization is when a rule matches all packets of a previous rule and their actions are different. Redundancy is when rules share identical actions for the same packets (Yuan et al. 2006). Irrelevance describes errors where the source and destination belong to the same zone, implying that such traffic will not pass through a filtering device. The complexity of administering a rule-set may unavoidably create anomalies or errors. Their detection is not a trivial activity.

2.5 Logic programming

The complexity of the security topic leads researchers to contemplate various techniques to deal with the problem. Data mining techniques can be used to analyse traffic log files in order to generate an efficient rule-set and reflect an up-to-date traffic trend. Hence, they achieve five goals: analysis of firewall policy rules, mining of association rules, frequency analysis of each rule, generation of a decision tree to improve the efficiency of the policy rules and detection of anomalies. Their detection of anomalies rests on shadowing, correlation, generalisation, redundancy, blocking legitimate traffic to existing service and allowing traffic to non-existing services. CBR is a paradigm that uses the knowledge from past experiences, stored in a database as cases, to propose a solution in new situations by comparison with the stored cases. In Xie et al. (2005), a graph is the means to analyse the reachability issues between hosts taking every elements into consideration. Sadly, their work remains only theoretical, whilst it provides a deep and very formal formulation of reachability. Still, logic programming techniques look promising because the languages based on the concept of formal logic enable the programmer to specify what has to be computed, instead of coding how the processing will be done (Salus 1998). Logic programming is a declarative method of knowledge representation and programming, based on the idea that the language of first-order logic is well-suited for representing data and describing desired outputs (Dantsin et al. 2001). The syntax of logic programs is biased towards their declarative reading (Ralston and Reilly 1995). Contrary to its imperative counterpart, logic programming combines declarative and procedural knowledge in the same representation. Thus, the declarative statements are more abstract and may correspond to several distinct procedures. One purpose of logic, in the context of computer science, is to model situations in order to reason formally about them. Thus, this allows one to prove that arguments are valid and that they can be defended rigorously and executed on a machine (Huth and Ryan 2004). The latter allows a separation between the reasoning logic and the implementation of the reasoning engine, and favours a clear specification of the reasoning logic. This, in turn, facilitates the incorporation of third-party security knowledge into a global security strategy. Logic programming may still however suffer from performance issues (Ou 2005). Event Calculus formalism is particularly well adapted to an event-driven system and may be combined with abductive reasoning. This association has the capacity of dealing with incomplete specifications of the initial state and can thus be used to analyse the rules, detect conflicts and provide explanations. A meta-policy about priorities may resolve conflicts among the rules and declarative understanding of anomalies facilitates their resolution. The combination of Event Calculus and abductive reasoning is well-adapted to represent policies and managed systems because they are event-driven, and the latter is selected for specifications analysis (Bandara et al. 2003). The role of logic-based tools is emphasised in support of formal reasoning methods and the activity of policy refinement is clearly defined. Event Calculus in conjunction with abductive reasoning can implement policies derived from high-level goals (Bandara et al. 2004; Russo et al. 2002) for instance through error detection, even when only partial requirement specification or partial knowledge about the domain is available. Event Calculus is a suitable technique because it possesses an explicit time structure, which is event-independent and because it is close enough to event-based requirement specifications to permit the automation of mapping into a logical representation. Abduction can be used in a refutation mode to verify global system invariants with respect to event-based system descriptions. In refutation mode, failure to identify a counterexample establishes the validity of the invariant with respect to the system description.

Event Calculus can deal with conflicts even if the text is targeted at the Network Dimensioning part of Quality of Service techniques. Obligation policies can be used to specify management operations that must be executed when a particular event occurs in concurrence with some other supplementary conditions. In analysis, conflict detection can be achieved by using one of the conflict fluents as a goal state of an abductive query (Charalambides et al. 2005). Constraint-based programming languages have elegant theoretical properties (Abdennadher 2001) and their relevance to the proposed research question is that constraint solvers collect, combine and simplify constraints, and they may detect inconsistencies too. Verification, analysis and modification of policies can be based on the Policy Description Language (PDL) where sets of event-condition-action (ECA) rules formulate the policies. Conflict detection and resolution may be dealt with for policies written in PDL. The policies define transducers to map sets of events into sets of actions and those functions are defined using Horn logic programs (Chomicki et al. 2003).

PDL can be guided by a policy specification that can be implemented efficiently and by succinct representations of the policies (Lobo et al. 1999). PDL and ECA rules can be used to trace or complete an event history that triggers an action given a complete history of the system. Hypothetical reasoning is shown to find or complete an event history that exploits partial observations of the history of the system (Son and Lobo 2001). Abductive Logic Programming (ALP) is a computational paradigm to resolve limitations in logic programming with respect to higher level knowledge representation and reasoning tasks (Denecker and Kakas 2002). ALP is seen as a framework for declarative problem solving, suitable for a broad collection of cases. In considering logic programming, the role of abstraction comes to mind and proves to be a recurring theme. The Oxford English Dictionary, in its second edition 1989, defines abstraction as the result of separating in thought, of considering something independently of its associations, of considering an attribute or a quality independently of the substance to which it belongs. To lighten systems and networks managers' tasks, the level of abstraction must be raised to hide the specifics (Wies 1994) and, clearly, a good level of abstraction for the network model simplifies the reasoning process (Ou et al. 2005). Better flexibility in the formulation and analysis of the requirements is provided by viewing preference policies at a higher level of abstraction. Administrators should be able to interact with management tools at the same level of abstraction as that at which the corporate security policy is defined or expressed (Mayer et al. 2006). Misconfiguration and potential security holes may be the consequence of working at too low a level of abstraction. Logic programming does indeed offer several avenues of investigation, even if they are less familiar to most and may still suffer from weaknesses. Among those are performance issues (Ou 2005). Formal logic-based approaches are not intuitive, do not map easily into implementation mechanisms and assume a strong mathematical background (Damianou et al. 2001). Thus, in spite of possible disadvantages, logic programming techniques should be able to help address the research question, particularly through their formalism, their capabilities to help reasoning, their support in analysing a model and their capacities to reach higher levels of abstraction.

We believe that formal logic methods may be applied to help understand the meaning of a body of firewall rules and to detect anomalies among them. Binding reasoning with a higher abstraction model of the environment seems a promising way to achieve a better security for organisations. We propose here an approach which will help security administrators understand the meaning of the configurations they manage, the importance of reasoning and anomaly detection, and the part higher abstraction and formal logic may play. The common goal remains to reach a compromise between the mandatory protection of the resources under the supervisors' responsibility and the need for openness and interaction. To achieve the desired level of security, network administrators need help, The evolution of management tools have not kept pace with the technological advances of the systems (Bartal et al. 1999). Some of these administrators' concerns must be addressed by techniques beyond those offered by standard solutions' vendors (Wool 2004b). The assistance to administrators must involve understanding the real meaning of a configuration. This may be based on interfaces to verify the meaning of firewall rule-sets (Hazelhurst 1999), or use modelling techniques. Modelling may take various forms: based on geometry (Eronen and Zitting 2001), or on a simulation of firewalls' behaviour against all possible incoming packets (Wool 2001). Anomalies or errors in a firewall's body of rules compromise the security of the organisation owning the firewall. Reasoning about security is another way to help achieve the desired goal. This may rest on a representation that lends itself to the activity of specialised tools (Ou et al. 2005), or a method to verify the consistency and completeness of rules (Gouda and Liu 2004). Logic programming looks well suited to the purpose that the last section was devoted to its characteristics and relevance in the context of this work and such techniques favour a clear specification of the reasoning logic, assisted by its separation from the reasoning engine.

3 A formal logic approach to firewalls

Security administrators have to tackle a number of difficulties when working with traditional software to fulfil their everyday tasks. Those difficulties range from the interdependencies of the filtering rules among themselves to the complexity involved in understanding the actual significance of a rule-set, including the trouble of grasping the real consequences of a modification in an existing policy because of network environment evolution or new security requirements. This paper presents our work into a logic-based representation of network elements and filtering rules. Such a model of the environment should help administrators reason independently on the significance of the rules they are creating and their adequacy to the stated security policy. The model would free them from the intricacies of firewall software and let them concentrate on the requirements to meet. On top of this, such a vendor-independent representation would ease evolution of the components, a characteristic quite desirable in today's fast evolving security environment. We believe that the logical formulation of the model eases the detection of errors and anomalies in the body of rules by applying formal logic reasoning on the set of predicates implementing the model. As formal logic is not only apt at helping people reason about the situation being modelled, but is also especially adequate for creating programs to be executed on computers, the model and the reasoning algorithms were implemented in our prototype. Our system allows the evaluation of controlled experiments and proves what sort of tool can be built. Naturally, such a demonstration is limited by nature and could never be compared with the complexity and completeness of the facilities expected from a full commercial application. In order to support the activities of administrators, we address network modelling, reasoning about firewall rules and ancillary activities such as the import of data into the model, the user interfaces or the generation of rules. As one purpose was to show the part logic programming may play to achieve the task, the prototype was implemented with the Prolog language, because it is among the most widely available ones in this paradigm. The facts and rules supporting the reasoning in the prototype were written for the University of Amsterdam's SWI-Prolog, version 5.6.31. It is a freely available[1] implementation based on a subset of the Warren Abstract Machine (WAM), a concept used to improve performance in deductive databases. In addition to the basic language environment, SWI-Prolog comes with some extensions, among which Constraint Logic Programming (CLP) looked promising. CLP provides a way to solve problems that are defined in terms of constraints among a set of variables. Here, a problem is stated by giving a set of variables, the domains from which the variables take their values and the constraints the variables have to satisfy. The solution is found by finding an assignment of values to the variables in order to satisfy all the given constraints (Bratko 2001).

3.1 Network modelling

Network modelling is important we believe as it raises the abstraction level, in order to help administrators think about the security and query the properties of the networks they manage. A model helps hide the network specifics by raising the level of abstraction (Wies 1994). Reasoning about policies and packet filtering based on network models can be performed (Guttman 1997). An independent network model is important and it needs be concise for easy management, formal and machine-readable to allow automated reasoning, vendor-independent, human-readable and extensible to stay open to technological innovations (Burns et al. 2001). This can be achieved with a policy implemented in a current network state

[1] http://hcs.science.uva.nl/QRM/software/.

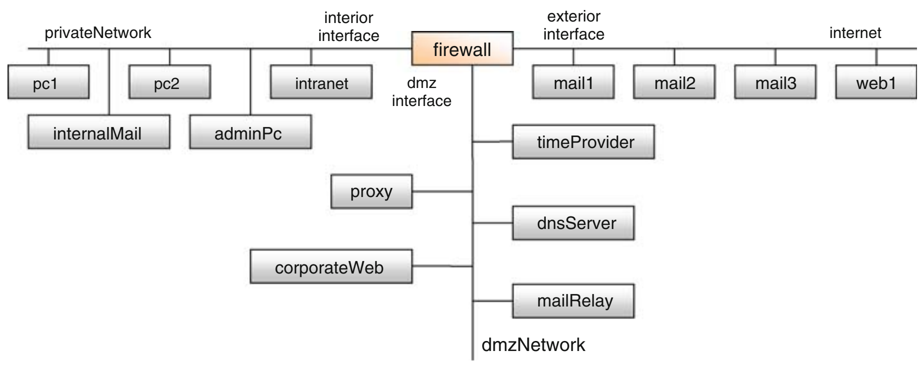


Fig. 1 Multiple-homed host diagram

(i.e. its topology), needs models (i.e. descriptions) of the network elements and services, and notions of device and service interactions. The models should be policy independent and capture the intrinsic properties of devices and services. In this work, the network model is based on an Entity-Relationship (E-R) model, chosen for its universality. To this model, a predicate calculus language is associated to express statements about entities and about their relationships. The choice is made to take advantage of the capacity of predicates to express properties in combination with logical relationships and dependencies (Huth and Ryan 2004). A logical framework should provide a means of representation and of reasoning about permissions. Therefore, a first order language can be associated to express the relationships holding between entities (El Kalam et al. 2003). Then, those facts are translated into the syntax of Prolog to describe the topology of the environment being modelled. On the base provided by this representation, network managers may submit queries in order to improve their understanding of the network properties (Chapman and Zwicky, 1995). The implementations covered the dual-homed host and the screened subnet, with some variations to explore the flexibility of the models. Those representations are described in more details in the following sections.

3.2 Multiple-homed host architecture

A host installed between the internal network of the enterprise and the Internet has one of its interfaces connected to the internal network and another one to the Internet. Compared to a router that checks packets for their destination addresses and forwards them if possible, the host has extended capabilities by being able to control the security policy to determine if the packets should be forwarded. This class of hosts is termed “screening router”. The first model was started on that base, but a third interface was soon added to the host in order to model traffic and filtering rules that could prove a little more complex. Figure 1 is a diagram of this model.

3.3 Screened subnet architecture

To strengthen the architecture, Chapman and Zwicky (1995) add a filtering device and insert a perimeter network between the internal network and the Internet. There is now one host with an interface connected to the internal network and another to the perimeter network. It is named “interior” or “choke” router. The other host has one interface connected to the perimeter network and another one to the Internet. It is called “exterior” or “access” router.

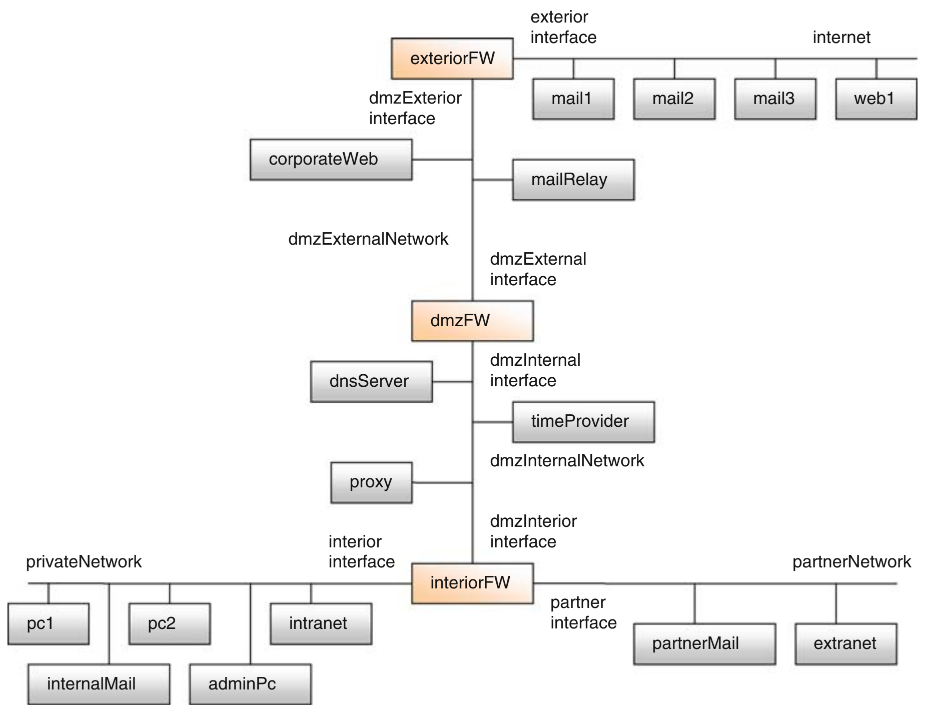


Fig. 2 Screened subnet architecture with cascaded firewalls diagram

The advantage of this architecture is that the security policy of the exterior router may be more restrictive and that the internal network is not threatened when the exterior router is compromised. Besides those advantages, it must be said that this architecture allows security managers to use more complex strategies. As an example, they may decide to run software from different editors on the two systems to protect against weaknesses or bugs in the codes. Again, work was started by adding a filtering device to the earlier version of the model, but, the concept of cascaded firewalls proved more interesting. The model involves three filtering devices in series to study more complex traffic and filtering rules. A "partner" network connected to one interface of the internal firewall was another addition to simulate yet another real-world possibility. In the description, the perimeter networks are identified with the term "De-Militarized Zone "(DMZ). Figure 2 is a diagram of this model.

At this stage, the work could rely upon network models and the question of packet filtering could be addressed. One purpose is to distance the model from the usual, commercial application that takes a device-centric approach. For administrators, the real question should be: what sort of traffic is allowed between a given origin and some destination? The model of a firewall rule is expressed as a five-tuple of the form:

<*action, source address, destination address, destination port, protocol*>

where

action is one of 'allow' or 'deny'
source address and *destination address* may be addresses of hosts or networks
destination port represents the service accessed on the destination address
protocol refers to the protocol used to transit on the network

The security policy specification represents the goals to be reached (Burns et al. 2001) and, thus, the rules included in the model would originally only represent traffic being allowed, because, as the purpose is to reinforce security, everything that is not explicitly allowed should be denied. Besides, this approach helped understanding the meaning of a rule-set as a mixture of allowances and denials makes the comprehension more difficult. Damianou et al. (2001) touch upon the subject by recognising that specification of negative authorisation policies does complicate the enforcement of authorisation in a system. Nevertheless, they note that high level access control is often expressed in terms of both positive and negative policies and, thus, that retaining the natural way people express policies is important and provides greater flexibility.

A mixture of acceptation and rejection is influenced by the sequence of the rules and Cuppens et al. (2004), in their argument about the complexity of rule ordering, conclude that a closed access control policy that includes only permissions (i.e. one with implicit 'deny') may be an alternative, if it is rigorously specified. Still, those default rejections are enforced when packets fail to match any of the rules within the rule-set and, therefore, may result in a performance issue, because they are implemented as the last rule in the set (Acharya et al. 2006). Chapman and Zwicky (1995) encourage a default deny stance because they consider that the fail-safe stance in security strategies should strengthen the protection, and, for Firmato, Bartal et al. (1999) generate a final default rule that drops every packet. We addresses two of Bandara et al. (2006)'s task categories in analysing firewalls: anomaly detection by analysis of the policy specification for potential errors and property checking by conducting "what-if" analysis to determine if a given class of traffic is blocked or allowed. The third task category quoted in the paper, anomaly resolution, has not been dealt with in this work.

3.4 Network model creation

To create the models, several approaches are possible. The administrators could create the clauses in the language of the reasoning engine, but it does not look like a very promising solution. Another possibility consists in translating the E-R model of the environment into a relational model and providing the managers with a data entry interface. A better possibility yet would be to automate the generation of clauses from the actual (Hinrichs 1999). The actual topology information should be either imported from an existing database (e.g. from some network management tool) or discovered automatically. Usually, the complete topology is unnecessary, only the details near the enforcing devices (i.e. firewalls and routers) being relevant. Some topological facts can be discovered automatically by systems providing network monitoring and control such as NESTOR from Columbia University (Burns et al. 2001. Still, they recognise that other facts will require human input. A scanner, OVAL (Open Vulnerability Assessment Language) can be used to create rules (Ou et al. 2005), and a firewall management tool, Smart Firewall can produce the network configuration. MDL (Model Definition Language) is used to produce a front-end module that converts routing tables into firewall connectivity files (Wool 2001). We also examined albeit in a limited scope, the interfaces between the reasoning engine and its human users. The possibilities to feed the model were dealt with earlier. The other side of the issue is to do with the way the representation may be shown to its users. The reports from the Lumeta Firewall Analyzer (LFA) are expressed in HTML to present the output at many levels of abstraction and from multiple viewpoints (Wool 2001) therefore to provide further flexibility, the results of the queries are dumped into formatted plain text output files, which can be converted by back-end utilities into the desired results. The transformation into HTML Web pages is developed, but the contents of the files could, theoretically, be converted into other formats such as graphic displays or charts.

The purpose of the models is to help administrators think about their security policy and execute queries to iron out any anomaly present. Once the model has reached a valid state, the next phase is to generate firewall rules corresponding to the description of the model. Administrators can help manage a distributed firewall environment with an analysis tool and automatic rule generation (Verma and Prakash 2005). Firewall configuration files can be generated automatically from the security policy to reduce the probability of errors (Bartal et al. 1999). It is shown to be achieved in two phases: the generation of generic rules from the E-R model, followed by a translation into vendor-specific formats (Cuppens et al., 2004). The rules relevant for each firewall among the whole body are extracted and expressed in a vendor-independent language based on the eXtensible Markup Language (XML) syntax. Then, concrete configuration rules, expressed in the specific target firewall language are derived from those original rules, thus providing support for multi-vendor solutions. In the text, an example of such a translation is obtained by applying specific eXtensible Stylesheet Language (XSL) transformations. The NESTOR platform can reconfigure network elements on the basis of the output from their policy engine and it allows the manipulation of network devices through their proxy objects in its repository (Burns et al. 2001). Likewise the Lumeta Firewall Analyzer, rule generation involves two steps (Wool 2001) where the core engine uses a generic and vendor-independent model of the rule-set and, after this, parsers must be able to convert the model while respecting the vendor's firewall semantics. Configuration is through a simple syntax transformation from the canonical form to the particular device syntax (Hinrichs 1999). There is a concrete implementation for Cisco Secure Policy Manager, where the control agents store the configuration into a commands buffer and, upon approbation, 'telnet' in and download the commands into the devices. However, future versions of the tool are expected to use mechanisms for program-controlled downloads (e.g. COPS—Common Open policy Service—or LDAP—Lightweight Directory Access Protocol).

4 Evaluation

Network representation started from an E-R model, to which a first-order language was associated to express the properties of the components in the model. The vocabulary of the language has three classes of symbols.

- Constant symbols: they are instances of the entities in the E-R model (i.e. they belong to one of Protocol, Firewall, Interface, Network, Host or Service).
- Variable symbols: they belong to the same entities as the constant symbols. They are used to reason about the relationships between entities and the filtering rules. They are placeholders for concrete values to avoid having to refer to all instances.
- Relation symbols: they represent relationships between entities in the E-R model.

The elements of the network are represented with predicates, functions from some domain to a truth value as quoted in the Dictionary of Computing (2004). The set of predicates considered is shown in Table 2, but it could easily be extended to suit further specific needs. The arity of the predicates is chosen to provide enough information about the component being modelled.

The relation symbols and their meaning are expressed in Table 3.

The formal logic representation helps administrators reason about their model. A typical query expressed in this language would look as follows:

Table 2 List of predicates

Predicate	Description
CommunicationProtocol(p)	p is a communication protocol implemented on the network
ServicePort(s, p)	Service s is listening on port p for incoming requests
Firewall(f)	f is a firewall. It requires special attention as the subject of the study
Interface(i)	i is the name of a firewall interface
Network(n, s, m)	n is the name of a network covering the IP addresses starting with s and subnet mask m
Host(h, a)	h is the name (e.g. DNS name) of the host having the IP address a

Table 3 List of relations

Relation	Description
$\mathrm{FI}(f,\{i_1, \ldots, i_n\})$	The firewall whose name is f has a set of interfaces i_1 through i_n
$\mathrm{IN}(i,\{n_1, \ldots, n_n\})$	The interface whose name is i has a set of networks n_1 through n_n connected to it
$\mathrm{NH}(n,\{h_1, \ldots, h_n\})$	The set of hosts h_1 through h_n belongs to the network n
$\mathrm{HS}(h,\{s_1, \ldots, s_n\})$	The host whose name is h harbours the set of services s_1 through s_n. This can be the empty set

(1) What hosts are located behind an interface?

A new predicate, HBI, expresses the relationship between an interface and the hosts to be found behind it. It can be derived from existing predicates.

$$\forall i \left(\mathrm{interface}(i) \wedge \mathrm{IN}(i, \{n_1, \ldots, n_n\}) \wedge (\forall n_x \in \{n_1, \ldots, n_n\}, \exists\, \mathrm{NH}(n_x, \{h_1, \ldots, h_n\})) \right.$$
$$\left. \rightarrow \mathrm{HBI}\left(i, \bigcup_x \{h_1, \ldots, h_n\} \right) \right)$$

This representation was easy to translate into a Prolog-based prototype because the language is based on the mathematical notion of relations. The logic predicates were hence translated into SWI-Prolog facts to produce models of various networks. Those facts are listed in Table 4.

There is a one-to-many relationship between interface and network because, while an interface obviously belongs to a single network, it may provide the connection to several networks situated behind it. It is the case in the internal network if there is a non-filtering router behind the interface and it is certainly the case for the interface leading to the Internet. Some predicates have an arity in their Prolog form different from their counterparts in the formal logic representation and predicates have appeared in the prototype in addition to those of the first-order language. These differences can be traced back to the extended functionalities provided by the prototype. As an example, the predicate calculus language defines a ‘firewall(f)’ to declare filtering hosts. In Prolog, the equivalent predicate is ‘firewall/2’,

Table 4 Prolog facts used to model networks

Fact	Description
CommunicationProtocol(protocolName)	Declares the protocols implemented in the network
Firewall(firewallName, softwareName)	Declares the hosts that will filter packets and the software they are running
FirewallSoftware(softwareName)	Declares the filtering software running on the firewalls declared in the model
Host(hostName, iPAddr)	Declares a host with the 4 bytes of its IP address
Interface(interfaceName, IPAddress, networkMask)	Declares the network interfaces of the firewalls
Network(networkName, startIPAddr, endIPAddr, networkMask)	Declares a network with its start and end IP addresses (4 bytes for each) and network mask
ServicePort(serviceName, portNumber)	Associates the name of the services with their port number
FirewallHasInterfaces(firewallName, listOfInterfaces)	Associates the interfaces present on a firewall with their host
HostHasServices(hostName, listOfServices)	Associates a host with the services it supports
InterfaceHasNetworks(interfaceName, listOfNetworks)	Associates an interface with the networks connected to it
NetworkHasHosts(networkName, listOfHosts)	Associates a network with its hosts

where the extra argument allows the prototype to generate filtering rules in the syntax of the target system, a functionality that is not required to reason about the model. The purpose is to model a situation involving more than one firewall and having them run different filtering software. Such a mix may improve the reliability of the architecture by taking care of security holes or bugs in one filtering software. This attribute becomes relevant when generation of firewall rules is considered in Sect. 4.3. Furthering the example, this explains as well the existence of the 'firewallSoftware/1' predicate that appears only in Prolog. In the prototype, the IP addresses of hosts and networks are represented as lists of four integers in order to produce a code that is easier to read. This representation allows the end-user to query the network configuration to reach a higher level of abstraction than is usually possible with firewall software and reason about the properties of the topologies created. Modelling using Horn clauses proved itself well suited to those activities.

4.1 Reasoning about firewall rules

One recurring difficulty seems to be the relationship between intention (i.e. the security policy) and execution (i.e. the body of rules). Thus, keeping in mind that the aim of the research is to investigate if logic programming techniques are relevant in improving the security within a network through their interaction with the body of firewall rules, the first-order language described above proved itself adequate to express firewall rules as predicates. The predicate has the form "rule(a,s_a, d_a, d_p, p)", where

- a is the action to be applied (i.e. allow or deny)
- s_a is the source address of the packet. It may represent a specific host or a network.

- d_a is the destination address of the packet. Again, it may represent a specific host or a network.
- d_p is the destination port of the traffic. Naturally, d_p must be present on d_a.
- p is the protocol with which the packet transits on the network.

Some constraints on this predicate may be expressed with the predicates defined to describe the network:

$$\exists\ (\text{host}(s_a, x) \vee\ \text{network}(s_a, x,x')) \wedge \exists\ (\text{host}(d_a, y) \vee \text{network}(d_a, y,y')) \wedge$$
$$\exists\ \text{servicePort}(d_p, z) \wedge \exists\ \text{communicationProtocol}(p)$$

This helps administrators reason about their security model by querying the sort of traffic allowed and detects automatically some anomalies. For instance, the query "*Is there a firewall rule that specifies a source or destination address that does not exist in the network model*?" is represented by the predicate IS, obtained as follows for a source address:

$$\forall\ s(\text{rule}(a, s, x, x', x'') \wedge\ \neg(\text{network}(s, y, y') \vee \text{host}(s, z)) \rightarrow \text{IS}(s))$$

And similarly by the predicate ID for a destination address:

$$\forall\ s(\text{rule}(a, x, d, x', x'') \wedge\ \neg(\text{network}(d, y, y') \vee \text{host}(d, z)) \rightarrow \text{ID}(s))$$

The corresponding classes of anomalies identified in the prototype are summed up in Table 5.

The prototype was initially written with the explicit definition of authorisations and a default "deny" action to terminate the sequence of firewall rules for the reasons explained in Sect. 3.2. However, experimenting with explicit deny actions proved an improvement to the semantic capabilities of the model. This adaptation was clearly facilitated by the separation of the network topology model from the logic rules themselves. Reflecting about the experience leads to the conclusion that explicit rejection of traffic should be limited to flows between components of the network under control of the administrator and the Internet. This approach has the dual advantage of protecting the elements of the network from unsolicited access, while keeping the rules within that very network easier to understand than they would be otherwise. In its final form, the fact that represents firewall rules has the form "firewallRule (action, source, destination, service, protocol)". Besides the explicit definition of services and protocols in the rules, a special value 'any' was introduced to specify that all services or protocols were covered by the rule. This proved particularly relevant for rules rejecting unwanted traffic (e.g. when traffic between two addresses was to be denied).

The network model is composed of a succession of facts, whereas the reasoning part of the prototype relies on rules, which rested on two different methodologies. Some of the rules were written in the classic Prolog form. To execute them, the end-user enters the predicate and, when a solution is found, has to type the semicolon sign to request backtracking to try and resatisfy the goal from the clauses following the current one. The other methodology rested on the 'fail/0' predicate to force backtracking and present all the possible solutions to the goal without user's intervention. While the first method described above is suited to people familiar with the Prolog environment, the latter one is easier to use and less likely to let administrators miss one occurrence of the solutions to a query. It was systematically used for the generation of firewall rules in the syntax of actual systems as this activity did not require interaction with the end-user. The development of the prototype showed how the declarative form of the language proved valuable in programming error detection. Actually, the rules locating the anomalies could be created by 'describing' each situation resulting in an error and giving them the same header to cover the cases exhaustively. Two extensions provided with SWI-Prolog also contributed to the versatility of the software. The first is Constraint Logic

Table 5 Anomaly detection

Type of anomaly	Comments
Contradiction	Pairs of rules are explored for contradicting actions on the same source and destination; this is considered an error. When contradiction appears between a rule specifying a host and another rule specifying its network, it is considered a warning because it could be an error, but it could as well be the administrator's intention to specify a host exception within a network
Incorrect relationship between a host and its network	A host is either declared without any relationship to a network or its IP address lies outside the range defined for its associated network
Inexistent element	Rules are verified for inexistent sources or destinations (hosts or networks), or for services on a destination host that does not support this service
Invalid format of IP address	Declarations of IP addresses are checked to confirm that they cover a valid range of values
Irrelevance	A rule is termed irrelevant when its source and its destination are located behind the same firewall interface, because this traffic would not pass through a filtering device
Private IP addresses on the Internet	A network is defined as connected to the interface leading to the Internet, but has private addresses (i.e. belonging to one of 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16)
Redundancy	Redundancies detection involves looking for pairs of rules specifying the same action for a host in one rule and for the network to which the host belongs in another rule

Programming (CLP), whose characteristics were described in chapter 3. In this work, CLP was applied to the validation of IP addresses values and to the classification of IP addresses as 'private' as defined in RFC1918.[2] It must be reported that it allowed for a very simple and readable declaration of the rule, clearly better than what could be achieved without this extension. As an example, in checking the validity of IP addresses, the permissible ranges of values were declared as such, instead of relying on expressing them as the conjunction of the extreme values permissible (i.e. greater than or equal to the minimum and less than or equal to the maximum). Similarly, the keyword 'internet' allowed the reasoning process to locate a network declared with a range of private IP addresses where they could never be found in reality. The second extension integrated into the prototype was the Constraint Handling Rules (CHR) tool. They consist in a committed-choice rule-based language embedded in Prolog and are considered useful for providing application-specific constraints. As an application of CHR, the declaration of the relationship between a host and the network to which it belongs was checked with this particular facility. Here, the validity of this relationship is linked to

[2] http://www.arin.net/reference/rfc/rfc1918.txt.

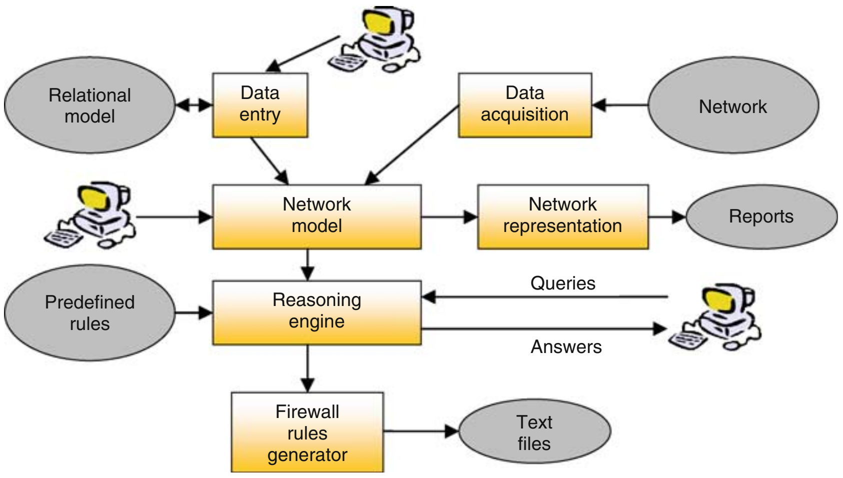


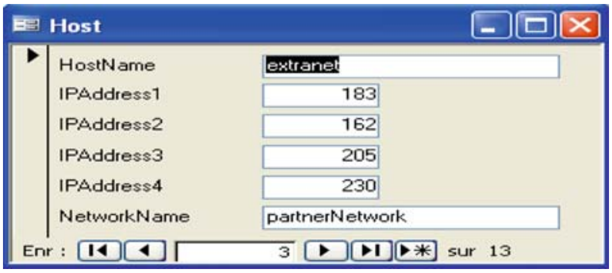
Fig. 3 Prototype architecture

the host appearing in the clauses that specify what hosts are located within a given network (i.e. with the 'networkHasHosts/2' predicate) and to the IP address of the host being within the limits of the IP addresses covered by the network. Once this relationship is declared, it can be used to prove the validity of the model in refutation mode through the Closed World Assumption (CWA) inherent to Prolog's databases. As explained in Sect. 2.1, this strategy establishes the validity of the model through failure to find a counterexample.

The experimentation with CHR ended by declaring a number of constraints each argument of the predicate 'firewallRule/5' has to satisfy to be semantically correct. Each individual constraint was created with simplification rules, one of the three types of rules available in CHR. The results were quite convincing, except for the type detection that was applied to the 'action' argument of the firewall rule. To verify that all the actions belonged to one of 'allow' or 'deny', those legal values were declared in a type, whose violation is detected by SWI-Prolog as a run-time error. When this happens, execution is aborted and no further investigation is possible. However, the action value in error is clearly identified in the error message, allowing the end-user to correct the model of firewall rules body. Rule duplication (i.e. rules for which all components of the predicate are identical) is only processed when actual firewall rules are generated. Semantically, this was not considered an error, but as it could have an untoward effect on performance, the duplicates are simply removed from the rule-set when rules are translated from the model representation into the syntax of the target filtering device. To achieve this, the predicate 'firewallRule/5' is declared dynamic in the Prolog code and the exceeding rules are removed from the database through the 'retract/1' mechanism. The structural tests, conducted to prove that anomalies could be detected, were conclusive. Figure 3 shows a diagram of the architecture of the prototype.

4.2 Data for the network model

Earlier we mentioned several possibilities to create data in order to generate the network model. The first approach mentioned would have network or security administrators writing Prolog clauses in the representation chosen for the model. It is an obvious possibility, though not very promising. To investigate the second possibility indicated, it was straightforward to



HostName	extranet
IPAddress1	183
IPAddress2	162
IPAddress3	205
IPAddress4	230
NetworkName	partnerNetwork

Fig. 4 Data entry screen for host information

transform the E-R model into a relational model, opening the door to a host of possible data supports to store the model. In the relational model, tables have been added that show no relationship to the other tables, but that are used in the prototype to provide a store for data that will be transformed into facts. On this basis, a simple application in Microsoft Access 2003 was written, which includes data entry screens and Prolog facts generation. An example of those facilities is shown in Fig. 4.

In this implementation, the logic rules to reason about the model or detect anomalies are written independently of the network model. This has the advantage of creating and updating the two components without undue interaction. Their integration is achieved with the 'include/1' directive that fetches the logic rules source code from the network model. The Unix 'netstat' command quoted earlier was also included in the evaluation in order to gather routing information. The actual environment was based on Sun Solaris and this showed how dependent upon the flavour of Unix such a facility would be. However, it could be concluded that, by writing parsers specific to all the required servers, it would be feasible to deduce a number of facts about the network. In Sun Solaris, 'netstat—r—anv' was most promising, with the network interfaces being recognisable by the value '255.255.255.255' in the columns 'Destination' and 'Mask'. To create a graph representation of the network model, we used uDraw(Graph) 3.1, a freely available package[3] from the University of Bremen, Germany. Actually, there are two ways in which this can be relied upon in this context. The application can connect to the uDraw(Graph) server using bi-directional TCP/IP socket communication through the API provided, or it can be adapted to generate a file, called "term representation" (extension ".udg"), to be interpreted by the uDraw(Graph) package. For documentation purposes, it is easy to export the graph in a JPEG file and include the picture in a document. Firewall rules could have been added to the graph by using another representation for the edges, but it made for a much cluttered graph and it did not prove helpful at all.

4.3 Firewall rule generation

The prototype was written to generate firewall rules with the information provided in the model. This function considers the traffic between the source and the destination to determine through which interfaces it will have to pass. For each firewall in the model, a set of rules is generated, restricted to those filtering relevant flows. As the traffic flows through more than one filtering device, this approach is related to the concept of multiple lines of defence evoked in the literature review. There are two classes of generation. The first produces a set of rules in an abstract syntax and is intended to be displayed on the screen to help administrators verify what will be generated from the model. The second class creates

[3] http://www.informatik.uni-bremen.de/uDrawGraph/.

```
# IPChains configuration file
ipchains -F
ipchains -P input REJECT
ipchains -A input -i lo0 -j ACCEPT
ipchains -A output -i lo0 -j ACCEPT
ipchains -A input -s 127.0.0.0/8 -j DENY -l
ipchains -A input -p tcp -s 192.168.0.30 -d 140.120.36.30 25 -y -j ACCEPT
ipchains -A input -p tcp -s 192.168.0.30 -d 140.120.36.30 -f -j ACCEPT
ipchains -A input -p tcp -s 192.168.0.30 -d 140.120.36.40 25 -y -j ACCEPT
ipchains -A input -p tcp -s 192.168.0.30 -d 140.120.36.40 -f -j ACCEPT
ipchains -A input -p tcp -s 192.168.0.30 -d 161.120.33.15 80 -y -j ACCEPT
ipchains -A input -p tcp -s 192.168.0.30 -d 161.120.33.15 -f -j ACCEPT
```

Fig. 5 Example of rules generation for IPChains and PIX

a text file with the rules written in the syntax of actual firewall software. The text file bears the firewall's name taken from the model and contains a list of statements that can be copied to the target device's configuration (e.g. using a 'telnet' connection). For this research, two systems were considered: IPChains and Cisco's PIX. IPChains is a simple, static packet filter facility available for Linux. PIX (Personal Internet eXchange) is a stateful packet filtering solution from the well-known networking solutions provider. In essence, the Prolog code generates the rules from the model by translating the appropriate elements into the target syntax (e.g. services into port numbers, hosts names into IP addresses, forms of network masks representation, verbs to specify the action to be applied to the traffic). The information available in the model provided the source for the addition of statements to declare the ports ('fixup protocol' statement) and the interfaces ('ip address' statement) besides the rules themselves for the PIX configuration generation. Similarly, some good practice recommendations (e.g. the loopback address protection) were included in the code for IPChains. The rules were created by starting with those involving the most precise elements (e.g. host to host traffic), to more generic values (e.g. network to network filters) and to finish with the most generic of all, that identified in the model with the name 'internet'. In the generated files, the Internet is represented by a lack of address in the rules generated (i.e. 'any' for PIX and no parameter for IPChains). An alternative could have been to translate the keyword 'internet' into a set of all networks while excepting those explicitly defined and the range of private addresses already quoted. However, this would have created a very verbose set of rules and would not have added a lot to the functionality of the prototype. Still, the keyword 'internet' in the model provided the means to detect the interface that was receiving traffic from the Internet. Thus, for IPChains rules generation, the addition of a control that the packets had to be incoming through this interface (parameter '-i') was included in the generation to show that such information could be relied upon to improve somewhat the level of security in the rule-set. Referring back to the notion of closed system elaborated upon earlier, the sequence of rules is completed with a default deny action (i.e. 'deny ip any any' in the syntax of PIX and 'ipchains—P input REJECT' for IPChains) (See Fig. 5).

From this experience, it is clear that generating rules for sophisticated software such as PIX allows an easier mapping to the model. To deal with the details and limitations of a simpler language such as that of IPChains, the generation code becomes more complex and more difficult to maintain. The aim here was to demonstrate that formal logic methods may be applied to understand the meaning of firewalls' rules and detect anomalies among them in order to improve overall security. While a comparison of this approach with traditional activities in this field is unavoidably limited because this work has only considered theoretical models, it remains interesting. To produce a more thorough evaluation, this method would have to be evaluated in a concrete environment. A first conclusion is that applying logic methods clearly raises the level of abstraction and it has been shown earlier why this is desirable. Another important improvement brought by this solution is that the view on the environment is global, a characteristic that facilitates the comprehension of the interactions when compared with the device-centric methods implemented on most filtering systems. This does not only facilitate the understanding, but also improves the cooperation among the components, a desirable property too. The capacity of the prototype to generate actual rules on different target systems has a double advantage: it relieves the administrators from the intricacies of the target software and it allows them concentrate better on the actual meaning of the security policy without being distracted by the peculiarities of the syntax of each filtering device on which it will be implemented. By running a check on a computer separate from that on which they will be executed, the rules may be verified and their anomalies detected before implementation. Most relevant is that the model created may be queried to help understand its characteristics, an activity seldom available on traditional vendors' packages, and the errors or anomalies may be detected, whereas most classic filtering software only produce compile-time errors without addressing the semantics of the body of rules.

5 Conclusion

We aimed here to investigate whether logic could help improve security in the context of firewalls. It is complex to efficiently manage network security. Abstraction was the starting point of our research and the creation of a model allowing network and system managers to reason about their environment has been demonstrated. Logic has been applied to the models considered in order to facilitate reasoning about the properties of the networks and about the security policy applied to it. Logic programming methods have been chosen to create a simple prototype in order to prove that such an approach was feasible and to confront the realities of such a strategy. However, the models have been created in a purely theoretical way, without the possibility to involve an actual network. There is a permanent underlying argument that logic programming is suited to the ever evolving nature of prototype development. In a context in constant evolution, it is another advantage to be drawn from this research. While reasoning on the properties of the model was obviously a strong element of the prototype, the generation of actual firewall rules in the syntax of existing systems was more ambiguous. The support for different software based on the translation of information available in the model is certainly likely to help administrators and writing for the syntax of systems accepting themselves a comparatively high level of abstraction proved reasonably successful. Other target devices, either because they required instructions at a fairly low level or because of the complexity to cover a sophisticated functionality, did not look as promising.

Our work presented here can be considered as a good starting point and this implies that a number of improvements would be required to start using its concepts in the real world of security. Among those improvements, there is the seamless integration of its various components. As an example of such integration, the prototype could start reading the model stored in the relational database instead of relying on exported text files.

Another improvement would be the support of firewall rules generation in the syntax of systems other than those considered here. This functionality is an interesting base to consider other needs that, in turn, bring improvements to the model. Besides, as the topic of the project was firewall rules, the notions of Network Address Translation (NAT), known as masquerading in IPChains, were not considered. Still, use of the prototype in a real environment would require those to be addressed. Naturally, the ultimate test would be to confront its functionalities with real networks. This would have two facets. First, the generated firewall rules could be tested on actual devices to investigate how they behave when confronted with real traffic.

The other facet of this would be to generate the model from existing networks and bodies of rules. An obvious weakness of the prototype is that the model needs to be created somehow from scratch, a frustrating activity for administrators dealing with an existing environment. NESTOR (NEtwork Self managemenT and ORganization) (Konstantinou 2003) seeks to replace labour-intensive configuration management with one that is automated and software-intensive. This is based on policy rules that access and manipulate network elements via a Resource Directory Server (RDS). The server accesses devices through a layer of adapters that translate its own object-relationship model into the syntax of a number of systems. Even if it has not been tried in the context of this work, it looks like a sensible way to generate data input to the network model automatically.

References

Abdennadher S (2001) Rule-based constraint programming, Habilitationsschrift, Ludwig-Maximilians-Universität, München, Germany, 15 July 2001

Acharya S, Wang J, Ge Z, Znati T, Greenberg A (2006) Simulation study of firewalls to aid improved performance. In: Proceedings of the 39th Annual Simulation Symposium (ANSS'06), Huntsville, 2–6 April 2006

Al-Shaer E, Hamed H (2003) Firewall policy advisor for anomaly detection and rule editing. IEEE/IFIP Integrated Management(IM'2003), Colorado Springs, pp 17–30, 24–28 March 2003

Al-Shaer E, Hamed H (2004) Discovery of policy anomalies in distributed firewalls. In: Proceedings of IEEE INFOCOM, Hong Kong, pp 2605–2616, 7–12 March 2004

Al-Tawil K, Al-Kaltham I (1999) Evaluation and testing of internet firewalls. In: Int J Netw Manage 9: 135–149, Wiley

Baboescu F, Varghese G (2005) Scalable packet classification. In: IEEE/ACM Trans Networking, vol 13, n° 1, pp 2–14 Feb 2005

Bandara AK, Kakas A, Lupu E, Russo A (2006) Using argumentation logic for firewall policy specification and analysis. In: 17th IFIP/IEEE workshop on distributed systems: operations and management (DSOM) 2006, Dublin, 23–25 Oct 2006

Bandara AK, Lupu EC, Russo A (2003) Using event calculus to formalise policy specification and analysis. In: Proceedings of the 4th IEEE international workshop on policies for distributed systems and networks (POLICY'03), Lake Como, 4–6 June 2003

Bandara AK, Lupu EC, Moffett J, Russo A (2004) A goal-based approach to policy refinement. In: Proceedings of the 5th IEEE international workshop on policies for distributed systems and networks (POLICY'04), Yorktown Heights, New York, 7–9 June 2004

Bartal Y, Mayer AJ, Nissim K, Wool A (1999) Firmato: a novel firewall management toolkit. In: Proceedings of IEEE symposium on security and privacy, Oakland, California, USA, pp 17–31, 9–12 May 1999

Begel A, McCanne S, Graham SL (1999) BPF+: exploiting global data-flow optimization in a generalized packet filter architecture. SIGCOMM'99, Aug 99, Cambridge, pp 123–134, 30 Aug–3 Sept 1999

Bratko I (2001) PROLOG programming for artificial intelligence. Pearson Education Ltd, Harlow

Burns J, Cheng A, Gurung P, Rajagopalan S, Rao P, Rosenbluth D, Surendran AV, Martin DM (2001) Automatic management of network security policy. In: DARPA information survivability conference and exposition (DISCEX II'01), vol 2, Anaheim, California, pp 12–26, 12–14 June 2001

Charalambides M, Flegkas P, Pavlou G, Rubio-Loyola J, Bandara AK, Lupu EC, Russo A, Sloman M, Dulay N (2005) Policy conflict analysis for quality of service management. In: Proceedings of the 6th IEEE international workshop on policies for distributed systems and networks, Stockholm, 6–8 June 2005

Chapman DB, Zwicky ED (1995) Building internet firewalls. In: Russell D (ed). O'Reilly & Associates, Inc., Sebastopol, CA, USA

Chomicki J, Lobo J, Naqvi S (2003) Conflict resolution using logic programming. IEEE Trans Knowl Data Eng 15(1):244–249

Cuppens F et al (2004) A formal approach to specify and deploy a network security policy. In: Formal aspects in security and trust, Toulouse, France, pp 203–218

Cuppens F, Cuppens-Boulahia N, Garcia-Alfaro J (2005) Misconfiguration management of network security components. In: Proceedings of the 7th international Symposium on System and Information Security(SSI 2005), Sao Paulo, 1–10 Nov 2005

Damianou N, Dulay N, Lupu E, Sloman M (2001) The ponder policy specification language. In: International workshop, policies for distributed systems and neworks (Policy 2001), LNCS 1995. Springer, Bristol, pp 18–39, 29–31 Jan 2001

Dantsin E, Eiter T, Gottlob G, Voronkov A (2001) Complexity and expressive power of logic programming. In: ACM computing surveys, vol. 33, No. 3, pp 374–425, Sept 2001, first presented at the 12th annual IEEE conference on computational complexity (CCC'97), Ulm, 1997

Denecker M, Kakas A (2002) Abduction in logic programming. In: Kakas A and Sadri F (eds) Computational logic: logic programming and beyond, essays in honour of Robert A. Kowalski LNCS 2407, Part I, pp 402–436. Springer, Berlin

Desmet L, Piessens F, Joosen W, Verboeten P (2006) Bridging the gap between web application firewalls and Web applications. In: FMSE'06, Alexandria, pp 67–77, 3 Nov 2006

Dictionary of Computing (2004) A dictionary of computing. Oxford University Press, 2004. Oxford Reference Online, Oxford University Press, http://www.oxfordreference.com/

El Kalam AA, El Baida R, Balbiani P, Benferhat S, Cuppens F, Deswarte Y, Miège A, Saurel C, Trouessin G (2003) Organization based access control. In: Proceedings of the fourth international workshop on policies for distributed systems and networks (POLICY'03), Lake Como, 4–6 June 2003

Eppstein D, Muthukrishnan S (2001) Internet packet filter management and rectangle Geometry. In: Proceedings of the 12th annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2001), Washington DC, pp 827–835, 7–9 Jan 2001

Eronen P, Zitting J (2001) An expert system for analyzing firewall rules. In Proceedings of the sixth Nordic Workshop on Secure IT-System (Nonlsec 2001), Lyngby, pp 100–107, 1–2 Nov 2001

Fu Z, Wu F, Huang H, Lob K, Gong F, Baldine I, Xu C (2001) IPSec/VPN security policy: correctness, conflict detection and resolution. In: Proceedings of policy'2001 workshop, Bristol, pp 39–56, 29–31 Jan 2001

Garlik (2007) UK Cybercrime Report. https://www.garlik.com/press/Garlik%20Cybercrime%20Report.pdf

Gordon L, Loeb M, Lucyshyn M, Richardson R (2006) CSI/FBI computer crime and security survey. Computer Security Institute publications, New York, USA

Gouda MG, Liu XYA (2004) Firewall design: consistency, completeness and compactness. In: IEEE International Conference on Distributed Computing Systems(ICDCS) 24. Tokyo, 24–26 March 2004

Gupta P, Mc Keown N (2001) Algorithms for packet classification. IEEE Netw 15(2):24–32

Guster D, Hall C (2001) A firewall configuration strategy for the protection of computer networked labs in a college setting. J Comput Sci Coll 17(1):187–193

Guttman J (1997) Filtering postures: local enforcement for global policies. In: Proceedings of the 1997 IEEE Symposium on Security and Privacy, California, pp 120–129, 4–7 May 1997

Guttman J, Herzog A (2003) Rigorous automated network security management. Technical report, MITRE Corp., 15 Aug. 2003. Preliminary version appeared in Proceedings VERIFY 2002, Copenhagen, 25–26 July 2006

Hamed H, Al-Shaer E (2006) Dynamic rule-ordering optimization for high-speed firewall filtering. In: ASIACCS '06, Taipei, pp 332–342, 21–24 March 2006

Hari A, Suri S, Parulkar G (2000) Detecting and resolving packet filter conflicts. In: Proceedings of IEEE INFOCOM, Tel Aviv, pp 1203–1212, 26–27 March 2000

Hazelhurst S (1999) Algorithms for analysing firewall and router access lists. Technical Report TR-Wits-CS-1999-5, Department of Computer Science, University of the Witwatersrand, South Africa, July 1999

Hazelhurst S, Attar A, Sinnappan R (2000) Algorithms for improving the dependability of firewall and filter rule lists. In: DSN '00 Proceedings of the 2000 international conference on dependable systems and networks 2000, New York, 25–28 June 2000

Hinrichs S (1999) Policy-based management: bridging the Gap. In: Proceedings of the 15th annual computer security application conference, Phoenix, pp 209–218, 6–10 Dec 1999

Hunt C (1992) TCP/IP network administration, USA, O'Reilly and Associates, Inc

Huth M, Ryan M (2004) Logic in computer science. University Press, Cambridge

Internet Systems Consortium, Inc. (2007) ISC Domain Survey. http://www.isc.org/index.pl?/ops/ds/host-count-history.php

Konstantinou AV (2003) 'NESTOR: an architecture for network self-management and organization. http://www1.cs.columbia.edu/dcc/nestor/. Accessed 14 July 2007

Lakshman TV, Stiliadis D (1998) High-speed policy-based packet forwarding using efficient multi-dimensional range matching. SIGCOMM'98, Vancouver, pp 203–214, 31 Aug–4 Sept 1998

Lobo J, Bathia R, Naqvi S (1999) A policy description language. In: Proceedings of AAAI, 1999, presented at 16th national conference on artificial intelligence, Orlando, 18–22 July 1999

Lupu E, Sloman M (1997) Conflict analysis for management policies. In: Proceedings of IFIP/IEEE international symposium on integrated network management (IM 1997), California, pp 430–443, 12–16 May 1997

Mayer A, Wool A, Ziskind E (2006) Offline firewall analysis. Int J Inf Secur 5(3):125–144

Oppliger R (1997) Internet security: firewalls and beyond. Commun ACM 40(5):92–102

Ou X (2005) A logic-programming approach to network security analysis, a dissertation presented to the faculty of Princeton University in candidacy for the degree of Doctor of Philosophy, Nov 2005

Ou X, Govindavajhala S, Appel AW (2005) MulVAL: a logic-based network security analyzer. In: 14th USENIX security symposium, Baltimore, 1–5 Aug 2005

Qiu L, Varghese G, Suri S (2001) Fast firewall implementations for software and hardware-based routers. In: Proceedings of 9th international conference on network protocols (ICNP'2001), Toronto, 11–14 Nov 2001

Ralston A, Reilly E (1995) Encyclopedia of computer science. International Thomson Computer Press, London

Russo A, Miller R, Nuseibeh B, Kramer J (2002) An abductive approach for analysing event-based requirements specifications presented at 18th international conference on logic programming (ICLP), Copenhagen, 29 July–1 Aug 2002

Salus P (1998) Handbook of programming languages, vol IV—Functional and logic programming languages. Macmillan Technical, Indianapolis

Screen digest (2007) The broadcast and media technology business: global market value, structure and strategy to 2010 (Jan 2007), http://www.screendigest.com/reports/ext/06ext_broadmediatec/readmore/view.html

Smith RN, Bhattacharya S (1997) Firewall placement in a large network topology. In: Proceedings 6th workshop future trends distrib. comput. Tunis, pp 40–45, 29–31 Oct 1997

Son TC, Lobo J (2001) Reasoning about policies using logic programming presented at AAAI (American Association for Artificial Intelligence) spring symposium on answer set programming, Stanford University, California, 26–28 March 2001

Srinivasan V, Suri S, Varghese G (1999) Packet classification using tuple space search. In: Proceedings of ACM SIGCOM-M 1999 annual technical conference, vol 29, Cambridge, pp 135–146, 30 Aug–3 Sept 1999

Team Cymru (2007) The Team Cymru Bogon List v3.4 22 Jan 2007, http://www.cymru.com/Documents/bogon-list.html

Uribe T, Cheung S (2004) Automatic analysis of firewall and network intrusion detection system configuration. FMSE'04 Washington DC, pp 66–74, 29 Oct 2004

Verma P, Prakash A (2005) FACE: a firewall analysis and configuration engine. In: Proceedings of the 2005 symposium on applications and the internet (SAINT'05), Trento, 31 Jan–4 Feb 2005

Wies R (1994) Policies in network and systems management. Netw Syst Manage 2(1):63–83

Wool A (2001) Architecting the Lumeta firewall analyzer. In: Proceedings of the 10th USENIX security symposium, Washington DC, pp 85–97, 13–17 Aug 2001

Wool A (2004) A quantitative study of firewall configuration errors. IEEE Computer 37(6):62–67

Wool A (2004b) The use and usability of direction-based filtering in firewalls. Comput Secur 23(6):459–468

Xie G, Zhan J, Maltz DA, Zhang H, Greenberg A, Hjalmtysson G, Rexford J (2005) On static reachability analysis of IP networks. In: IEEE INFOCOM, 2005, Miami, pp 2170–2183, 13–17 Mar 2005

Yang J, Papazoglou MP (2000) Interoperation support for electronic business. Commun ACM 43(6):39–47

Yuan L, Mai J, Su Z, Chen H, Chuah CN, Mohapatra P (2006) FIREMAN: a toolkit for FIREwall modelling and ANalysis. In: Proceedings of the 2006 IEEE symposium on security and privacy, California, 21–24 May 2006