Assessing network authorization policies via reachability analysis

Assessing Network Authorization Policies via Reachability Analysis

Cataldo Basile[a], Daniele Canavese[a], Christian Pitscheider[a], Antonio Lioy[a], Fulvio Valenza[a,b,*]

[a] *Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy*
[b] *CNR-IEIIT, c.so Duca degli Abruzzi 24, Torino I-10129, Italy*

Abstract

Evaluating if a computer network only permits allowed business operations without transmitting unwanted or malicious traffic is a crucial security task. Reachability analysis – the process that evaluates allowed communications – is a tool useful not only to discover security issues but also to identify network misconfigurations. This paper presents a novel approach to quantify network reachability based on the concept of *equivalent firewall* – a fictitious device, ideally connected directly to the communicating peers and whose policy summarizes the network behaviour between them – that can be queried to derive reachability information. We build equivalent firewalls by using a mathematical model that supports a large variety of network security controls (like NAT, NAPT, tunnels and filters up to the application layer) and allows an accurate analysis. The presented approach is efficient and highly scalable, as confirmed by tests with a large corporate network as well as synthetic networks.

Keywords: network reachability; authorization policies; security policy assessment; network modelling; security assessment; vulnerability analysis; infrastructure security modelling; risk analysis and management

1. Introduction

At the dawn of computer science, computer networks were designed to freely exchange data. Nowadays, several security considerations have limited the initial enthusiasm: only data belonging to authorized communications must be exchanged. These are identified by means of a network authorization policy, which is part of a more general security policy. Moreover, some data must not be transferred in clear, for instance private or sensitive data that must be kept confidential via message or channel protection techniques. Data requiring protection is identified by means of a so called data protection policy.

The actual enforcement of the security policies is obtained by configuring security controls, e.g. firewalls and Virtual Private Networks (VPN) gateways, placed in proper points of the network topology. Today's networked systems are hard to configure for the administrators, even for skilled ones, because they have to setup different types of security controls.

As a consequence, this scenario introduces several issues. It may cause service interruption, malfunctioning, or disruption, when allowed communications are not properly authorized. Furthermore, it can expose sensitive portions of the network or open the door for various attacks, when unwanted communications are not properly forbidden. A secure and correctly working network is a mandatory requirement in several areas, but it is crucial in the context of dams, electrical plants and, in general, in infrastructures where huge money and human lives are at stake.

In this paper we are interested in using *reachability analysis*[1] to assess if the network authorization policy is correctly enforced. This process evaluates the allowed communications, that is, the kind of packets that can travel

*Corresponding author

Email addresses: cataldo.basile@polito.it (Cataldo Basile), daniele.canavese@polito.it (Daniele Canavese), christian.pitscheider@polito.it (Christian Pitscheider), antonio.lioy@polito.it (Antonio Lioy), fulvio.valenza@ieiit.cnr.it (Fulvio Valenza)

[1] In literature, the term *reachability* is also used in the context of computer networks to refer to connectivity issues related to routing problems, and is thus limited to IP connectivity. In this paper we are interested in verifying the reachability on all the network layers with security analysis purposes, where routing is only a facet of a bigger problem.

from one node to another. Network reachability is particularly important in critical systems for avoiding catastrophic failures and to prevent attacks that might lead to human losses.

Quantifying reachability is, however, no easy task. There are many security and network controls that may drop, alter, or forward packets along specific paths. Together with the network endpoints (e.g. workstations and servers, which merely send and receive packets) there are many classes of controls that affect reachability: *filtering controls* permit or block specific traffic, *routing controls* implement a packet forwarding policy, while *packet transformation devices* first modify then forward the packets. Transformation devices include Network Address Translation (NAT) or Network Address and Port Translation (NAPT) controls, that alter the packet IP addresses and ports according to a policy, and IP tunnelling, that encapsulates and de-encapsulates packets, usually after encrypting them. Indeed, reachability analysis can be applied to different scenarios to cope with different security related tasks. As anticipated, the most obvious application is the *validation of a network authorization policy*, as the reachability analysis can be used to detect permitted and unwanted connections between hosts (i.e. clients and servers). Moreover, it can be used to query if services are exposed to new vulnerabilities or common attack vectors. Reachability analysis can also be used to *assist in network design*: by working on an abstract representation of the network prior to its deployment, it permits the anticipation of potential issues and aids in the development of more secure and resilient infrastructures. Furthermore, reachability analysis may support administrators when performing *change impact analysis*; by inserting the desired changes in the network description and executing reachability queries, it is possible to foresee the effects of maintenance and update activities.

Reachability can be computed by directly probing an existing network (*online analysis*) or performing queries on an abstract representation of the network (*offline analysis*). Online analysis is the most used in practice, but offline analysis has many advantages and more applications. Offline analysis is an important modeling tool that can be used for the study and analysis of security and dependability of both normal systems and critical infrastructures. The main advantage of offline analysis is that it does not require physical access to the target network, since it relies on its model. Furthermore, since it has insight on the configurations it can evaluate reachability issues on a very detailed level and therefore produce a very comprehensive report. Offline analysis can also be applied during the network design phase and preempt reachability problems. Drawbacks arise if model on which the analysis is performed does not capture the actual complexity and implementation details of real networks, resulting in potentially inaccurate reports. However, this limitation does not apply to the design phase.

In the literature, other works dealt with offline reachability analysis, which are discussed in Section 7. Four major limitations are evident in the state-of-the-art. First, several approaches only take into account a subset of the security controls usually displaced in the network. For instance, Xie et al. [1], Bandhakavi et al. [2], or Sveda et al. [3] provide limited or no support for transformation controls. This limitation seriously affects the applicability to real cases and the analysis accuracy. Second, reachability analysis models that support a broader range of security controls [4] require a long pre-computation time that makes them unusable on networks of reasonable size with frequent changes or for evaluating alternative designs.

Third, reachability analysis models that support various security controls and have good performance (such as in [5]) are unable to perform complex queries concerning multiple hosts and aggregate results, being only able to answer point to point communications queries. Forth, none of the existing approaches supports application layer filters nor end-system firewalls.

We present in this paper a new approach to quantify network reachability that overcomes the major limitations of current offline analysis approaches.

This new model can deal with a large variety of security controls such as routers, stateless and stateful firewalls, end-system firewalls, NAT/NAPT, and tunnelling devices. Moreover, we also support application layer filters, even if we have tested it only with HTTP, FTP, and DNS. Additionally, our model also comes with improvements on query flexibility and expressiveness. We provide an extension of the Structured Reachability Query Language (SRQL) format [4], to support a larger set of options for stateful analysis. The innovation of our approach relies on the concept of *equivalent firewall*, a fictitious device, (ideally) directly connected to the reachability query endpoints, whose policy summarizes the network behaviour between two endpoints. Equivalent firewall policies deal with a limited portion of the network (the part that involves the query endpoints), thus they are small and, when they deal with entire subnets or bigger portions of the network, can be reused for several queries. As a consequence, equivalent firewall policies can be queried very efficiently to derive specific reachability information. Moreover, the equivalent firewall policy can also be globally inspected to gain a holistic view of allowed and denied communications.

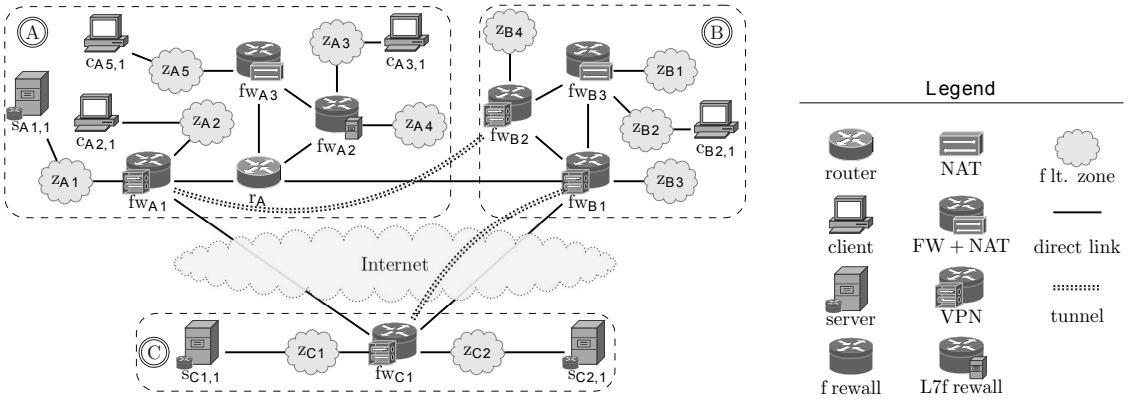


Figure 1: The reference network.

A preliminary version of this work was presented in a conference paper [6]. The main improvements over this initial version are the support for tunnelling devices and end-system firewalls, the definition of more precise queries thanks to a richer set of condition types (covering also application layer filters), and more complex Boolean expressions. Furthermore, this paper presents the theoretical foundations proving that the model is flexible and can be extended with new protocols and additional future security controls.

This paper is structured as follows. Section 2 introduces a simple but complete reference example used through the document to present our approach. Section 3 presents our approach to offline analysis, the supported queries and the improvements over the state-of-the-art. Section 4 introduces our formal model for representing networks, policies and querying reachability. Section 5 discusses how to create an equivalent firewall with several kind of devices. Section 6 presents the complexity and correcteness analysis of our model together with an analysis of its performance in a prototype implementation and in Section 7 we compare our solution to other state-of-the-art reachability analysis techniques. Finally, Section 8 draws conclusions and presents future work plans.

2. Reference Example

To make our discussion concrete, we will refer our examples to the scenario in Figure 1 that shows three networks [2]. A is a company network providing some services to an external office network B. Both A and B outsource some of their services to the network C. The three networks are connected via the Internet.

The elements depicted in the figure are:

- *firewalls*, enforce a network access policy and many of them also provide application layer filtering($\mathrm{fw_{A2}}$)[3], NAT/NAPT ($\mathrm{fw_{A3}}$ and $\mathrm{fw_{B3}}$), and tunnelling capabilities ($\mathrm{fw_{A1}}$, $\mathrm{fw_{C1}}$, $\mathrm{fw_{B1}}$ and $\mathrm{fw_{B2}}$);
- *filtering zones*, group elements that can reach each other without passing through a firewall. They are usually subnets (as assumed in this paper) or union of subnets. They play an important role in our model as within them it is impossible to apply any policy (like dropping or ciphering packets).
- *clients and servers*, are the endpoints and they are shown as connected to the filtering zones they are part of; all servers are also protected with an end-system firewall.

Networks A, B, and C have been designed hierarchically[4], however, for resilience purposes, several redundant physical connections have been added. In the figure, double dotted lines represent secure tunnels, while solid edges depict end-to-end secure channels.

[2] The reference example network presented here was not built for a typical real-world scenario. Our purpose was to design the smallest network able to allow us to present all the interesting query cases we can deal with in order to define our reachability model. For instance, all the NAT rules would be practically unnecessary.

[3] More precisely, we assume that $\mathrm{fw_{A2}}$ is able to filter HTTP connections.

[4] `http://docwiki.cisco.com/wiki/Internetwork_Design_Guide`

To avoid long tables (no one wants to read), we use formulas for the addressing scheme:

- zones z_{Ax}, z_{Bx}, z_{Cx} are respectively assigned to the IP ranges 10.1.x.0/24, 10.2.x.0/24 and 10.3.x.0/24;
- endpoints $e_{Ax,h}$ (either client or server) are assigned to 10.1.x.h, analogously $e_{Bx,h} \rightarrow$ 10.2.x.h and $e_{Cx,h} \rightarrow$ 10.3.x.h;
- prior to NAT, the zones z_{B1}, z_{B2} and z_{A5} are associated to 192.168.1.0/24, 192.168.2.0/24 and 192.168.5.0/24 respectively;
- the addresses 192.168.5.$\{3, 7, 17, 41\}$ are NATted 1:1 to 10.1.5.$\{3, 7, 17, 41\}$ (like it sometimes happens for public servers) while the rest of the addresses are translated many-to-one to 10.1.5.1.

Finally, we will denote as $fw_{e1 \rightarrow e2}$ the equivalent firewall built to answer a reachability query to assess communications between the endpoints $e1$ and e_2.

3. Our approach: the equivalent firewall

We defined a model to perform offline reachability analysis. Offline reachability analysis is quantified by executing specific queries, which ask if (a subset of) the traffic between a source and a destination is allowed or denied. Queries are formulated in order to identify violations to the authorization policy or to check the expected network behaviour.

The equivalent firewall is the key element of our model. Theoretically, an equivalent firewall is a fictitious security control used to summarize all the decisions and transformations that the packets undergo when flowing from a source to a destination. The equivalent firewall behaviour (i.e. its decisions) is defined by means of a filtering policy that summarizes the actual behaviour of all the security controls in the portion of the network under analysis.

Practically, a reachability query is translated to a single firewall query executed on the equivalent firewall policy[5]. This approach has a main advantage, that is, by analysing a single firewall policy it is possible to black-box analyse the behaviour of the network between the endpoints, ignoring its internal complexity. This simplification does not sacrifice diagnostic power, as when creating the equivalent firewall policy we preserve references to rules in the original controls so that it is possible to identify the cause of a reachability problem. The unavoidable drawback is that equivalent firewalls need to be built before executing queries. For instance, the equivalent firewall $fw_{B2,1 \rightarrow A1,1}$ (see Figure 1) is built to answer the query asking for the allowed communications between $c_{B2,1}$ and $s_{A1,1}$. The $fw_{B2,1 \rightarrow A1,1}$ policy considers the effects on packets of filtering, NAT, and routing decisions performed at fw_{B3}, then filtering, channel protection and routing decisions at fw_{B1}, routing decision at r_A, and finally, filtering and channel protection decisions at fw_{A1}. Thus, $fw_{B2,1 \rightarrow A1,1}$ is queried to know if the communication is allowed, denied, or to determine the subset of the traffic that is actually allowed.

Building equivalent firewalls is not computationally expensive as the process considers only the rules that match source and destination of a query. Therefore, the number of involved rules in the equivalent firewall computation is small and this guarantees good performance.

Moreover, equivalent firewalls can be reused for queries involving the same (or a subset of the) source and destination entities, thus leading to a further optimization. All packets from nodes in the same filtering zone as the query source (in the previous example z_{B2}) directed to nodes in the same filtering zone as the query destination (that is, z_{A1}) will encounter the same security controls. Hence, it is convenient to build the equivalent firewall between the source and destination zones, as all queries between elements in these zones can be executed on it. We name these equivalent firewalls *zone-to-zone equivalent firewalls*. Filtering zones are very important for our model: they are the maximal subset of the network that can share an equivalent firewall and hence the optimal choice for reusability. If the query source or destination span more than one filtering zone, the corresponding *multi-zone equivalent firewall* is obtained as the union of zone-to-zone equivalent firewalls between a source and destination zone. The multi-zone equivalent firewall is obtained by simply merging the individual equivalent firewalls. In fact, rules in different zone-to-zone

[5]By abuse of notation, we will sometimes refer to the equivalent firewall to indicate its policy. An equivalent firewall is a black box security control, connected to the endpoints, whose behaviour is described by the policy. That is, philosophically, a policy is the intensional definition of an equivalent firewall.

equivalent firewalls are non-overlapping and can be directly merged (regardless of the routing paths), because zone IP addresses are non-overlapping. As an example, a query about which nodes the host $c_{B2,1}$ can reach in the range $10.1.1.0/21$ requires the calculation of four equivalent firewalls $fw_{B2\rightarrow A1}$, $fw_{B2\rightarrow A2}$, $fw_{B2\rightarrow A3}$, and $fw_{B2\rightarrow A4}$. As it will be clear later, when the model will be presented, the rules from the four equivalent firewalls can be merged while preserving just the relative priorities among the rules at each firewall.

3.1. Reachability Model

Having presented the use of equivalent firewalls, we introduce now the process for constructing them, that is, the underlying network and policy models. Networks are modelled as non-simple graphs[6] and standard algorithms are used to find paths from source to destinations[7]. Each network node may be associated to one or more capabilities, used to indicate the type of security control it implements. Currently, we support filtering at all layers[8], routing, VPN tunnelling and NAT/NAPT capabilities.

Firewall policies are described by means of the geometric model introduced by Basile et al. in [7] and sketched in Section 4. In this model, rules and queries are hyper-rectangles (or union of hyper-rectangles) and policies are obtained as superposition of hyper-rectangles. The superposition is regulated by a resolution strategy, like the First Matching Rule (FMR) strategy that uses priorities to decide which rule prevails in the "overlapping areas".

To support reachability analysis, we extended the geometric model in several ways. First, we added transformation policies and rules. Transformation rules are modelled geometrically via two hyper-rectangles: the first one defines the domain of packets to transform (*transformation input clauses*) and the second one defines the domain to which packets matching the transformation input clauses will be actually transformed (*transformation output clauses*). However the most complex extension concerns the ability to compose policies: the extended model permits the composition of firewall policies arranged in a network topology. The composition is performed by manipulating hyper-rectangles of rules from different policies with ad-hoc functions that model how to merge (1) the policies of two or more cascaded firewalls (serially connected) and (2) transformation policies with filtering policies. Hyper-rectangles of filtering rule that intersect some transformation output clauses are "translated or stretched" to let the equivalent apply to packets being transformed the actions they would have applied to the transformed ones (see Figure 2 for a graphical hint).

Routing information is used to determine the paths from source to destination. Therefore, it only serves to identify the security controls to compose when building equivalent firewalls. Thus, supporting routing has not required any modification to the geometric model. However, routing affects the analysis. We support two types of reachability analysis based on static and dynamic routing information. With static routing information, the static routes (listed in routing tables) are used to determine the only path from source to destination. Equivalent firewalls are generated accordingly. On the other hand, with dynamic routing, the paths from source to destination may change over time to adapt to topology changes or network failures. Therefore, our dynamic routing analysis does not take into account routing information and it simply computes all acyclic paths from source to destination. In this case, the equivalent firewall considers all paths. Moreover, for compatibility with SRQL, we allow the specification of a path in the query, that is, instantaneous queries. In that case, the equivalent firewall is computed based on the path information, which can be different from the one implied by static routes.

When considering multiple paths, as in the dynamic routing analysis, the traffic allowed by two alternative paths may be different. Therefore, we use different strategies to generate the result: 'lower bound', 'upper bound' and 'highlight path anomalies'. The lower and upper bound schemes are defined in [4], while the highlight path anomalies one is introduced here for the first time:

- *lower bound* reports only the traffic allowed by all paths, i.e. permitted by all equivalent firewalls computed on the paths;
- *upper bound* reports all the traffic allowed by at least one path, that is, there is at least one equivalent firewall permitting this traffic;

[6] A simple graph does not contain loops or multiple edges.
[7] Liu [4] describes the fundamentals of graph theory for reachability analysis purposes.
[8] The current implementation only supports HTTP, FTP and DNS, but our model can be easily extended to support other protocols.

	priority	Source IP	PortS	Dest IP	PortD	Proto	Protocol States	Action
r_1	1	any	any	any	any	any	ESTABLISHED,RELATED	ALLOW
r_2	2	10.1.2.7	any	10.1.1.9	any	any	any	DENY
r_3	3	10.1.2.7	any	10.1.1.19	any	any	any	DENY
r_4	4	10.1.2.0/24	>1023	10.1.1.0/24	80	TCP	NEW	ALLOW
r_5	5	10.2.2.1	>1023	10.1.1.1	80	TCP	NEW	ALLOW
...								
	∞	*	*	*	*	*	*	DENY

Table 1: An excerpt of the filtering rules in $\mathrm{fw}_{\mathrm{A1}}$.

- *highlight path anomalies*, when combining the equivalent firewalls different paths which enforce different actions are highlighted by means of the Unspecified action. Finding in the equivalent firewall a Unspecified action explicitly indicates that the action depends upon the selected path. This is our suggested strategy as it highlights *path inconsistencies*.

Depending on the type of unidirectional communications between a source and a destination, our model supports the possibility to perform three types of queries:

- *stateless communication*, evaluates the possibility for a packet sent by the source to reach the destination;
- *stateful communication*, evaluates the possibility to send packets to and receive answers from the destination (which cannot send unsolicited/unrelated packets to the source);
- *multiple stateful communication*, evaluates the possibility for the source to establish more than one simultaneous connection with the destination. Stateful firewalls may limit the number of connections from the same IP.

Bidirectional communications must be evaluated with two separate queries.

3.2. Reaction to changes

As all the offline analysis models, changes to the network and its policies may trigger the re-computation of the equivalent firewalls. Structural changes to a network may or may not have an impact on our model. For instance, adding/removing a node to a zone does not require the re-computation of the zone-to-zone equivalent firewalls. It only requires to check the proper zone-to-zone firewall to use when answering a query. Changes to the communication backbones or modifications that influence at least an entire zone (such as moving a filtering control), only require a partial rebuild of the model, since only the equivalent firewalls related to the modified filtering zones need to be recomputed. The time spent for recomputing the equivalent firewalls can be mitigated by caching the intermediate artifacts obtained when generating them, thus avoiding a complete rebuild of the model. Fortunately, these re-computations are infrequent since, in critical infrastructures scenarios (e.g. SCADA), massive network updates are quite rare. Changes in the security control policies require, in the general case, the re-computation of our model. Also in this case, only a small subset of the equivalent firewalls needs to be invalidated, that is, the equivalent firewalls whose computation required the use of the changed policy. The caching mechanism can also adopted here to increase the performance.

3.3. Query format

An essential aspect of reachability analysis is the interrogation phase, which is usually performed via a set of reachability queries. In this scenario, we defined a query language (an extension of SRQL [8]) as follows:

```
reachability_type 𝒯
connection_type 𝒪
fixed_path 𝒫
equivalent_firewall_method ℰ
select ℱ
where ℬ ∧ (action = ⟨dec⟩)
```

Where:

$\mathcal{T}$ can be I for instantaneous queries, LB for lower-bound, UB for upper-bound, H to use the highlight path anomalies strategy, and S to force the use of static routing information (default is H).

$\mathcal{O}$ can be either SF for stateful analysis, SL for stateless analysis, or MSF for testing multiple stateful connections.

$\mathcal{P}$ is a path expressed as a list of network nodes from source to destination (meaningful only if $\mathcal{T} = I$).

$\mathcal{E}$ defines the equivalent firewall generation strategy, Z for computing zone-to-zone equivalent firewalls, or Q for query-specific equivalent firewalls (default is Z).

$\mathcal{F}$ is the set of packet fields S_i and our model supports the following ones: source (*S*) and destination (*D*) IP address, source (*SP*) and destination (*DP*) port, protocol type (*PT*), protocol state (*PS*), and application protocol (*AP*). Additionally, we support connection limit (*CL*), used to define the maximum number of connections from the same source IP, time (*T*) that determines rule validity period, and HTTP-specific filtering fields, such as HTTP_req_method (*QM*), HTTP_req_header (*QH*), and HTTP_rep_header (*PH*). We also support hit-limit (*L*), and hit-limit-burst (*LB*) used to specify conditions on rule hits, i.e. the number of times that a given rule is applied in a given time period. Finally, we allow specification of conditions on URL domains (*UD*), URL paths (*UP*), URLs (*U*), and browser (*B*) that use regular expressions and are supported in the geometric model [6].

$\mathcal{B}$ is any Boolean expression of conditions on the previous fields, which includes as subcase the SRQL expression:

$$(F_1 \in S_1) \wedge \cdots \wedge (F_k \in S_k) \wedge (\text{action} = \langle \text{dec} \rangle)$$

Reachability queries can be classified according to the result they return. Some queries only admit a unique answer, an unconditional allow or deny, while others do not. For instance, given the policy in Table 1, a query asking if the hosts in 10.1.2.0/24 are allowed to reach the service at 10.1.1.8:80/TCP will return that the traffic is allowed only from ports greater than 1023. To deal with these cases we added a third result type, *partly*, indicating that the communication is allowed only for some packets. Therefore, queries returning "partly" need to report the set of allowed or denied connections, which we named the *query result domain*.

Another important aspect of a query result is its *accuracy*, a feature that has been ignored in previous works. The main factors influencing the accuracy are the type of security controls (e.g. firewall, NAT and tunnelling devices) and their capabilities, the protocol fields on which they can pose conditions and the actions or the transformation that they can enforce. Therefore, the results may not be reliable when a path contains an unsupported control or a policy with unsupported condition types. The accuracy can be evaluated only after its execution. If the query result domain is not tautological in the domain of an unsupported condition (that is, the action varies according to an unsupported field), then the analysis is inaccurate. For instance, if communication with a web server is allowed only with safe HTTP methods, a model that does not consider this field could return that the communication is unconditionally permitted, which is incorrect. Therefore, we introduce the "accuracy" concept to flag possible errors in our analysis. In this way, we accept a complete description of the target network even when it contains items not supported by our model, rather than generating a false sense of precision by limiting the input to incomplete descriptions containing only the supported elements (as most models do). Accuracy, in practice, is a self assessment of the reachability model.

In our approach, the query result consists of four parts:

- *answer*, may be Allow, if all the traffic implied by the query is allowed, Deny, if all the traffic implied by the query is blocked, or Partly, if only a subset of the traffic implied by the query is allowed;
- *query result domain*, the set of the allowed traffic, expressed as a set of rules from equivalent firewall policy. This field is only used when answer is Partly (see also Section 4.4);
- *stateful domain*, the set of the allowed stateful answers, that is, the communications from the destination to the source that are allowed by some stateful rules as replies to allowed communications (used only for stateful queries when answer is Partly, see Section 5.3);
- *accuracy*, reports if any of the security controls in the path is not supported by the model or, if all the controls are supported, if the query result involves any unsupported field (see Section 5.5).

The computed result domain and stateful domain include conditions on all fields to convey full information. The user can select the fields he is interested in at query time through the `select` clause. However, result domains can be manipulated (e.g. to inspect fields where conditions are non-tautological) or simply viewed, as they are normal firewall policies.

3.4. Practical use of our model

The proposed algorithm requires three main inputs:

- the network topology, which must include all the network nodes, the security controls (with the explicit indication of their capability), and their interconnections;
- the policies of all the security controls and the routing tables of all the routing devices;
- the queries to process.

The equivalent firewalls can be calculated on start-up or on demand. In the former case, the initialization phase, needed to compute all the equivalent firewalls, takes longer, but all the reachability queries are then instantaneous. In the latter case, a database with all the pre-computed equivalent firewalls is maintained. If the equivalent firewall needed to answer the query is not already available, it is calculated and stored to be reused when required. Equivalent firewalls corresponding to multi-path analysis are not computed on start-up. When a multi-path query is required, the equivalent firewalls of all the involved paths are retrieved (or computed if not yet available), then combined according to the selected strategy (lower-bound, upper-bound, highlight path anomalies), and the result stored.

3.5. Research issues and contributions

This section lists a set of practical scenarios that we addressed in the definition of our reachability model.

3.5.1. Modelling firewalls and their composition

The first research issue appears if we want to know if the host $c_{A2,1}$ or the host $c_{A3,1}$ can reach the company web server available at $s_{A1,1}$ on port 80. Since $c_{A2,1}$ is separated from $s_{A1,1}$ by fw_{A1}, this query is reduced to a single firewall query. Thus we implemented in the geometric model, that already defines firewall policies, what is a query, the answer and the query domains (section 4.4). On the other hand, communications between $c_{A3,1}$ and $s_{A1,1}$ cross two firewalls, fw_{A2} and fw_{A1}. Therefore, we model how rules in different firewalls interact and are merged into a single equivalent firewall policy, starting from the trivial consideration that a communication is permitted only if all the firewalls in the path allow it (Section 5.1).

3.5.2. End-system firewalls

An end-system firewall installed on a host or a server influences the reachability properties of the network. Let's suppose, for example, that the server $s_{A1,1}$ has installed an end-system firewall blocking all the traffic arriving from the zone Z_{B1}. In this case the equivalent firewall must include this filtering policy, so that the reachability analysis is performed correctly. Therefore, the proposed mathematical model incorporates, when computing a zone-to-zone equivalent firewall, the filtering policies at the end-systems (Section 4.2). All the end-system firewalls are modelled as additional "virtual" devices, serially connected to zone-to-zone equivalent firewall, which process all traffic between the query endpoints.

3.5.3. Basic VPN

Another issue appears when we want to know if hosts in z_{B4} can reach the company web server $s_{A1,1}$. In this case, all the network traffic from $\mathrm{z_{A1}}$ to $\mathrm{z_{B4}}$ is encapsulated into a tunnel established at fw_{B2} and terminated at fw_{A1}. This tunnel crosses fw_{B1} and r_A that apply their rules to the encapsulated packets. Since the equivalent firewall directly connects source to destination, it "cannot see" the encapsulated packets that logically pertain to the tunnel. Therefore, equivalent firewalls have been modelled to include rules that enforce to the original packets the actions the downstream firewalls apply to encapsulated packets. Therefore, our mathematical model describes how tunneling rules change policies of downstream firewalls (i.e. fw_{B1} and fw_{A1}) to produce the equivalent firewall (Section 5.2). The equivalent firewall obtained after including the transformation can be later composed with other firewalls in the path, as in the previous section. Furthermore, we also defined how the secure channel is terminated at the second endpoint by means of the 'inverse' operations that annihilates the effect of first transformation (Section 5.2.3).

3.5.4. Handling NAT/NAPT

Modelling NAT/NAPT functionality is needed to know if the client $c_{B2,1}$ can reach the web service offered by $s_{A1,1}$. In this case we have a single transformation, applied to all traffic from $\mathrm{z_{B2}}$ to $\mathrm{z_{A1}}$, fw_{B3} applies the transformation to its traffic thus fw_{B1} contains rules which match the transformed packets. Even in this case, the equivalent firewall cannot see transformed packets and must enforce the policy on the original ones. Therefore, we also model NAT/NAPT rules (Section 4.5) and how these rules change downstream firewall policies (Section 5.2). After having modified fw_{B1} policy to match packets before the transformation, the equivalent firewall is obtained by composing it with other firewall policies in the path.

3.5.5. Multiple transformations

The query asking if $c_{B2,1}$ is allowed to reach $s_{C1,1}$ introduces the last issue: how to manage multiple transformations. In this example all traffic from $\mathrm{c_{B1,1}}$ is NATted at fw_{B3} then inserted into a tunnel at fw_{B1}. In this case, our approach recursively 'eliminates' all the transformation controls by adding specific rules to all the downstream firewalls (Section 5.2.2).

4. Formal policy model

We summarize here the basic concepts of the geometric model presented in [7], then we use it to model transformation policies and queries for reachability analysis purposes.

4.1. Modelling filtering devices

In our geometric model a firewall policy is a function represented as a four-tuple $(R, \mathfrak{R}, E, a_d)$, where:

- $R = \{r_i\}_i$, $i \in [1, n]$ is the rule set.
- $\mathfrak{R} : 2^R \rightarrow \mathcal{A}$ is the resolution function used to decide the action for packets matching more than one rule. An example of resolution function is the already presented FMR that, in case of multiple matching rules, selects the action from the rule at highest priority. The firewall action set includes "allow" and "deny" actions and will be indicated as $\mathcal{A} = \{a, d\}$.
- $E = \{E_1, E_2, \ldots\}$ is the set of external data associated to the rules. External data are not part of the rules, nevertheless they are used to take decisions. The association is done using a set of external data functions $\varepsilon_k : R \rightarrow E_k$. FMR uses priorities as external data. In the rest of the paper, the function that associates rules to priorities will be denoted by π.
- a_d is the default action, applied when a packet matches no rules.

Rules $r_i = (c_i, a_i)$ are composed of a condition clause c_i and an action $a_i \in \mathcal{A}$. The conditions clause is defined as:

$$c_i = s_{i1} \times \cdots \times s_{i1} \subseteq S_{i1} \times \cdots \times S_{im} = \mathcal{S}$$

where each condition s_{ij} is a subset of a *selector* S_{ij}. Examples of selectors are the protocol fields mentioned in Section 3.3, such as IP source address and port number. The set $\mathcal{S} = S_{i1} \times \cdots \times S_{im}$ is named *decision space*. The condition clause c_i is thus a hyper-rectangle (or the union of hyper-rectangles) in $\mathcal{S}$.

Selectors are categorized as follows:

- *exact match selectors*, when conditions can be stated using only the $=$ and $\neq$ operators (e.g. *P* = TCP);
- *range-based selectors*, when conditions can be stated using $=$ and $\neq$ and inequalities (e.g. $DP > 1023$), and their sub-case, the *prefix-based selectors*, when conditions can be stated by defining a prefix then wildcards (e.g. *D* = 1.2.3.*);
- *regex-selectors*, when conditions can be stated as regular expressions, and their sub-case, when conditions are expressed as string matching conditions. Support to the regex selectors has been added into the geometric model in a previous work [9].

A condition s_{ij} is a *point* if it is composed of a single element, or a *range* if it is an ordered (compact) subset of a range-based or prefix-based selector. A condition clause is a point if all its conditions are points.

By extending $\mathfrak{R}$ with $\mathfrak{R}\{r_i\} = a_i$ and $\mathfrak{R}\{\emptyset\} = a_d$, a policy p represented as $(R, \mathfrak{R}, E, d)$ is a function $p : \mathcal{S} \to \mathcal{A}$ such that, if $x \in \mathcal{S}$, $p(x) = \mathfrak{R}\{\mathrm{match}_R(x)\}$, where $\mathrm{match}_R(x)$ is a subset of R composed of rules that match x, defined as $\mathrm{match}_R(x) = \{r = (c, a),\ r \in R \mid x \in c\}$.

Consider the policy in Table 1. The decision space is:

$$[0.0.0.0, 255.255.255.255] \times [0, 65535] \times [0.0.0.0, 255.255.255.255] \times [0, 65535] \times \{\mathrm{TCP}, \mathrm{UDP}, \ldots\} \times \{\mathrm{NEW}, \ldots\}$$

and, for instance, rule r_4 is:

$$r_4 = (([10.1.2.0, 10.1.2.255] \times [1024, 65535] \times [10.1.1.0, 10.1.1.255] \times \{80\} \times \{\mathrm{TCP}\} \times \{\mathrm{NEW}\}), a)$$

and is associated to its priority with $\pi(r_4) = 4$. Then the ruleset becomes $R = \{r_1, r_2, \ldots, r_6, \ldots\}$. The last rule, with the lowest priority, is the default action. The policy becomes $(R, \mathrm{FMR}, \mathbb{P}, d)$ where $\mathbb{P} \subset \mathbb{N}$ is the set of the used priorities.

4.2. Adding support for end-system firewalls

End-system firewalls are software applications installed on hosts and servers that filter the network traffic. Their key property is that the source or destination of all their rules is the host/server itself. Since the subjects of different end-system firewall are always different, the rules have no correlation and never intersect each other. This fact gives the opportunity to merge all end-system firewall rules of one zone into a single virtual firewall. Rules of one end-system firewall will never change the behavior of rules of another end-system firewall. For this reason, end-system firewalls are modelled as a fictitious device, named virtual firewall, placed between the border firewall of the zone and all hosts/zones.

The policy $p_{VFW} = (R_{VFW}, \mathfrak{R}_{VFW}, E_{VFW}, d_{VFW})$ of the virtual firewall has the following components:

$$R_{VFW} = R_1 \cup \cdots \cup R_n \qquad E_{VFW} = E_1 \cup \cdots \cup E_n$$
$$d_{VFW} = \texttt{ALLOW} \qquad \mathfrak{R}_{VFW} = \mathfrak{R}_1 \cup \cdots \cup \mathfrak{R}_n$$

The rule set R_{VFW} is the union of all end-system firewall rule sets. The external data E_{VFW} is the union of all external data of the end-system firewalls. The default action d_{VFW} is ALLOW because if no rule matches, the virtual firewall must be transparent. The resolution strategy $\mathfrak{R}_{VFW}$ is the superposition of the resolution strategies of all the involved end-system firewalls. $\mathfrak{R}_{VFW}$ will only operate among rules from the same end-system firewall, that is, the actual resolution strategy to apply is selected based on the involved rules.

4.3. Adding support for stateful conditions

Some firewalls allow the use of various stateful conditions to specify sophisticated policies. All stateful conditions can be easily represented by the geometric model with ad-hoc selectors. Stateful conditions can be used to allow traffic belonging to already "established" TCP connections or "related" ones, as in Table 1 or the following iptables rule:

```
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

To model this case we define a new exact match selector that includes the RELATED, ESTABLISHED, NEW, INVALID[9] values (endowed with $=$ and $\neq$ operations). We denote with $\mathbb{S}$ the set of all the possible states.

Stateful firewalls may also enforce bandwidth control. This may be used to specify the maximum number of connections allowed to a given destination, or the maximum packet rate per destination address and per port. Additionally, it may be used to bound rule hits, i.e. to limit the packets allowed per time unit (e.g. the iptables "limit" module), as often done for ICMP packets:

```
iptables -A INPUT -p icmp -m limit --limit 10/second -j ACCEPT
```

[9] According to iptables tutorial, a packet is labelled INVALID if its state cannot be identified or it has no state. This may happen if packets do not respond to any known connection or the system is running out of memory.

In other cases, it is possible to specify the maximum number of connections that can be established between two nodes (e.g. the iptables "connlimit" module):

```
iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 20 -j REJECT
```

To model these cases, we defined three range-based selectors: *limit*, *limit-burst*, and *conn*. In particular, *conn* is the selector to watch for multiple stateful connection analysis.

4.4. Executing queries and getting results

Our query format includes options to specify the type of analysis and equivalent firewall:

`select` $S_i, S_j, \ldots$ `where` $\mathcal{B} \wedge (\text{action} = \langle \text{dec} \rangle)$

The `where` clause determines the communications to query. If we represent the Boolean expression $\mathcal{B}$ in Disjunctive Normal Form, we notice that it is the logical disjunction of AND-ed conditions like the following:

$$\mathcal{B} = (q_1^{(i)} \subseteq S_1) \wedge \cdots \wedge (q_m^{(i)} \subseteq S_m) \subseteq \mathcal{S}$$

therefore $\mathcal{B}$ is a hyper-rectangle. It is worth mentioning that if no explicit condition is specified in query condition for a given selector S_x, the condition is implicitly assumed as tautological (that is, $q_x^{(i)} = S_x$).

Thus, in the geometric model, the *query condition clause* $\mathcal{B}$ is a hyper-rectangle or the union of hyper-rectangles and each query $q = (\mathcal{B}, \text{dec})$ is equivalent to a rule whose action is dec and condition $\mathcal{B}$. For instance a query Q_1, asking for the TCP destination ports on the server $S_{A1,1}$ (10.1.1.1) reachable from the host $C_{B2,1}$ (10.2.2.1) with a stateless analysis, generates the query condition q_1:

$$q_1 = (\{10.2.2.1\} \times [0, 65535] \times \{10.1.1.1\} \times [0, 65535] \times \{\text{TCP}\} \times \mathbb{S}, a)$$

We introduce here the *query processing algorithm*. When a query is executed on a firewall whose policy is $(R, \mathfrak{R}, E, a_d)$ the condition clauses of all rules are intersected with $\mathcal{B}$ to form a ruleset composed of the restrictions of rules whose condition clause intersects $\mathcal{B}$:

$$R^{(q)} = \{r_i^q = (\mathcal{B} \cap c_i, a_i) \mid r_i \in R \wedge \mathcal{B} \cap c_i = \emptyset\}$$

The query result domain is a new policy $\left(R^{(q)}, \mathfrak{R}, E^{(q)}, a_d\right)$ where also the external data are restricted to non-discarded rules. As an example, the execution of Q_1 on fw_{A1} generates a policy where $R^{(q_1)}$ is composed by a single rule:

$$r_i^q = (\{10.2.2.1\} \times [1024, 65535] \times \{10.1.1.1\} \times \{80\} \times \{\text{TCP}\} \times \mathcal{S}, \{a\})$$

Since Q_1 only asks for destination ports, the returned result would be $D = \{80\}$.

However, only asking for destination ports does not give precise information. In fact, it is also useful to show administrators that port 80 can only be reached from registered and dynamic ports $[1024, 65535]$. Therefore, we can generalize the scenario by presenting an *algorithm for identifying influential selectors*. Selectors where the action remains the same (for the whole subset in the query condition clause) are marked as *uninfluential*.

In the last example, regardless of the source and destination IP addresses and protocol states, the answer would be "allow". On the other hand, selectors where the action changes are marked as *influential*, e.g. source port for which $[0, 1023] \rightarrow d$ and $[1024, 65535] \rightarrow a$, and destination port for which $[0, 79] \cup [81, 65535] \rightarrow d$ and $\{80\} \rightarrow a$. If there are no influential selectors then the query answer is either *Allow* or *Deny*. If there are influential selectors the result is *Partly*, even if all selectors in `select` clause are uninfluential. In fact, if at least one of the influential selectors is not included in the `select` clause, the accuracy field of the result is set to *influential selectors ignored*.

Thanks to the expressivity of our model, we extended the original SRQL syntax to allow the inclusion of more selectors $S_i, S_j, \ldots$, in the `select` clause.

Processing zone-to-zone queries is an easy task. As a first job, we extend the query condition so that the conditions on source and destination IP in $\mathcal{B}$ match not only the endpoints specified in the query but the entire IP range associated to the filtering zone. After this initial modification of the query condition, the query processing algorithm is executed with no modifications. Finally, when the IP source or destination fields in the query condition clause span more than one filtering zone, the query condition clause is split in one query condition clause for each couple of zones in the

source and destination fields. Each query condition clause is used to build an equivalent firewall. The generated equivalent firewalls are disjoint (no rules in one equivalent firewall intersect rules in other equivalent firewalls) and can be merged before executing the query.

4.5. Transformation policies

Transformation policies include actual packet transformation (e.g. NAT/NAPT) and tunnelling controls that use IP-in-IP to encapsulate packets. We will assume in this model, without loss of generality, that transformations are always performed by changing the original packet header instead of adding a new one.

Transformation policies are composed of a set of t transformation rules τ_i

$$T = \{\tau_i\}_i,\ i \in [1, t] \qquad \tau_i : \gamma_i^{(\text{in})} \mapsto \gamma_i^{(\text{out})}$$

where $\gamma_i^{(\text{in})}, \gamma_i^{(\text{out})} \subseteq \mathcal{T}$, are the transformation input and output clauses defined as:

$$\gamma_i^{(\text{in})} = \prod_{j \in I_\text{T}} \sigma_{ij}^{(\text{in})} \quad \gamma_i^{(\text{out})} = \prod_{j \in I_\text{T}} \sigma_{ij}^{(\text{out})}$$

$\sigma_{ij}^{(\text{in})} \subseteq S_j$ and $\sigma_{ij}^{(\text{out})} \subseteq S_j$ are conditions. Transformation clauses are defined over a transformation space:

$$\mathcal{T} = \prod_{j \in I_\text{T}} S_j \subseteq \mathcal{S}$$

where $I_\text{T} \subseteq [1, m]$ is the subset of the indices of the selector used to decide transformations, i.e. it is used by some transformation control in the network to analyse. We will say that S_i is in $\mathcal{T}$ if $i \in I_\text{T}$. For instance, in case of NAPT, source and destination IP addresses, source and destination ports, and protocol type are in $\mathcal{T}$ whereas tunnelling controls currently use IP source, IP destination, and protocol ID fields.

We preferred to define the transformation space from the filtering space (and we did not define it independently), because transformation rules operate on filtering rules (see Section 5.2), therefore they need to share the selectors. Consequently, a transformation policy is a piecewise function in $\mathcal{T}$ defined as:

$$\begin{array}{rcl} \eta : \mathcal{T} & \to & \mathcal{T} \\ x & \mapsto & \begin{cases} \gamma_i^{(\text{out})} & \text{if } x \cap \gamma_i^{(\text{in})} \\ & \text{and } \gamma_i^{(\text{out})} \text{ is a point} \\ \text{rand}\left(\gamma_i^{(\text{out})}\right) & \text{if } x \cap \gamma_i^{(\text{in})} \\ & \text{and } \gamma_i^{(\text{out})} \text{ is not a point} \\ x & \forall i, x \cap \gamma_i^{(\text{in})} = \emptyset \end{cases} \end{array}$$

The first case describes one-to-one transformations, that is, if a packet matches a transformation rule all values of fields in $\mathcal{T}$ in the received packet are transformed with the values in the output clause. There is no ambiguity as all the output values are fixed (i.e. $\gamma_i^{(\text{out})}$ is a point). The second case describes many-to-one or many-to-many mappings. In these cases, not all the values in the output clause are fixed (i.e. $\gamma_i^{(\text{out})}$ is not a point), therefore, the values are randomly chosen with $\text{rand}\left(\gamma_i^{(\text{out})}\right)$. For instance, for many-to-one translation the source port is randomly chosen in the dynamic ports (this case is sometimes referred as Dynamic NAT), and for many-to-many translation source port is randomly chosen in the dynamic ports and source IP is randomly chosen in the source IPs of the output clause. The last case just says that the packet is left untouched if no rule matched.

We assume that transformation policies do not contain loops or inconsistencies, that is we assume that the following properties are always true in transformation policies:

$$\begin{array}{ll} \forall i, j,\ \gamma_i^{(\text{in})} \cap \gamma_j^{(\text{in})} = \emptyset & \text{(decorrelation)} \\ \forall i, j,\ \gamma_i^{(\text{out})} \cap \gamma_j^{(\text{in})} = \emptyset & \text{(no chaining)} \end{array}$$

The "decorrelation" hypothesis excludes the possibility that two rules intersect, thus rule order becomes uninfluential and no priorities are needed. The "no chaining" hypothesis excludes the possibility that a packet matches another rule after its transformation. This excludes the possibility of transforming a packet more than once in a single control. Both hypotheses are met in real transformation policies and avoid the unneeded complexity of defining transformation policies according to the 4-tuple formulation presented in section 4.1 for filtering policies.

5. Generating the equivalent firewall

Having modelled policies for each device category, we proceed to present the generation of equivalent firewalls.

5.1. Composing filtering policies

To model firewall compositions, it is worth considering that security controls do not directly interact, i.e. there is no information flow among two security devices concerning the decision. A downstream control just sees the effects of the decisions of the upstream ones (or it does not see, if the packet is dropped). In our case, filtering devices only allow two actions, Allow and Deny. A packet can arrive at its destination only if all the devices in the path decide to forward it. On the other hand, if the upstream control drops a packet as a consequence of a deny rule, then the actions of the downstream ones on the same packet are irrelevant. To model this scenario, we introduce the *serial composition* as the function:

$$+ : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$$

which describes the composition of two actions when the filtering devices are arranged serially:

$$a + a = a \qquad a + d = d + a = d + d = d$$

The result is Allow only when both operands are Allow[10].

Given the policies p_1, represented as $(R_1, \mathfrak{R}_1, E_1, a_{d1})$, and p_2, represented as $(R_2, \mathfrak{R}_2, E_2, a_{d2})$, and the serial composition '+', we define the *composed policy* $p_1 + p_2$ as a policy $(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, a_{d1} + a_{d2})$, which uses, the composed resolution strategy $\mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}$ defined as:

$$\begin{array}{rccc} \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}: & 2^{R_1 \cup R_2} & \longrightarrow & \mathcal{A} \\ & S_1 \cup S_2 & \longmapsto & \mathfrak{R}_1(S_1) + \mathfrak{R}_2(S_2) \end{array}$$

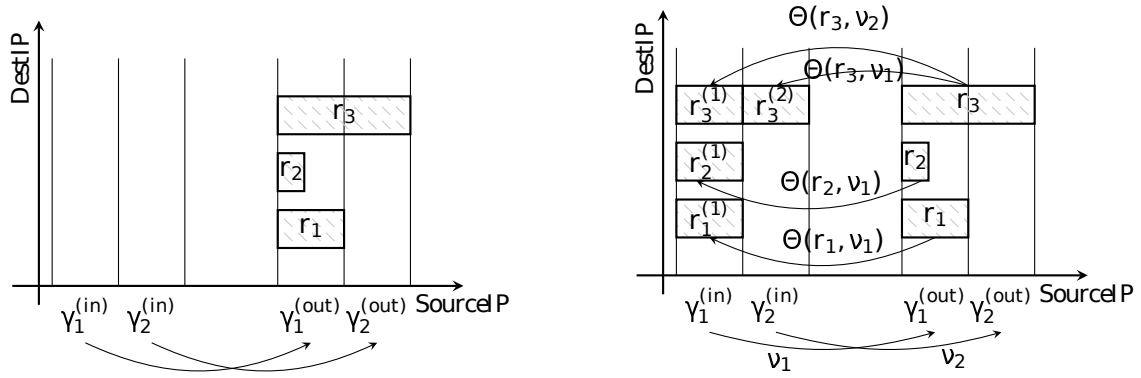
This formula also describes the composition between the default actions, in fact, when $S_1 = \emptyset$ and $S_2 = \emptyset$, we have that $\mathfrak{R}_1(S_1) + \mathfrak{R}_2(S_2) = a_{d1} + a_{d2}$. It is worth noting that $(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, a_{d1} + a_{d2})$ is a (single) policy in the geometric model and describes the behaviour of two serially connected policies. That is, it defines the equivalent firewall that can substitute the two original ones.

After having generated the equivalent firewall, this new policy can be transformed using the FMR morphism [7] or "compressed" [10]. Typically, two firewalls in the same path share many rules. For instance, fw_{B2} and fw_{B1} either share the same rule that allows z_{B4} to reach z_{A4} or fw_{B1} contains a "larger" rule that allows communication between network B and network A. In both cases, one rule can be dropped so that in most cases the obtained equivalent firewall contains about the same rule number as the original firewalls. Even better, since rules that do not concern source and destination query zones are irrelevant, equivalent firewalls frequently contain (orders of magnitude) less rules than the original firewall policies.

The approach used to compose two firewalls extends well to more than two serial firewalls as an associative operation $(p_1 + p_2) + p_3 = p_1 + (p_2 + p_3)$ represented as $(R_1 \cup R_2 \cup R_3, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2,\mathfrak{R}_3}, a_{d1} + a_{d2} + a_{d3})$.

It is worth noting that different firewalls may not be able to pose conditions on the same fields. Only five-tuple rules are supported by any firewall. For instance, stateful conditions or application layer conditions are not available in packet filters. In these cases, before actually composing the policies, we prepare a new decision space that includes all the selectors in the involved firewalls. Conditions in selectors that cannot be specified in a firewall are assumed

[10] We use the + symbol because it recalls the operation used to compute total resistance of resistors in series and because it does not extend to controls other than firewalls.



(a) A policy p with rules intersecting some $\gamma^{(\mathrm{out})}$.

(b) The policy p after the transformation.

Figure 2: Composition of transformation and filtering policies.

as tautological. This simple task, named *decision space homogenisation*, is fundamental to compose controls and execute queries and can be performed once for all security controls in the network.

It should also be noted that rules in the composed firewall are obtained as composition of the original firewall rules, thus it is possible to maintain a link from equivalent firewall rules to the original ones, which is useful for debugging purposes.

5.2. Composing transformation and filtering policies

When a transformation rule matches a packet, it transforms the values in the input clause fields into the values in the output clause fields. Let us consider a packet x transformed by the NAT policy η into a new packet $x' = \eta(x)$ which is then processed by a firewall with policy $p = (R, \mathfrak{R}, E, a_d)$. To model the composition of NAT and firewall we can add in the firewall ruleset R a new rule, at higher priority than the existing ones, to tell the firewall to apply to x the action it would apply to x'. That is, a new rule $(x, p(x'))$ is added to R. This "packet-wise" approach is unfeasible as too many rules would be added to the original policy to support all the possible packets in the query condition clause. Therefore we take a different approach and proceed rule-wise. We take all the rules $r = (c, a)$ in R whose condition clause $c = s_1 \times s_2 \times \cdots \times s_m$ intersects some output clause $\gamma_i^{(\mathrm{out})}$ of $\tau_i = (\gamma_i^{(\mathrm{in})}, \gamma_i^{(\mathrm{out})})$ and create from each of them a new rule $r^{\mathrm{T}} = \left(c^{\mathrm{T}}, a\right)$ where $c^{\mathrm{T}} = s_1^{\mathrm{T}} \times s_2^{\mathrm{T}} \times \cdots \times s_m^{\mathrm{T}}$ and each $s_i^{\mathrm{T}} \subseteq S_i$ is obtained as follows:

- if S_i is in $\mathcal{T}$ then $s_i^{\mathrm{T}} = \sigma_i^{(\mathrm{in})} \subseteq S_i$ from $\gamma_i^{(\mathrm{in})}$;
- if S_i is not in $\mathcal{T}$, then $s_i^{\mathrm{T}} = s_i$.

Figure 2 presents an example of a policy before and after application of the rule-wise transformation, which anticipates the formal model presented later in this section. Figure 2a shows the original firewall configuration with three rules $\{r_1, r_2, r_3\}$, and, on the x-axis (the source IP axis), the NAT policy, composed of two rules, one mapping the x-interval $\gamma_1^{(in)}$ into $\gamma_1^{(out)}$ and the second mapping $\gamma_2^{(in)}$ into $\gamma_2^{(out)}$.

The rule r_1 on the x-axis spans over the entire $\gamma_1^{(out)}$, r_2 partly covers $\gamma_1^{(out)}$, and r_3 spans over the union of $\gamma_1^{(out)}$ and $\gamma_2^{(out)}$. Figure 2b shows the resulting equivalent firewall.

All the three original rules need to be "moved" and stretched before being part of the equivalent firewall to model the fact that the equivalent firewall has to apply to packets before the transformations, the actions that the original firewall would have applied to the packets after the transformation.

Therefore, both r_1 and r_2 need to be duplicated and modified to fit into $\gamma_1^{(in)}$ to originate $r_1^{(1)}$ and $r_2^{(1)}$. r_3 has been first split into two rules, corresponding to subsets $\gamma_1^{(out)}$ and $\gamma_2^{(out)}$, then the single sub-rules have been moved and stretched to fit into $\gamma_1^{(in)}$ and $\gamma_2^{(in)}$, respectively, originating $r_3^{(1)}$ and $r_3^{(2)}$.

5.2.1. Formalization

Changes into condition clauses forced by transformation controls are formally described by the *selector substitution function* that substitutes all the selectors in $\mathcal{T}$ of c_i with the corresponding conditions in $\gamma^{(\text{in})}$:

$$\Sigma(c_i, \gamma^{(\text{in})}) \mapsto c'_i = \prod_{j \in [1,m]} x_{ij}$$

where $x_{ij} = s_{ij}$ if $j \notin I_{\text{T}}$ and $x_{ij} = \sigma_{ij}^{(\text{in})}$ if $j \in I_{\text{T}}$.

The following functions, working on rules $r_i = (s_{i,1} \times \cdots \times s_{i,m}, a_i)$, help in making this treatment compact:

- $\text{in}(\tau_i) = \gamma_i^{(\text{in})}$, named input clause extraction;
- $\text{out}(\tau_i) = \gamma_i^{(\text{out})}$, named output clause extraction;
- $\Psi\left(r_i\right) = \prod_{j \in I_{\text{T}}} s_{ij}$, named projection on transformation space;

where $\mathcal{C}(r_i) = c_i$. The new rules added in the downstream policy are obtained by the *rule transformation function* Θ:

$$\Theta(r_i, \tau_j) : r_i \longmapsto r_i^{(j)}$$

where $r_i^{(j)}$ is defined as:

$$r_i^{(j)} = \begin{cases} (\Sigma(c_i, \text{in}(\tau_i)), a_i) & \text{if } \Psi\left(r_i\right) \cap \text{out}(\tau_j) \neq \emptyset \\ \emptyset & \text{if } \Psi\left(r_i\right) \cap \text{out}(\tau_j) = \emptyset \end{cases}$$

Practically, a new rule $r_i^{(j)}$ is returned by the selector substitution function only if c_i intersects the transformation-output clause. If c_i does not intersects the transformation-output clause there is nothing to add to R, thus it returns the empty set. By abuse of notation, we will write $r_i = \Theta^{-1}\left(r_i^{(j)}\right)$ if $r_i^{(j)} = \Theta(r_i, c)$ for some τ_j.

The last case to cover is when a packet is transformed but, after the transformation, it does not match any rule. In this case, we must ensure that the equivalent firewall enforces the default action. Therefore, for all the transformation rules whose output clause do not intersect any rule in R we create ad-hoc rules (put at higher priority) that enforce the rules matching the corresponding input clause the default action.

Formally, if $\forall r_i, \Psi\left(r_i\right) \cap \text{out}(\tau_j) = \emptyset$ then we create the rules:

$$r^{(\tau_j)} = (\Sigma(S, in(\tau_j)), d)$$

We name $D^{(T)}$ the set of all the $r^{(\tau_j)}$. The cardinality of $D^{(T)}$ is always less than or equal to the cardinality of T. By applying $\Theta(r_i, \tau_j)$ for all the $r_i \in R$ and $\tau_j \in T$ and by adding the $D^{(T)}$, we obtain a set of rules:

$$R^{(T)} = \{\Theta(r_i, \tau_j)\}_{i,j} \cup D^{(T)}, \text{ with } i \leq m, j \leq t$$

These rules inherit the values of the external data from the rules they are derived from, i.e. the external data functions in $\{\Theta(r_i, \tau_j)\}_{i,j}$ are defined as:

$$\forall k, \varepsilon_k(r_i^{(j)}) = \varepsilon_k(r_i)$$

while for the rules in $D^{(T)}$ it is enough to suppose they have the highest priority.

By composing the transformation policy defined by T with the policy $(R, \mathfrak{R}, E, d)$, we obtain the T-modified p policy, denoted as $p^{(T)}$, which can be represented as:

$$(R \cup R^{(T)}, \mathfrak{T}, E, d)$$

The *T-modified resolution strategy* $\mathfrak{T}$ is defined as:

$$\begin{array}{ccc} \mathfrak{T} : 2^{R \cup R^{(T)}} & \to & \mathcal{A} \\ M_{\text{eq}} \subseteq R \cup R^{(T)} & \longmapsto & \begin{cases} \mathfrak{R}\left(H\right) & \text{if } M^{(T)} \neq \emptyset \\ \mathfrak{R}\left(M\right) & \text{if } M^{(T)} = \emptyset \end{cases} \end{array}$$

	priority	Source IP	PortS	Dest IP	PortD	Proto	Protocol States	Action
$r_5^{(1)}$	5	192.168.1.0/24	>1023	10.1.1.1	80	TCP	NEW	ALLOW
$r_6^{(1)}$	6	192.168.2.0/24	>1023	10.1.1.1	80	TCP	NEW	ALLOW
r_1	1	any	any	any	any	any	ESTABLISHED,RELATED	ALLOW
r_2	2	10.1.2.7	any	10.1.1.9	any	any	any	DENY
r_3	3	10.1.2.7	any	10.1.1.19	any	any	any	DENY
r_4	4	10.1.2.0/24	>1023	10.1.1.0/24	80	TCP	NEW	ALLOW
r_5	5	10.2.2.1	>1023	10.1.1.1	80	TCP	NEW	ALLOW
r_6	6	10.2.2.2	>1023	10.1.1.1	80	TCP	NEW	ALLOW
...								
	∞	*	*	*	*	*	*	D

Table 2: An excerpt of the filtering rules in $\mathrm{fw}_{\mathrm{A1}}$ after the NAT transformation.

where $M_{\mathrm{eq}} = \mathrm{match}_{R \cup R^{(T)}}(x) = M \cup M^{(T)}$, $M \subseteq R$, $M^{(T)} \subseteq R^{(T)}$, and $H = \{\Theta^{-1}\left(r_i^{(j)}\right) \mid r_i^{(j)} \in M^{(T)}\} \subseteq R$.

It is worth noting that the T-modified resolution strategies are sound. That is, they give a result for any subset of $R \cup R^{(T)}$, because they always decide by using $\mathfrak{R}$ and, if available, external data are inherited from the original rules. Moreover, this is also valid for resolution strategies requiring unique values of the external data, such as FMR priorities. In fact, after the transformation there are rules with duplicated external data values but they are never used in the same decision, as either original or transformed rules are taken into account. Table 2 presents the policy in Table 1 after the NAT transformation.

5.2.2. Multiple transformations

If, before reaching a firewall, a packet encounters first the transformation control T_1 then the T_2, we generate the (T_1, T_2)-modified p policy $p^{(T_1,T_2)}$ recursively. That is, $p^{(T_1,T_2)}$ is the T_1-modified p^{T_2} policy: $p^{(T_1,T_2)} = \left(p^{(T_1)}\right)^{(T_2)}$

In these cases, the rule set is $R \cup R^{(T_1)} \cup \left(R \cup R^{(T)}\right)^{(T_2)} = R \cup R^{(T_1)} \cup R^{(T_2)} \cup R^{(T_2,T_1)}$. The resolution strategy obtained from these two transformations works in stages. First it checks if some of the newly generated rules in $R^{(T_2,T_1)}$ apply, then it checks in $R^{(T_2)}$, then it checks if some rule from previous transformation apply ($R^{(T_1)}$), finally it checks in R. Consider the scenario in Figure 3, with two serially connected filtering devices configured to perform pre- and post-NAT. The steps to obtain the equivalent firewall are:

- compute the $T_{1,pre}$-modified p_1 (filtering) policy $p_1^{(T_{1,pre})}$ (i.e. the inner dashed rectangle in Figure 3);
- compute the $(T_{1,pre}, T_{1,post}, T_{2,pre})$-modified p_2 (filtering) policy $p_2^{(T_{1,pre},T_{1,post},T_{2,pre})}$ (i.e. the external dashed rectangle in Figure 3);
- compute the equivalent firewall p_{eq} by serially connecting $p_1^{(T_{1,pre})}$ and $p_2^{(T_{1,pre},T_{1,post},T_{2,pre})}$.

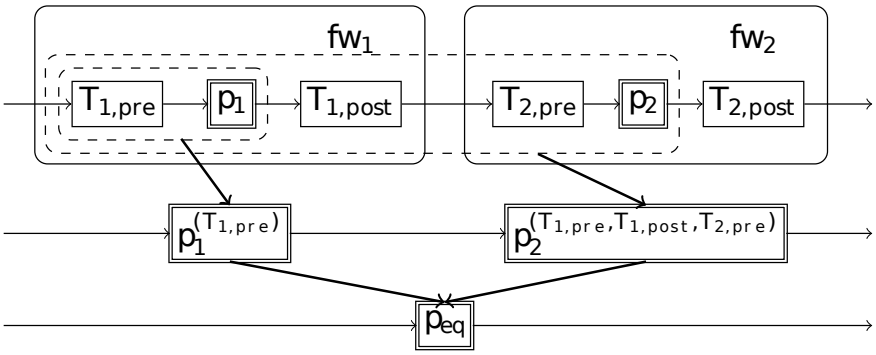


Figure 3: Composition of two cascaded firewalls with pre- and post-NAT (filtering policies are marked with a double border).

5.2.3. Inverse transformations

Another very important case needs to be considered when working with tunnels: inverse transformations. In fact, NAT/NAPT controls perform the inverse transformation for back traffic based on state information without the need of explicit rules. For tunnelling, this is not true and the inverse transformation must explicitly indicate the other endpoint policy. Even if the inverse can be considered as an independent transformation, that would evidently be a sub-optimal approach. For this reason, we pre-process transformation controls in the path to identify and eliminate inverse rules.

This scenario usually happens when a gateway with policy p_1 contains a rule $\tau_1 : \gamma^{(\text{in})} \mapsto \gamma^{(\text{out})}$ and another one with policy p_2 contains a rule $\tau_2 : \gamma^{(\text{out})} \mapsto \gamma^{(\text{in})}$. However, rule annihilation must be considered even if a subset of the input and output clauses intersect. That is, given $\gamma_1^{(\text{in})} \mapsto \gamma_1^{(\text{out})}$ and $\gamma_2^{(\text{in})} \mapsto \gamma_2^{(\text{out})}$, the annihilation happens if $\gamma_2^{(\text{in})} \subseteq \gamma_1^{(\text{out})}$. In this case, instead of eliminating both rules, we simply leave in one of the transformation controls the "biggest" rule, that is, either $\gamma_1^{(\text{in})} \setminus \gamma_2^{(\text{in})} \mapsto \gamma_1^{(\text{out})}$ or $\gamma_2^{(\text{in})} \setminus \gamma_1^{(\text{in})} \mapsto \gamma_2^{(\text{out})}$.

5.3. Processing stateful queries

In our model, executing stateful queries does not require computing a separate equivalent firewall but only changing the query condition clause. Practically, verifying that also stateful answers from the destination to source are allowed (i.e. executing a stateful query) is computed by simply performing two more operations. First, we extend the query condition to a *stateful query condition clause* whose conditions are:

- *S* = the destination address from the original query;
- *D* = the source address from the original query;
- *PS* ={ESTABLISHED, RELATED}.

During the computation of the policy restriction (as in Section 4.4), the equivalent firewall rules are also intersected with the stateful query condition clause. The non-empty intersections become the stateful domain (introduced in Section 3.3).

Practically, in the great majority of (if not all) the cases, well configured stateful firewalls only have one single rule to manage all the stateful communications, usually at very high priority, like r_1 in Table 1. Therefore, executing stateful queries is as fast as the stateless case and produces results that include about the same number of rules.

5.4. Coping with multiple paths

When dynamic routing analysis is performed, source and destination may be connected through different paths. In that case, each path generates a separate equivalent firewall[11]. Therefore, if there are two equivalent firewall policies p_1 and p_2 where $p_1 = (R_1, \mathfrak{R}_1, E_1, a_{d1})$ and $p_2 = (R_2, \mathfrak{R}_2, E_2, a_{d2})$, corresponding to two paths from source to destination, their results need to be "merged" to quantify reachability.

The three equivalent firewall compositions introduced in Section 3.1 are formally defined as:

- lower bound, the equivalent firewall is represented as $(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, a_{d1} + a_{d2})$, that is exactly the same as the case of serially connected firewalls;
- upper bound, the equivalent firewall is represented as $(R_1 \cup R_2, \mathbb{R}_{*,\mathfrak{R}_1,\mathfrak{R}_2}, a_{d1} * a_{d2})$, where $* : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ is an action composition such that $d * d = d$ and $a * d = d * a = a * a = a$ hold, and
- highlight anomalies, the equivalent firewall is represented as $(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, a_{d1} ? a_{d2})$, where $? : \mathcal{A} \cup \{u\} \times \mathcal{A} \cup \{u\} \to \mathcal{A} \cup \{u\}$ such that $a?a = a$, $d?d = d$, and $a?d = d?a = u?* = *?u = u$ hold. Note that in this case $\mathcal{A}$ is extended to include the Unspecified action u. Rules containing the Unspecified action are the ones that present path anomalies.

[11] From the implementation point of view, since these equivalent firewalls share portions of the path, their computation can be optimized.

5.5. Query accuracy

The types of accuracy results supported are:

1. *unsupported control*, if any node in the path has a capability not in the list of supported ones. This model supports NAT, filtering, and tunnelling and not supported are proxies (with application layer authentication), load-balancers, and URL-rewriters.
2. *unsupported field*, if at least one control in the path uses a condition not supported by the geometric model (like the URL fields of application layer filters listed in Section 3.3).
3. *unsupported action*, if an action in any composition is not supported (currently only Allow and Deny can be composed by action compositions, while tunnelling actions are only used during the inverse transformation).
4. *ignored influential selectors*, if at least one of the influential selectors is not in the `select` clause (see Section 4).

6. Validation

We validated our model in different ways, to prove that our model correctly describes the real network behaviour (*correctness analysis*), our model can theoretically compute the query results in reasonable time (*complexity analysis*), and, finally, our model can practically compute the query results in reasonable time (*performance analysis*).

6.1. Correctness analysis

We validated the correctness of our model by testing whether the decisions applied by our equivalent firewall are the same as the ones applied in the real networks. There are no quantitative measure to evaluate correctness, either the results are correct or wrong (i.e., Boolean). Practically, all the compositions and transformations presented in Section 5 have been built to be equivalent to the actual behaviour of the networks. However, we report in this section a summary of the motivations that prove their correctness. We assume that the functions used to model filtering and transformation devices are correct as well as the process to actually merge policies (defined and formally proved as correct in [7]), and we prove that the atomic compositions correctly model the real network behaviour.

We consider the following atomic compositions: (1) two cascaded firewalls, (2) a filtering control preceded by a transformation one, and (3) the composition of two alternative paths.

Two cascaded firewalls. The equivalent firewall of two cascaded firewalls is represented by $(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, a_{d1} + a_{d2})$. In practice, the original rule sets are merged and the resolution strategy $\mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}$ is used. As explained in Section 5.1, $\mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}$ has been built to first use the original resolution strategies in the original policies, then the serial composition operator composes the actions. The serial composition '$+$' perfectly models the behaviour of two serially connected firewalls (i.e. drop if at least one filtering control would drop). The policy of the equivalent firewall is globally correct (i.e. the contributions from different possibly overlapping rules are managed correctly) because of the correctness of the policy merging process.

A filtering control preceded by a transformation device. In this case the equivalent firewall is described as $(R \cup R^{(T)}, \mathfrak{T}, E, d)$. There are three aspects to consider for the correctness analysis: (1) if the original packet is transformed or not, (2) if the original packet matches any of the rules in the filtering policy, and (3) if the transformed packet matches any of the rules in the filtering control (it is actually transformed). If the original packet is not transformed, we have to prove that it will not match any of the rules added by our equivalent firewall construction. Indeed, all the rules in $R^{(T)}$, the ones generated by our construction, have condition clauses derived by the fields in the transformation input clause, therefore, they will never match a packet that is not transformed. Thus, only rules in R will match the original packet, and if none of the rules in R matches it, the default action will be applied. If the packet is transformed and matches any of the rules in the filtering control, we have to prove that the equivalent firewall will enforce the same action as the original filtering control. Indeed, the equivalent firewall contains the rules in $\{\Theta(r_i, \tau_j)\}_{i,j}$ that are at higher priority than the original rules. Therefore, the equivalent firewall correctly depicts the real case. Finally, the packet is transformed but the transformed packet does not match any of the rules in the filtering control, the equivalent firewall must enforce the default action, even if the original packet matches any of the rules in the filtering control. Indeed, to address this case the equivalent firewall includes the rules in $D^{(T)}$.

The composition of two paths. In this case, the equivalent firewall is obtained by merging the equivalent firewalls computed on individual paths using either the lower bound, upper bound, or unspecified function. The proof of the correctness lies in the definition of the resolution strategies that describe how to compose the actions when rules from the initial equivalent firewalls would have enforced different actions. Indeed, in the areas of the decision space where actions from the different paths are conflicting, the lower bound selects deny, the upper bound selects allow, and the unspecified uses a dummy action, as prescribed. Even in this case, the proof relies on the correctness of the policy merging process form.

6.2. Complexity analysis

We present now a brief complexity analysis of the composition algorithms. We introduce some considerations on the number of equivalent firewalls expected in a network and the number of rules that each equivalent firewall will contain. First of all, in a network the number of zones, firewalls and rules at each firewall are correlated. In fact, hosts and servers (that populate the zones) are connected to firewalls located at the border of a hierarchically structured network (i.e. networks are mostly shaped as trees with redundant communication edges).

There are also other firewalls in the core of the network, but no hosts or servers are usually located between two intermediate firewalls, therefore, no other zones are present. Hence, the number of zones is a (small) multiple of the firewalls placed at the border of the network. The number of zone-to-zone equivalent firewalls grows quadratically w.r.t. the number of zones[12], however, the fact that the quantity of filtering zones is usually small makes our approach feasible in real scenarios. Furthermore, several types of reachability queries do not require the computation of all the possible paths from all zones, but only the actually considered ones.

According to a survey by Algosec[13], 93% of deployed firewalls have less than 1000 rules and 60% have less then 200 rules. Firewalls at the edge contain a modest number of rules, since they are connected to a limited number of zones (i.e. to a relatively small number of subnets and nodes) they have to protect. On the other hand, firewalls in the core, which can cover several zones, may have a noticeable higher number of rules. Fortunately, in practice, core firewalls rules are not very fine grained (rules are given for the entire subnets, not for individual hosts). Therefore, as confirmed by Algosec, in real networks we expect to have few firewalls with a high number of rules while the majority of the filtering devices will only contain a small quantity of rules.

In addition, the number of rules in zone-to-zone equivalent firewalls is very limited. In practice, only a subset of the rules needs to be considered, i.e. the rules such as 'the source IPs are in the first zone and the destination IPs are in the second zone'. If the firewalls have a total number of n rules and there are z zones in the network, the corresponding equivalent firewall will contain n/z rules on the average. We can easily state that the more the zones, the more the equivalent firewalls to compute, and the less the rules in the equivalent firewalls and the n/z is the most affecting parameter.

We estimate now the computational complexity of compositions. The creation of an equivalent firewall out of two firewalls (with respectively n_1 and n_2 rules) takes a constant time[14] and initially the equivalent firewall will contain $n_1 + n_2$ rules. However, it is usually convenient to remove the redundancies and those rules not matching the IP addresses in the source and destination zones. These operations have a quadratic cost in the number of rules (for rule pair analysis), and a slightly higher cost for a multi-rule analysis [7] and compression [10]. The conversion of the equivalent firewall into a FMR policy (that can be optionally done only once for the final result) is in the worst case exponential, but it has much better performance in real scenarios [7] (e.g. a few seconds for 2500 rules).

We consider now the case of the composition of a transformation policy with t rules with a firewall policy with n rules. We assume that $t = t_\Theta + t_D$ where t_Θ is the number of rules in the transformation rule set T that matches any of the rules in the firewall and t_D the number of rules that does match any rules. The composition has a complexity of $t_\Theta \cdot n + t_D$ and the resulting equivalent firewall may contain up to $t_\Theta \cdot n + t_D$ rules. Even in this case, it is possible to prune the equivalent firewall rules as the transformation rules are non-overlapping. As explained in the previous sections, it is extremely unlikely that all the transformation rules intersect all the filtering ones, also because the filtering rules regulate the access control to different filtering zones. Our experiments with realistic networks show

[12] If z is the number of zones, we need to compute all the ordered couples, that is $z \cdot (z-1)$

[13] `http://wp.eurosecglobal.de/wp-content/uploads/2013/04/www.algosec.com_resources_files_Specials_Survey%20files_12_10_11_security_complexity.pdf`

[14] Or linear if the programming language cannot merge two lists by means of references/pointers.

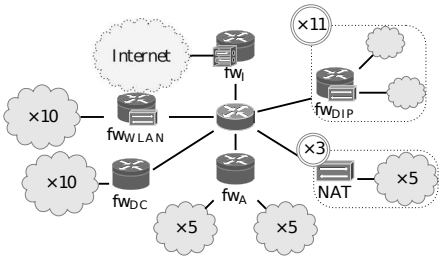


Figure 4: The sample network used for testing and validation purposes.

that the actual computational cost is reasonably low (see Section 6.3). Practically, it shows a bound of $n + Kt_\Theta + t_D$ where K is the maximum number of intersections between rules in filtering and transformation policies. According to the statistics presented by Taylor [11], K is a small integer ($k \leq 4$).

When multiple transformations apply to a path, every firewall policy of n rules is composed with all the upstream transformations in the path. If each upstream transformation has t_i rules, the complexity is theoretically $n \prod_i t_{\Theta,i} + \sum t_{D,i}$, but practically bounded to $n + K \sum_i t_{\Theta,i} + \sum_i t_{D,i} + T$ where T takes into account the intersections between rules in different transformation policies.

The number of paths, quite important for dynamic routing queries, is very limited as the networks are usually designed hierarchically[15] , thus only a few paths are configured between any two nodes (or zones) and this parameter can be considered another constant. Of course this is not valid for the Internet. However, our solution is not designed to analyse the Internet but only Intranets. In our model, the Internet is modelled as a single router that interconnects the different (geographically separated) parts of the Intranet. Therefore, all the redundant paths available in the Internet are reduced to one single device.

6.3. Performance analysis

We tested the performance of our implementation in two different scenarios. First, we used a sample network, which approximates a corporate network, that has been manually described in-depth with topology data and policies. Second, we performed more complete performance and scalability tests on synthetic networks.

We executed the tests inside a virtual machine that used two cores of an Intel Xeon E5620 (2.4 GHz) CPU, with 4 GB RAM and Red Hat Enterprise Linux 6 operating system. Each test has been performed 100 times and the results have been averaged. The tests focused on two parameters: time to compute zone-to-zone equivalent firewalls and time to execute queries. Our model has been implemented in Java 1.6. To convey the inputs to our tool, we have defined two ad-hoc XML (defined via an XML Schema), one for the network topology and one for the security policies.

For the first experiment, we considered the sample network depicted in Figure 4, where the subnets on the top right corner is marked with $\times 11$ to indicate that the scenario we tested contained eleven replications of the same subnet. Analogously, the subnet marked with $\times 3$ has been replicated three times. The whole network contains 15 firewalls, 15 NAT/NAPT devices, and about 10,000 hosts (about half of them are WLAN clients). The topology is a star network with all firewalls connected to a core router. It has 36 filtering zones with private IP addresses, all connected through a NAT/NAPT, and 30 additional zones with public IP addresses, connected to a firewall, thus making it a 68 zones wide network (if we include also the Internet). The whole network can be logically partitioned in the following parts:

- The border firewall ($\mathrm{fw_I}$) connects the core router to the Internet, contains about 150 rules, and acts also as a VPN concentrator for external remote connections.
- The data center contains various services (web, e-mail, file servers, ...) some of which are accessible also from the Internet. It uses 10 zones, connected to the core router via a single firewall ($\mathrm{fw_{DC}}$) with about 50 rules.
- There are 10 wireless networks, connected to the core router via a single device ($\mathrm{fw_{WLAN}}$) acting as NAT/NAPT and firewall, with about 20 rules. Wireless clients can access Internet and public parts of the corporate network.

[15] `http://docwiki.cisco.com/wiki/Internetwork_Design_Guide`

- The administration network uses 5 zones for its servers and an additional 5 zones for the employees' workstations. All these zones are connected to a firewall ($\mathrm{fw_A}$) containing approximately 250 rules. The services in the 5 server zones have higher security requirements than the ones allocated in the data centre and are accessed mainly by the users from the 5 administration user zones.

- Laboratories use 3 private networks, each one consisting of 5 zones. Each private network is connected to the core router via an individual NAT/NAPT device (NAT). These laboratories can access the corporate network and the Internet, but are not visible from the outside.

- There are 11 departmental networks, composed of 2 zones each: a private one, connected to the core router by means of a NAT/NAPT and a firewall ($\mathrm{fw_{DIP}}$), and a public zone connected just via the firewall ($\mathrm{fw_{DIP}}$) to the core router. The private zone can reach the corporate network and the Internet but is not visible from outside. The public zone is reachable from the Internet.

For this network, there are 4624 equivalent firewalls and our implementation calculates all of them in about 3 minutes. On average, each equivalent firewall contains 130 rules and the composition of two equivalent firewalls for multi-path analysis takes less than 1 ms. The queries performed over these equivalent firewalls required at most 3 ms to be executed.

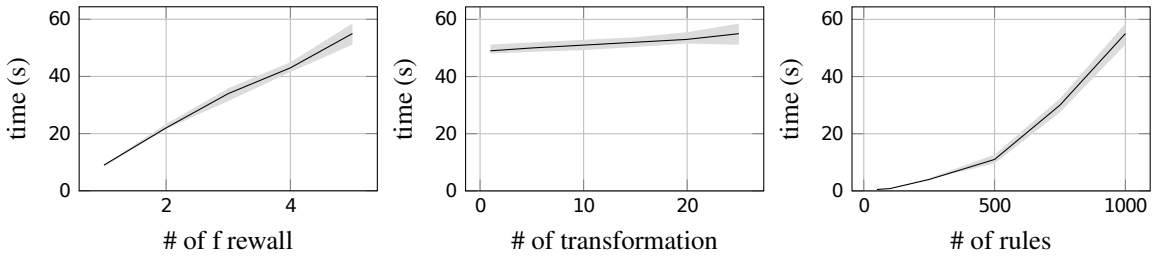
For the second category of performance testing, we performed an extensive performance and scalability analysis on synthetic networks. The presented tests focus on how long it takes to calculate one single equivalent firewall. The calculation of all the equivalent firewalls in a network has a linear dependency on the number of paths and therefore can be easily calculated based on our results. This approach seems the most efficient since the number of equivalent firewalls is limited (Section 6.2) and depends on how many variables need to be tested.

We generated reachability queries between two zones, containing 100 clients and 100 servers respectively. Zones are connected by a network path containing an increasing number of cascaded firewalls (up to 5), each one configured with up to 1000 filtering rules. The translation rules implement one of the common transformation policies: NAT 1-to-1, NAT many-to-1, and tunnelling. We used them to generate transformation policies that we applied in different positions of the path, either before or after a firewall (or both). In the worst case scenario, we considered paths with 25 translation policies and 5 firewalls, with 1000 filtering rules each.

Practically, we used three independent parameters in our experiments: (i) f, the number of firewalls in each path, variable from 1 to 5; (ii) n, the number of rules per firewall, which varies from 50 to 1000; (iii) t, the number of translation policies for each path, which varies from 1 to 25.

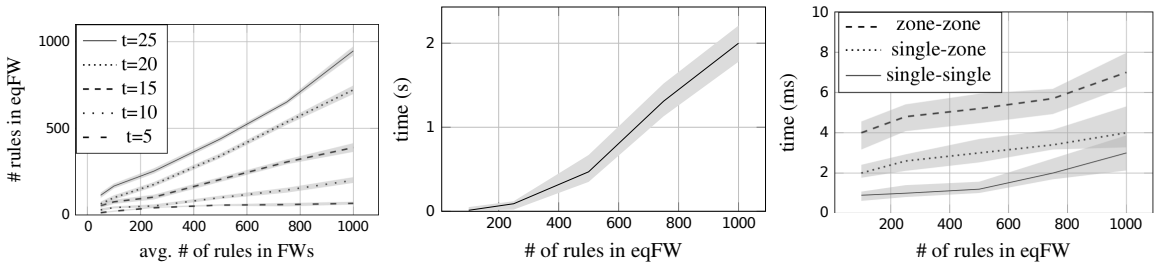
Generating rules for the synthetic network test scenarios has been a complex task. Indeed, random generation is not suitable, as rules in cascaded firewalls must be related to allow end-to-end communication paths. On the other hand, manually specifying policies was impossible for the large networks we decided to test. Hence, we constrained the random process with several requirements. First, we ensured reachability between selected filtering zones. This implies that "allow rules" must be carefully placed in selected paths. Indeed, when a firewall blocks the communications for the entire query condition clause, the computation of the equivalent firewall stops and tests are imprecise. Furthermore, we generated rules according to the statistical data from Taylor [11] and average number of anomalies from Al-Shaer [12]. These works estimate the number of rules that intersect simultaneously (up to 5) and the number of intersecting rules in a rule set (at most 25%).

Figure 5 shows the average time needed to calculate one zone-to-zone equivalent firewall. For all the tests we calculated a 95% confidence interval, which is depicted as a grey area around the graphs. The first plot (Figure 5a) shows the time to compute the equivalent firewall when the number of firewalls varies from 1 to 5, if the number of rules per firewall is fixed at 1000 and the number of transformations per path is fixed at 25. The second plot (Figure 5b) shows the time to compute the equivalent firewall when the number of transformations per path varies from 1 to 25 if there are 1000 rules per firewall and exactly 5 firewalls per path. Finally, the third plot (Figure 5c) shows the time to compute the equivalent firewall when firewall rules vary from 100 to 1000 if the number of firewalls per path is 5 and there are 25 transformations per path. It is possible to see that the computation time depends linearly on the number of firewalls in the path (Figure 5a) and the number of translation rules (Figure 5b). The number of rules is the factor that affects the execution time the most. As shown in Figure 5c, the computation time quickly increases with the number of rules and, in the worst case scenario, reaches about 50 s. These results prove that, since the equivalent firewall has to be calculated only once, the computation time is more than acceptable also considering



(a) Time depending on f, with $n = 1000, t = 25$.
(b) Time depending on t, with $f = 5, n = 1000$.
(c) Time depending on n, with $f = 5, t = 25$.

Figure 5: Time to create zone-to-zone equivalent firewalls.



(a) Number of rules in equivalentFW depending on the rules in the original firewall policies.
(b) Time to compose two parallel paths depending on the number of rules in the equivalentFW.
(c) Time to execute a query depending on the number of rulesin the equivalentFW.

Figure 6: Time to compute equivalent firewalls and executing queries.

the huge number of firewalls and transformations in the path. It is worth noting that the majority of firewalls contain less then 200 rules, where our algorithm takes less than 5 s.

Figure 6a plots the number of rules in the equivalent firewall obtained by serially composing two firewalls, depending on the number of rules in the original firewalls. We see that the number of rules in the equivalent firewall depends on the number of rules in the involved policies and the number of transformation rules, but it does not depend on the number of involved policies. Figure 6b shows the average time to compose two zone-to-zone equivalent firewalls computed on different paths between the same two zones. The calculation time has a slightly more than linear dependency on the number of rules in the equivalent firewalls. In the worst case tested (the composition of two equivalent firewalls with 1000 rules each), the computation takes less than 2 s. We measured that the time to perform this computation is independent on the strategy (lower bound, upper bound or highlight anomalies). Figure 6c plots the average time needed to execute a query on an already computed equivalent firewall. Three types of queries are evaluated: reachability between two hosts (single-single), between a host and a filtering zone (single-zone), and between two filtering zones (zone-zone). We measured that query computation time has a linear dependency on the number of rules in the equivalent firewall. In the worst case scenario, the algorithm only needs 7 ms to calculate the query result, thus query execution is instantaneous, once the equivalent firewall has been created.

7. Related works

We present the most influential work in this field and compare them with our proposal.

7.1. Reachability analysis

Authors in [13] define an algorithm for offline reachability analysis on networks containing packet filters, routers and NAT transformations. Additionally, they proposed the definition of a standard set of queries as "user do not know what to query". However, their solution does not support tunnels and the query interface offers only basic functionalities. Xie based its reachability analysis on graph theory and dynamic programming [1]. Network is modelled as a

triple containing the set of routers, their physical connections and a set of labelling functions used to express packet filtering and transformation rules, so that firewalls and NAT devices can be described. This approach is purely theoretical and thus lacks any experimental results. In addition, it can only be used to represent static NAT and filtering rules based on the destination addresses, and it does not take into account the existence of connectionless and connection-oriented protocols and cannot provide exact reachability values, but only upper and lower bounds. The work in [2] overcomes some limitations of Xie's one. Authors use a more general model for describing firewalls, packet filtering and transformation rules, thus adding the possibility to handle policies that depend on the source addresses and the filtering states. Also in this case, performance analysis is absent. Another theoretical approach, used to compute the network-wide reachability was proposed in [3] where authors use traditional graph-based algorithms, such as Floyd-Warshall, whereas [1] and [2] require ad-hoc techniques to mimic routing protocols. Additionally, they describe how to represent both routing and filtering controls, but do not mention how to express packet transformations rules. [14] is a generalization of Xie's work based on "header space" information of packets. This approach is compatible with filtering, routing, and transformation technologies, but it is limited to the classic 5-tuple rules of packet filters and cannot be used for filtering and security devices that work at a higher level of the network stack.

Mai et al. [15] present a reachability verification algorithm which is expressed as constraints rather then sets of packets. The algorithm is based on a SAT solver, where the reachability query is represented as a Boolean formula. The SAT solver searches for a symbolic packet which can be forwarded between two vertices of the network.

In [4] authors show how to use FDDs (Firewall Decision Diagrams) to compute the reachability of a network. This model supports packet routing, filtering and transformation rules. The implementation works in two steps. First, a pre-computation phase (which can require hours on large networks) is performed. Afterwards, the tool can be used to retrieve the reachability, by using a SQL-like language named SQRL. We consider this work as the most advanced one before our work.

Networks can also be viewed as giant finite state machines as in [5], where authors model rules as a Boolean expression and represent them by means of BDDs (Binary Decision Diagrams). Rules can be analysed by using CTL (Computation Tree Logic) and symbolic model checking to infer the reachability. Authors model routers, firewalls, NAT devices and IPsec gateways, but not NAPT or other packet transformation policies. Also this approach requires a long pre-computation phase before performing the actual reachability queries.

The presented model and reachability analysis algorithm has the advantage over similar works in literature. We summarize them in Table 3. The equality symbol (=) indicates that the proposed model has the same features and the plus symbol (+) indicates that our approach has this additional feature. Compared to [1, 2, 3], we support a broader range of security controls. Our model has an advantage over [4] because it has better performance, a more precise query language, and it supports also end-system firewall. It is superior with respect to [5] because it supports zone-to-zone queries and end-system firewalls.

Paper	Filtering	NAT	IPsec	End-System	Zone-to-Zone
[1, 2, 3]	=	+	+	+	+
[4]	=	=	=	+	=
[5]	=	=	=	+	+

Table 3: Comparison between the state of the art and the proposed model.

7.2. Configuration verification

A different approach to reachability is *configuration verification* that uses a formal model of the desired network behaviour as a baseline and checks if the network follows it or not. The perspective is slightly different, as the behaviour must be defined in advance. An automated validation method for security policies is presented in [16] for firewall configuration validation and reachability analysis. Authors formally introduce the concept of Executable Security Policies and use reachability graphs to ensure that they are conform with the firewall configuration. The proposed solution can only be applied to the validation of a single packet filter configuration based on 5-tuple rules. The model does not support other firewall types, translation controls, routers and tunnels.

Finally, recent literature [17, 18] has extended the firewall analysis towards the configuration verification of middleboxes. Middleboxes are stateful network functions that process traffic based on their configurations and internal states, like a learning firewall.

7.3. Anomaly analysis

Another orthogonal area of research with a similar objective, that is, discovering errors in the implementation of authorization policies, provides models and tools to perform *anomaly analysis* whose objective is to identify possible configuration errors by identifying simultaneously activated rules, i.e. anomalies. This approach cannot interpret the "meaning" of rules but only the relations between them, therefore is has a main disadvantage: it can not identify undesired behaviours of the network. Even in the best case, when numerous redundant and unnecessary rules are identified, correcting these anomalies does not guarantee that the security controls behave as expected. However, it is interesting for our research as it can be applied to individual firewall policies and to equivalent firewalls. The anomaly analysis was first introduced by Al-Shaer [19] for packet filters and serially connected packet filters. From this paper, other researchers have proposed alternative models and classifications. Garcia-Alfaro et al. [20] proposed MIRAGE, a tool for the analysis and deployment of configuration policies also capable of configuring network intrusion detection systems (NIDS) and VPN routers. Other tools are Firecrocodile [21] that works on Cisco PIX rules, FIREMAN [22] that also checks if the filtering configurations correctly implement an end-to-end policy.

All these works share the same disadvantages even if they rely on different analysis algorithms: none of them can handle other security technologies but packet filters. An example of stateful analysis of a firewall configuration is presented in [23]. Finally, [7, 24, 25] generalizes Al-Shaer's classification to multi-rule anomalies, i.e. involving more than two rules.

8. Conclusions and Future Work

This paper has presented a static reachability analysis approach based on the construction of end-to-end equivalent firewalls, using a geometric model of policies.

Our model provides fast and accurate reachability results coupled with an excellent flexibility, allowing its application to a wide array of areas such as traditional corporate networks, but also critical infrastructure scenarios.

The proposed approach can represent a large variety of network security controls, such as routers, stateless and stateful firewalls, NAT/NAPT and tunnelling controls, and allows administrators to execute precise queries, thanks to a rich set of conditions types and an improved support for stateful reachability analysis. The approach is efficient and presents good scalability results. We also provide an extension of the SRQL format supporting a larger set of reachability analysis options and a richer set of condition types, covering also application layer filters. Moreover, our approach offers more complex types of Boolean expressions that allow to better characterize the traffic to query.

Our future work aims at increasing the expressiveness of the tool to be as close as possible to real environments. The mathematical model of equivalent firewalls can be extended to support other categories of security controls currently unsupported, like forward and reverse proxies. For instance, currently, our approach does not fully support all the capabilities of a proxy since we cannot model authentication systems and (stateful) authentication information. In addition, our model only supports HTTP and some stateful characteristics of the FTP and DNS protocols. However, we are planning to handle other protocols (e.g. SSH) and message protection techniques (e.g. WS-Security) as well.

Moreover, this approach suits well virtualized network environments, where most of the topological and configuration data needed to perform an analysis are available at the Management element. Integrating our tool in virtualized environments and extending our analysis to Software Defined Network is an interesting future improvement.

Acknowledgment

The research described in this paper has been partly supported by the SECURED (FP7 grant agreement no. 611458) and SHIELD (H2020 grant agreement no. 700199) project , co-funded by the European Commission.

References

[1] G. Xie, D. Maltz, A. Greenberg, G. Hjalmtysson, J. Rexford, On static reachability analysis of IP networks, in: Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, 2005, pp. 2170–2183.

[2] S. Bandhakavi, S. Bhatt, C. Okita, P. Rao, Analyzing end-to-end network reachability, in: Int. Symp. on Integrated Network Management, IEEE, 2009, pp. 585–590.

[3] M. Sveda, O. Rysavy, G. D. Silva, Static analysis of routing and firewall policy configurations, in: ICETE 2010, Athens, 2010, pp. 39–53.

[4] A. Liu, A. Khakpour, Quantifying and Verifying Reachability for Access Controlled Networks, IEEE/ACM Trans. Netw. 21 (2) (2012) 551–565.
[5] E. Al-Shaer, W. Marrero, Network configuration in a box: Towards end-to-end verification of network reachability and security, in: Int. Conference on Network Protocols, IEEE, 2009, pp. 123–132.
[6] C. Basile, D. Canavese, A. Lioy, C. Pitscheider, Improved reachability analysis for security management, in: Euromicro Int. Conference on Parallel, Distributed, and Network-Based Processing, IEEE, 2013, pp. 534–541.
[7] C. Basile, A. Cappadonia, A. Lioy, Network-level access control policy analysis and transformation, IEEE/ACM Trans. Netw. 20 (4) (2012) 985–998.
[8] A. X. Liu, M. G. Gouda, Firewall policy queries, IEEE Trans. Parallel Distrib. Syst. 20 (6) (2009) 766–777.
[9] C. Basile, A. Lioy, Analysis of application-layer filtering policies with application to http, IEEE/ACM Trans. Netw. 23 (1) (2015) 28–41.
[10] A. X. Liu, E. Torng, C. R. Meiners, Firewall compressor: An algorithm for minimizing firewall policies, in: INFOCOM, IEEE, 2008, pp. 176–180.
[11] D. Taylor, Survey and taxonomy of packet classification techniques, ACM Comput. Surv. 37 (3) (2005) 238–275.
[12] E. Al-Shaer, H. Hamed, Modeling and management of firewall policies, IEEE Trans. Netw. Service Manag. 1 (1) (2004) 2–10.
[13] A. Mayer, A. Wool, E. Ziskind, Offline firewall analysis, Int. J. of Information Security 5 (3) (2006) 125–144.
[14] P. Kazemian, G. Varghese, N. McKeown, Header space analysis: Static checking for networks, in: USENIX conference on Networked Systems Design and Implementation, ACM, 2012, pp. 113–126.
[15] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, S. T. King, Debugging the data plane with Anteater, in: SIGCOMM conference 2011, ACM, Toronto, Canada, 2011, pp. 290–301.
[16] R. Abassi, S. Guemara El Fatmi, A model for specification and validation of security policies in communication networks: The firewall case, in: Int. Conference on Availability, Reliability and Security, IEEE, 2008, pp. 467–472.
[17] S. Spinoso, M. Virgilio, W. John, A. Manzalini, G. Marchetto, R. Sisto, Formal verification of Virtual Network Function graphs in an SP-DevOps context, in: ESOCC: European Conference on Service-Oriented and Cloud Computing, Springer, Cham, 2015, pp. 253–262.
[18] F. Valenza, T. Su, S. Spinoso, A. Lioy, R. Sisto, M. Vallini, A formal approach for network security policy validation, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 8 (1) (2017) 1–20.
[19] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan, Conflict classification and analysis of distributed firewall policies, IEEE Selected Areas in Communications 23 (10) (2005) 2069–2084.
[20] J. Garcia-Alfaro, MIRAGE: a management tool for the analysis and deployment of network security policies, in: Int. SETOP Workshop on Autonomous and Spontaneous Security, Springer, Berlin, Heidelberg, 2010, pp. 203–215.
[21] N. Lehmann, R. Schwarz, J. Keller, Firecrocodile: A checker for static firewall configurations, in: Int. Conference on Security & Management, IEEE, 2006, pp. 193–199.
[22] L. Yuan, H. Chen, J. Mai, C.-n. Chuah, FIREMAN: A toolkit for firewall modeling and analysis, in: Symp. on Security and Privacy, IEEE, 2006, pp. 199–213.
[23] J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, S. Martinez, J. Cabot, Management of stateful firewall misconfiguration, Computers & Security 39, Part A (2013) 64 – 85, 27th {IFIP} International Information Security Conference.
[24] C. Basile, D. Canavese, A. Lioy, C. Pitscheider, F. Valenza, Inter-function anomaly analysis for correct sdn/nfv deployment, International Journal of Network Management 26 (1) (2016) 25–43.
[25] F. Valenza, S. Spinoso, C. Basile, R. Sisto, A. Lioy, A formal model of network policy analysis, in: 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015, pp. 516–522.

Cataldo Basile received a M.Sc. (summa cum laude) in 2001 and a Ph.D. in Computer Engineering in 2005 from the Politecnico di Torino, where is currently a research assistant. His research is concerned with policy-based management of security in networked environments, policy refinement, general models for detection, resolution and reconciliation of specification conflicts, and software security.

Daniele Canavese received the M.Sc. degree in computer engineering in 2010 and a Ph.D. in Computer Engineering in 2016 from the Politecnico di Torino, where he is currently a research assistant. His research interests are concerned with policy-based management systems, models for network analysis, security management via inferential systems and public key cryptography.

Antonio Lioy is full professor at the Politecnico di Torino, where he leads the TORSEC research group active in information system security. His research interests include network security, policy-based system protection, and electronic identity. Lioy received a M.Sc. in Electronic Engineering (summa cum laude) and a Ph.D. in Computer Engineering, both from the Politecnico di Torino.

Christian Pitscheider received the M.Sc. degree in Computer Engineering in 2012 and a Ph.D. in Computer Engineering in 2016 from the Politecnico di Torino. His research is focused on policy-based systems for security management in networked environments, and general models for detection, resolution and reconciliation of conflicts.

Fulvio Valenza received the M.Sc. degree (summa cum laude) in computer engineering in 2013 from the Politecnico di Torino, where he is currently a third year PhD student. He is working as a Researcher with the CNR-IEII Torino, Italy. His research activity focuses on network security policies, with particular emphasis on policy conflict analysis, resolution, comparison, refinement and optimization.