Static Analysis of Routing and Firewall Policy Configurations

Miroslav Sveda, Ondrej Rysavy, Gayan de Silva, Petr Matousek, and Jaroslav Rab

Brno University of Technology, Brno 612 66, Czech Republic
sveda@fit.vutbr.cz
http://www.fit.vutbr.cz/∼sveda/

Abstract. Network design that meets customer's security requirements needs careful considerations when configuring routing and filtering rules. This paper deals with an approach to security analysis based on reachability calculations in dynamically routed networks. The contribution consists of proposing routing abstract model that enables to extend existing reachability analysis approaches to obtain a finer approximation. This approximation captures the effect of routing on packets forwarding. Thus in the combination with reachability calculations based on packet filtering analysis it provides valuable information for a network designer on possible security issues in designed network.

1 Introduction

Fundamental steps in network design consist of definition and implementation of routing and access control mechanisms. Network designers are seeking such configurations that would best implement customers requirements, while considering the limits of underlining technologies. The goal is to provide reliable network services as requested. With growing complexity of network designs the task of configuration implementation, maintainance and modification becomes a challenge for network administrators. It is widely understand that network misconfigurations together with device failures stand behind the most of network outages. Implementation of supportive tools that can identify potential configuration problems to the network design process can valuably aid network designers and administrators.

Particularly difficult task is a configuration of firewall rules. Based on analysis of real configuration files, Wool reports on possible threats of incorrectly configured firewalls [25] and provides suggestions for improving the quality of firewalling rules. One of the observations made concerns about the complexity and size of firewall rulesets. He points out difficulty of creating errorless complex firewall configurations. Although Wool considers only a small set of relatively obvious errors, the survey demonstrates that a ruleset having 1000 items includes more than 8 errors in the average.

The paper focuses on the area of automatic analysis of a network that consists of L3 devices (hosts, routers, firewalls etc.) connected by links and, optionally, with firewall rules applied on them. Based on the network configuration and considering dynamic behavior of the network, we can ask questions like "Is this network protected against P2P connections?", "What packets can be delivered to the given host?", "Is this WWW

M.S. Obaidat, G.A. Tsihrintzis, and J. Filipe (Eds.): ICETE 2010, CCIS 222, pp. 39–53, 2012.

service accessible under every state of the network?", or "What can cause the unreachability of the important bussiness service?".

Of course, those questions can be partially answered by scanning and testing tools (ping, nmap), or vulnerability assessment tools (Nessus). However, testing can analyze the network only in immediate state, which means in practice, for a fixed configuration. When the topology is changed, the response of the network can be different. In our work we explore how security and safety properties can be verified under every network configuration by employing principles known from program and computer system verification. The model checking [9] is a technique that explores all reachable states and verifies if the properties are satisfied over each possible path to those states. Model checking requires specification of a model and properties to be verified. In our case, the model of network consists of hosts, links, routing information and access control lists. Specifications of properties have form of security policy descriptions expressible in a formal language.

1.1 Our Contribution

Our approach is close to the work by Xie et al. [27], J. Burns [3], and Bera, Ghosh and Dasgupta [6]. Unlike these works we build a model that includes both static and dynamic behavior, i.e. firewall rules and routing information, see [24]. In this model, the verification of reachability properties can be made. In comparison to Ritchey's work (Ritchey and Ammann, 2000) we do not focus on hosts vulnerability and their resistance to attacks, but on stability of services in dynamic networks.

Main contribution of this paper consists in a method that integrates the effect of routing to reachability analysis based on evaluation of distributed access control lists. Contrary to work by Xie et al. [27], where routing is modeled by extending filtering with route related rules, we developed a split model in which the routing information is analyzed from the network wide perspective. The information from routing analysis is employed in reachability analysis implemented by existing methods. This paper is an extension of our previous work published in [22].

1.2 Structure of the Paper

In section 2, we review the state of the art in the area of automatized configuration analaysis of routing and filtering. In section 3, we provide backgrounds and fundamental definitions for the rest of the paper. In section 4, we present overview of our novel approach to routing modeling and analysis. In section 5, we discuss the issue of packet filter represetnation. In section 6, we review the method for analysis of security policies in network models and examine couple of subtle issues in more detail. The paper concludes in section 7, in which we give a summary and present the possible future work.

2 State of the Art

Research in the area of network security and vulnerability detection has been conducted since the beginning of the Internet. Many papers concentrate on detection of

vulnerabilities of hosts and their protection against the network attack, see e.g. [23], [30], or [20].

In [28], an automatic deduction of network security executed in Prolog is introduced. The authors define reasoning rules that express semantics of different kinds of exploits. The rules are automatically extracted from the OVAL scanner and the CVE database [17].

Another approach is an automatic generation of network protection in the form of firewall rules as shown in [4]. The security policy is modeled using Model Definition Language as the first step. Then, the model of a network topology is translated into firewall-specific configurations. These configuration files are loaded into real devices (firewalls).

Ritchey and Ammann in [19] explain how model checking can be used to analyze network vulnerabilities. They build a network security model of hosts, connections, attackers and exploits to be misused by the attacker. Security properties are described by temporal logics and verified using the SMV model checker. However, their goal is different from ours. They verify if hosts on the stable network are vulnerable to attacks. In our case we concentrate on dynamically changing networks and reachability of their nodes.

In 1997, Guttman defined a formal method to compute a set of filters for individual devices given a global security policy [12]. To achieve a feasibility, the network is abstracted such that only network areas and border routers occur in a model. This decision reflects naturally the real situation as internal routers usually do not participate in data filtering. Similarly, data flow model is defined in terms of abstract packets, which are described by abstract source and destination addresses and service type. An algorithm computes a feasibility set of packets that can pass all filtering rules along the path. The rectangle abstraction of packet representation makes the procedure practical and efficient.

Yan et.al have developed a tool called FIREMAN [29], which allows to detect misconfigurations in firewall settings. The FIREMAN performs symbolic model checking of the firewall configurations for all possible IP packets and along all possible data paths. The underlaying implementation depends on a Binary Decision Diagram (BDD) library, which efficiently models firewall rules. This tool can reveal intra-firewall inconsistencies as well as misconfiguration that leads to errors at inter-firewall level. The tool can analyze Access Control List (ACL) series on all paths for end-to-end connection thus offering network-wide firewall security verification.

Jeffrey and Samak in [14] aims at analysis firewall configurations using bounded model-checking approach. They focus at reachability and cyclicity properties. To check reachability, it means to find for each rule r a packet p that causes r to fire. To detect cyclicity of firewall configuration, it means to find a packet p which is not matched by any rule of the firewall ruleset. They implemented analysis algorithm by translating the problem to SAT instance and showed that this approach is efficient and comparable to tools based on a BDD representation. Similar work was done by Pozo, Ceballos and Gasca [18] who provided a consistency checking algorithm that can reveal four consistency problems called shadowing, generalization, correlation and independency.

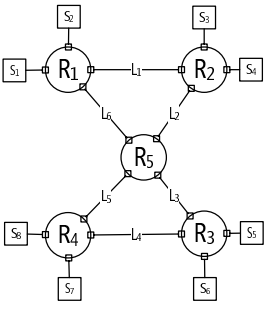
Liu et.al developed a method for formal verification and testing of (distributed) firewall rules (see [16] and [10]) against user provided properties. They represent firewall rules in a structure called firewall decision diagram (FDD), which forms an input to a verification algorithm. Another input is a property rule, which describes a property that they want to check, e.g. description of a set of packets that should pass the firewall. By a single traversing an FDD from the root to a leave it is possible to check the given property.

Xie et.al reports in [26] on their extensive work on static analysis of IP networks. They define a framework able to determine lower and upper approximations on network rechability considering filtering rules, dynamic routing and packet transformations. The method computes a set of packets that can be carried by each link. By combination of these sets along the possible paths between two end points, it is possible to determine the end-to-end reachability. The upper approximation determines the set of packets that could potentially be delivered by the network, while the lower approximation determines the set of packets that could be delivered under all possible forwarding states. In their paper, the authors also present a refinement of both upper and lower approximations by considering the effect of dynamic routing.

Bera, Dasgupta and Ghosh (see [7], and [6]) define a verification framework for filtering rules that allows one to check the correctness of distributed ACL implementations against the given global security policy and also to check reliability (or fault tolerance) of services in a network. To check the correctness, the filtering rules are translated to assertions in the form of first order logical formulas. They are together with logical description of global security policy sent to SAT solver that mechanically checks the satisfiability. In the case of an inconsistency, the SAT solver produces a counter example that helps an administrator with debugging ACL rules. To check the reliability, the framework accepts a description of a global security policy, a collection of ACL rules and a network description and computes whether the rules are consistent with the given policy. Policy is understood as a description of service availability with respect to defined network zones. The method used computes first a network access model, which is a directed graph with ACLs assigned to its edges. Next, the service flow graphs (SFG) are generated for all services in the interest, e.g. SFG for ssh traffic. An SFG is a subgraph of the network accesss graph. The fault analysis is performed by computing minimum cuts in all SFGs. These values then represent how many link failures can be tolerated.

3 Backgrounds

In this section, we establish some basic facts and definitions that we use in the development presented in the paper. We first define terminology required to understand the problem formulation and the method proposed. We then describe a graph model for computing reachability. At the end of the section, we define the concept of forwarding states, which captures the dynamics of the network and creates a framework for what-if analysis.



(a) Network Topology

$\mathcal{F}$	*Links*	$\mathcal{F}$	*Links*	$\mathcal{F}$	*Links*
f_{12}^{L}	$L_1 \rightarrow L_2$	f_{12}^{S}	$S_1 \rightarrow S_2$	f_{11}^{SL}	$S_1 \rightarrow L_1$
f_{21}^{L}	$L_2 \rightarrow L_1$	f_{21}^{S}	$S_2 \rightarrow S_1$	f_{11}^{LS}	$L_1 \rightarrow S_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
f_{54}^{L}	$L_5 \rightarrow L_4$	f_{87}^{S}	$S_8 \rightarrow S_7$	f_{58}^{SL}	$L_5 \rightarrow S_8$
f_{45}^{L}	$L_4 \rightarrow L_5$	f_{78}^{S}	$S_7 \rightarrow S_8$	f_{85}^{LS}	$S_8 \rightarrow L_5$

(b) Filter map

Fig. 1. A Simple Network Example

3.1 Terminology

We assume that networks under analysis consist of intermediate devices (routers) that forward packets through the network according their internal knowledge. We assume that routers can be configured with a mix of static and dynamic routing information. In the present paper, we use the following terms which refer to various routing related concepts:

- *Local Routing Information Base (RIB)* is stored in routing process address space running on a router. Each process has its own RIB, e.g. RIP maintains RIP database. Similarly, *local Forwarding Information Base (FIB)* is a single datatable, which is used by a router to decide where to forward incoming packets.
- *Network RIB/FIB* is a network wide view of routing information. This represents a shared routing knowledge of forwarding devices. Similarly, network FIB represents a global view on routing information that governs forwarding packets in the entire network.
- *Forwarding device* is a network device that actively decides where to forward packets based on its local routing information stored in FIB.
- *Redistribution* stands for copying route information to a target protocol instance, which is done in the scope of a single router.
- *Reachability Set* is a set of packets that can be delivered to the given network location.
- *Routing instance* is a collection of routing processes of the same type on different devices. They share the routing information in order to create the consistent view on (a part of) network.

3.2 Abstract Network Model

The abstract model of the network dealing with routing processes and packet filtering stems from a combination of techniques introduced in [27] and [8]. The network is regarded as a directed hypergraph where vertices are routing devices and edges are communication channels that form abstractions of communication links. In real networks there are other network device then routers. However, every end-point device, such as PC or Web server, can be represented using a router with only one interface, and one outgoing filtering rule representing routing all traffic to a default gateway. Formally, an *abstract network model*, see example in Figure 1, is a tuple $N = \langle R_N, L_N, F_N \rangle$, where

- R_N is a finite set of network devices,
- L_N is a finite set of links between routers, and
- $F_N : L_N \times L_N \rightarrow \mathcal{F}$ is a function assigning to each pair of links a filter from the set of filters $\mathcal{F}$.

Filters can have various represetation. For instance, similarly to [26], we can define filters to be reachability sets. A reachability set is a set of all packets that are permited by the filter. There are defined standard operations on such set; thus, it is possible to compute a reachability set over a collection of filters by applying the intersection operation. Benson, Akella and Maltz in [5] showed the symbolic computation of reachability sets based on normalized ACL representation. Other approach, which is used in this paper represents filters as logical formulas, e.g.

$$(dst\_adr \in 10.10.23.0/24 \vee dst\_adr \in 10.10.12.0/24) \wedge proto = udp \wedge dst\_port = 53$$

that allows only IP packets destined for 10.10.23.0/24 and 10.10.12.0/24 subnetworks carrying UDP datagrams with destination port 53. Such representation stands for the characteristic formula of the corresponding reachability set. In section 5, we deal with this representation in more detail.

3.3 Forwarding State

A forwarding state describes a condition of a network determined by a content of network forwarding information base (NFIB). In general, the content of NFIB for the given network depends on the health of devices, state of links, and external routing information pumped into the network.

Consider again the toy network in Figure 1. All routers is running RIP and exchange all information on connected subnetworks. We show how the NFIB depends on the state of links. RIP uses hop count metrics and thus the shortest path from S_1 to S_6 found is $\langle S_1, R_1, L_6, R_5, L_3, R_3, S_6 \rangle$. This path is placed in NFIB. If links L_6 and L_3 were unavailable then NFIB would contain longer path $\langle S_1, R_1, L_1, R_2, L_2, R_5, L_5, R_4, L_4, R_3, S_6 \rangle$. Similarly for other paths we can define NFIB for two forwarding states q and r as follows:

q	S_1	...	S_6	...
S_1	·	...	R_1, L_6, R_5, L_3, R_3	...
⋮	⋮	⋮	⋮	⋮

r	S_1	...	S_6	...
S_1	·	...	$R_1, L_1, R_2, L_2, R_5, L_5, R_4, L_4, R_3$	...
⋮	⋮	⋮	⋮	⋮

It is possible that different combinations of link states or device failures can lead to the same forwarding state. It is because links have different topological importance for packet forwarding. The assesment of links and routers from the viewpoint of their importance on traffic delivery was done, e.g. in [21]. In this paper, we do not deal with packet transformations, which is performed by the network gateways such as NAT gateway or MPLS routers. This would complicated the representation of NFIB and we left it for future work.

4 Routing Approximation

Geoffrey G.Xie et al. in [27] showed that routing information can be added to the static model of the network using additional filtering rules. These filtering rules can be changed as the state of links is changed, so the filtering rules depend on the actual state of the network. In this paper, we present an alternate approach to approximate routing. Instead of simulating routing protocols to find the content of routers' FIBs, we compute the network RIBs and network FIBs directly by using standard graph algorithms.

4.1 Routing Protocols

Routing protocols employ distributive algorithms to compute the common view on a network in each engaged router. In the simplest case, when no customization of routing is applied the routers compute the shortest path to every known destination with respect to the defined metrics. If the network changes its state, which is triggered by, for instance, link or device failure, the routers engage in the process of disseminating updates and computing new local routing tables.

Configuration of dynamic routing protocols can include number of options that affect the distribution of routing data. For instance, configuring passive interfaces prevents a router to send RIP updates through this interface thus the connected routers are not informed about the routing related events. Similarly, using ACL to filter outgoing routing updates can be used to customize the routing information that spreads from the router. Different instances of routing protocols implicitly do not share routing information. Nevertheless, route redistribution can be configured to insert a selected routing information from a source RIB to a destination RIB whithin a router.

A router selects and installs the best route from a collection of local RIBs based on the defined priorities. This route is then used for packet forwarding, which means to determine links used for forwarding packets to their destinations.

4.2 Calculation of Routing Approximation

Our aim is to determine a reachability matrix that would contain all best paths between pairs of destinations in a given forwarding state induced by the current state of links and routers. We use a cost-path pair for representing a forwarding path in the reachability matrix. This reachability matrix corresponds to a network-wide view of the Forward information base (FIB). The calculation is performed in several steps as follows:

1. computation of network RIBs for static routing information,
2. computation of network RIBs for every routing instance, and
3. (repeated) application of the effect of redistribution and selection of the best paths to the network FIB.

Now, we give some details for these computational steps. Generally, in every step we begins with defining an adjacency matrix, which can be obtained directly from the analysis of configuration files. Then, the standard graph algorithm, efor instance, Floyd-Warshall's algorithm [11] with $|V|^3$ time complexity, is applied to get all shortest paths.

Depending on the step, we get either resulting reachability matrix or we need to (iteratively) perform an additional update operation and a recalculation of RIBs.

Calculation of Static RIBs. A static route configuration is analyzed and the adjacency matrix Adj_S for every destination S is constructed. If router R has defined a static route for destination S such that packets should be forwarded by link L, which connects router Q, then $Adj_S[R, Q] = 1$. A RIB matrix is computed for every unique adjacency matrix. A static reachability matrix for destination(s) S, RIB_S^{Static}, is initialized with adjacency matrix and then the transitive closure is computed. During the compution the usual path concatenation operation is used. A column $RIB_S^{Static}[*, S]$ then defines all possible forwarding paths to the destination S from any node in the network. This matrix defines a reachability graph, which can be discontinued depending on the completeness of information implemented by static routing configuration.

Calculation of Dynamic RIBs. Network RIBs that correspond to dynamic routing protocols is computed using the same techniques but in a different way. Adjacency matrix,Adj_S^P, of routing instance P for each destination S, is determined by analysis adjacency among routing processes within a routing instance. If there are no routing update filters then it is sufficient to consider a single adjacency matrix Adj^P. We will consider the case, in which routing update filters are deployed in the network.

Route update filtering works by regulating the route advertisments sent to neighboring routers and filtering routes advertised by other routers before they are added or updated in the local routing protocol database. Route update filters have only effect on distance vector protocols [1]. In our model, a routing update filter eliminates some adjacencies in an adjacency matrix. Considering the example network. If there is a routing update filter from R_1 to R_2 that denies to propagate information about destination S_2 then $Adj_{S_2}^P$ will have no adjacency from R_2 to R_1. In other words, the adjacency is removed in the opposite order to the effect of routing update filter.

In real world scenario not all networks are filtered or a single filter affects multiple networks. Therefore, we can lower the overall computational complexity by collecting all networks that are treated in the same way, which can be again implemented by grouping networks that have the same adjacency matrix.

Redistribution and Route Selection. The redistribution mechanism is vendor dependend, but most platforms obey two additional rules when doing redistribution: i) The route can only be redistributed if it is installed in routers FIB. ii) Even if a route is redistributed in the routing process with lower AD this new route is not installed into

routers FIB. These two rules cause that redistribution is not transitive as pointed out by Le, Xie and Zhang in [15], which requires some additional steps in the reachability computation.

Redistribution configure d on router R_v, which redistributes routing information from RIB_s to RIB_t causes to insert paths at row v of matrix RIB_s to corresponding items in matrix RIB_t if these paths have lower costs and can be found in the network FIB. The impact of redistribution rules is best observable in the process of route selection. The route selection finds the most appropriate information from local RIBs and puts it in the routers FIB. Redistribution then must check whether the information that can be redistributed was selected for FIB to conform with rule i). To satisfy rule ii) the route selection must not choose redistributed network into the FIB.

Among all information in all local RIBs, the router needs to select a single (or a collection of alternate paths for load balancing) route to its FIB. This process is vendor dependent but most often the routing information is prioritized by using administrative distance (AD) measure.

A computation of the network FIB means to implement an abstract route selection process. Informally, the computation proceeds as follows:

1. Take the lowest priority network RIB and copy all information to the network FIB. This will initialize the network FIB.
2. Take a RIB with immediately higher priority and replace paths in the FIB with existing paths in this RIB. This corresponds to the selection of route information with less administrative distance. If there is not path in this RIB then the path from a lower priority RIB remains in the FIB.
3. For each path in the RIB we must also check if this path can replace a suffix in an existing path in the FIB. This means, that if the FIB contains a path $\langle r1, r4, r2, r5, r3\rangle$ and the RIB contains a path $\langle r4, r7, r8, r9, r3\rangle$ we should replace the FIBs path with $\langle r1, r2, r4, r7, r8, r9, r3\rangle$. This replacement corresponds to installation of a route with lower AD in athe router on the existing path.
4. Repeat from step 2 until we process all RIBs.

Information stored in the network FIB (network reachability matrix) can be used to determine paths to all destinations. Forwarding state is uniquely determined by content of the network FIB.

5 Representation of Packet Filters

In this section, we review an existing methods to packet filter representation and develop a simple but efficient method for capturing the semantics of packet filters that is suitable for our network design validation method.

5.1 List-Based Packet Filters

A list-based packet filter (firewalls) consists of rules imposing network security policies ordered by the priority, which depends on the rule's position within the list. Although there may be other kinds of packet filters, we assume the most common case, in which

evaluation of packet filters is based on the first match principle. This means that an action of the highest priority rule that matches the analyzed packet is executed.

Each rule is a multidimensional structure. Dimensions are sets of network fields, e.g., source and destination adresses, port numbers, protocol type, or an action field, e.g., accept, deny, redirection. A typical rule can be formally defined as a tuple $\langle src, dst, srv, act \rangle$, where *src* and *dst* are set of addresses, *srv* is a set of services, and *act* is an action.

Packet filters can suffer from conflicts and dependencies, which complicate its analysis. The goal of packet filter preprocessing is to remove conflicts, redundancies and dependencies such that we avoid the need to evaluate the rules in the imposed order. If the resulting rule set is completely disjoint then it is possible to use a straightforward transformation into logical representation as will be shown at the end of this section. First, we discuss a method to obtain the disjoint rule set from an ordinary rule set.

We discuss the method proposed in [18]. The method consists of two steps. The first step isolates the possibly conflicting rules and figures out their dependencies. Two rules are potentially conflicting, if both rules have different actions and one rule is either subsumed by another or there is an nonempty intersection in one or more dimensions. The result of the first step is a conflicting graph.

The second step analyses the conflicting graph to identify a minimal set of rules that generate inconsistencies with another rules. The result is a collection of two level trees. Each root and leaf pair of trees defines a conflict that needs to be properly characterized to propose a modification that would remove the conflict making these rules independent.

The commonn approach to make the rule independent (disjoint) is to split conflicting rules into more rules and remove conflicting parts. This phase may be followed by merging process in order to optimize the rule set representation as the previous splitting may unecessary increase the rule set size. Note that the similar approach is taken also for redundancy elimination as shown in [2].

5.2 Logical Representations of Rule Sets

Semantics of rule set consisting only of independent rules is invariant to rule ordering. We will use this property to define for each rule set its logical representation. This representation has form of formula of propositional logic in disjunctive normal form.

Recall that filtering rule is a tuple with network fields. In the simplest case it consists of selectors, namely, source address set, destination address set, service set, and the action. A logical formula that is a translation of a simple rule $r = \langle s, d, v, a \rangle$ consists of a conjunction of all selectors. A selector is represented by a predicate that extracts required network fields from some packet p. thus, for rule r the formula is written as follows:

$$src\_adr(p) \in s \wedge dst\_adr(p) \in d \wedge service(p) \in v.$$

A list of possible selector functions in shown in Table 1

We adapt a network-mask convention for representation of a set of continuous addresses. For instance, it allows us to consider 147.229.12.0/24 as a set of addresses ranging from 147.229.12.0 to 147.229.12.255. We can use the standard set operations, e.g.,

Table 1. Network Field Selectors

Function	Description
$dst\_adr(p)$	Destination adress of a packet p.
$src\_adr(p)$	Source address of a packet p.
$dst\_port(p)$	Destination port of udp or tcp datagram carried in packet p.
$src\_port(p)$	Source port of udp or tcp datagram carried in packet p.
$service(p)$	Service of a packet p.

$src\_adr(p) \in 147.229.12.0/24$ or $dst\_adr(p) \in 147.12.28.0/24 \cup 147.12.30.0/24$. The latter can be expanded to $dst\_adr(p) \in 147.12.28.0/24 \vee dst\_adr(p) \in 147.12.30.0/24$, which allows us to use network-mask format for canonical address set representation.

Often, rule sets implicitly assume the existence of a default rule, which has the lowes priority and matches all packets not hitting by the other rules. While the previously described methods cope with default rules transparently, it may cause to split the default rule in a large number of rules appearing in a disjoint rule set. To overcome this issue we ignore the default rule in the process of conflicting rule elimination and consider it again when we compute a logical representation.

A logical representation can be either *positive*, representing all accepted packets or *negative* representing all denied packets. The most commonly, we want to compute a positive representation of rule sets with default deny all rule. In this case we only select all rules with *allow* action and calculate the logical representation for these rules. It completely eliminates the need to take care of the default rule.

6 Analysis of Security Policies

Packet filters implement basic level of security policies in the network. By restricting the accessibility of certain services, computers or subnetworks, we deploy rough but efficient security measures.

Our network model deals only with IP addresses and services or ports. Therefore, the analysis does not reflect hardware or OS attacks. We also don't examine the contents of TCP/UDP packets although it is possible to extend the description to support this. Our primary goal is verify safety or resistance of the network with respect to the effect of dynamic routing. Therefore, our classification includes only basic categories of network security properties. Since it can utilize typical fields from IP, TCP, or UDP headers, namely source/destination IP address and service/port [24], it allows to specify wide range of different communications to be analyzed in the network.

6.1 Reachability Sets

As proposed in [26] the output of reachability analysis and thus the input for consecutive security properties analysis consists of a collection of reachability sets for forwarding paths in an analyzed network. There are various methods to calculate reachability sets. In the following we discuss several issues related to this calculation. We first overview the problem of efficient address representation and rule sets representation. Then, we

give an idea of security properties verification and its position with respect to routing analysis and reachability analysis.

Address Scheme. An approach invented by Guttman in [12] deals with *abstract address* scheme. An abstract address is the symbolic name of a host or a subnetwork, which avoids the need to explicitly deal with a huge ip address space, which consists of 2^{32} all unique addresses. An abstract packet consists of an abstract source address, an abstract destination address, a service identification, and a flow orientation, which is either client to server, or server to client. This approach leads to very reasonable complexity in representation depending on the size of a network and mainly on the number of interesting destinations and services. For instance, considering a network with N different distinguished addresses, S different distinguished services, then we get an abstract packet space of size $N^2 \cdot 2S$.

A different approach is to explicitly represent the IP address space by bit variables each for a single bit in the address as used, for instance, by Bera, Ghosh and Dasgupta in [6]. This means that a packet is defined by bit variables $s_1, \ldots, s_{32}$ representing a source address, bit variables $d_1, \ldots d_{32}$ representing a destination address, and a vector of bit variables, $v_1, \ldots, v_n$ of the appropriate length n, representing a service. A flow direction may be modeled separately by a single bit variable or encoded in the service vector. In this way, we have an explicit representation not only for each packet but also for each network set represented in network-mask format.

Rule Sets Representation. Independently on whether we use abstract address representation or explicit representation, we construct logical formula for each rule set as discussed at the end of section 5. Each such formula can be encoded as a SAT instance using the boolean reduction approach, which is defined in detail for explicit address scheme in [7]. If we use the abstract address scheme we model each abstract address by a single boolean variable.

These two approaches thus differ by the number of boolean variables of generated SAT instances. While explicit representation requires the fixed number of variables, the number of variables used by abstract approach depends on the number of abstract addresses. On the other hand, the former may generate a large number of clauses while the latter tends to keep number of clauses smaller. It remains for future work to analyze and compare both approaches from the practical perspective on real scale data.

Verification of Security Policies. Capabilities of used network abstract model allows us to verify security properties that can be expressed in terms of service reachability. Contrary to most of the related work and in the similar way as Xie et al. in [27], we consider a tighter approximation of reachability sets obtained from estimation of routing in a given forwarding state.

In general, our verification method stands for checking whether a service is guaranteed under some assumption on connectivity. This assumption is converted to forwarding state by the method described in section 4. Each forwarding state consists of a set of active forwarding paths encoded in a path reachability matrix (denoted as network FIB). For any forwarding state it is possible to perform reachability analysis to determine the conditions leading to security violations.

In general, the considered analysis method can determine for every path the reachability for the given packet definition. Based on this, it is, for instance, possible:

- to perform security policy verification as proposed by Gutmann [12].
- to verify the access control based security implementations in a similar way as done by Bera, Ghosh and Dasgupta in [6], and
- to compute reachability estimations as shown by Xie at al. in [26].

In general, computing upper and lower bound, which means to consider all paths or finding at least a single path satisfying reachability condition, enables to verify the security policy implementation. However, considering routing effect allows us to find a particular network state in which security policy violation is detected. This gives a network designer refined information that aids him to find the configuration problem. Integration of routing model to the analysis process represents the main contribution of the presented work.

7 Summary

In this paper, we demonstrate the problem of automatic security analysis of IP based computer networks. The presented verification method aims at validating network design against the absence of security and configuration flaws. The network model allows describing effects of static and dynamic routing and access control lists configured on the network devices. The verification technique is based on the encoding problem into SAT instance solved by automatized solvers. Discussed procedure aims at refining of the approximation models proposed in [27],[6], [13], [3], and [16] to include routing effects, which would allow network designers to get an insight to issue of interweaving routing and filtering, and the impact on network security properties. Although not validated on real scale cases, we believe the application of the presented approach is feasible for a large class of network models and properties. Further work is aimed at refining the method to experimental implementation and performing experiments to evaluate its performance and scalability.

Acknowledgements. This project has been carried out with a financial support from the Czech Republic state budget through the CEZ MMT project no. MSM0021630528: Security-Oriented Research in Information Technology, by the Grant Agency of the Czech Republic through the grant no. GACR 102/08/1429: Safety and Security of Networked Embedded System Applications, and by the Brno University of Technology, Faculty of Information Technology through the specific research grant no. FIT-10-S-1: Secured, Reliable and Adaptive Computer Systems. Also, the first co-author was supported by the grant no. FR-TI1/037 of Ministry of Industry and Trade: Automatic Attack Processing.

References

1. Filtering routing updates on distance vector ip routing protocols. Cisco Systems, Document ID:9105 (September 2006)
2. Acharya, S., Wang, J., Ge, Z., Znati, T., Greenberg, A.: Simulation study of firewalls to aid improved performance. In: 39th Annual Simulation Symposium, 2006, p. 8 (2006)

3. Burns, J., et al.: Automatic management of network security policy. In: DARPA Information Survivability Conference and Exposition, pp. 1012–1026 (2001)
4. Bartal, Y., Mayer, A., Nissim, K., Wool, A.: Firmato: A Novel Firewall Management Toolkit. In: IEEE Symposium on Security and Privacy, pp. 17–31 (1999), citeseer.ist.psu.edu/article/bartal99firmato.html
5. Benson, T., Akella, A.: Unraveling the complexity of network management. In: NSDI 2009: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (2009), http://portal.acm.org/citation.cfm?id=1559000
6. Bera, P., Ghosh, S., Dasgupta, P.: Formal analysis of security policy implementations in enterprise networks. International Journal of Computer Networks and Communications 1(2), 56–73 (2009)
7. Bera, P., Ghosh, S., Dasgupta, P.: Formal Verification of Security Policy Implementations in Enterprise Networks. In: Prakash, A., Sen Gupta, I. (eds.) ICISS 2009. LNCS, vol. 5905, pp. 117–131. Springer, Heidelberg (2009)
8. Christiansen, M., Fleury, E.: An Interval Decision Diagram Based Firewall. In: 3rd International Conference on Networking (ICN 2004). IEEE (February 2004)
9. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
10. Gouda, M., Liu, A.X., Jafry, M.: Verification of distributed firewalls. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM), New Oreleans, Louisiana (November 2008)
11. Gross, L., Yellen, J. (eds.): Handbook of Graph Theory. CRC Press (2004)
12. Guttman, J.D.: Filtering postures: Local enforcement for global policies. In: Proceedings of 1997 IEEE Symposium on Security and Privacy, pp. 120–129. IEEE Computer Society Press (1997)
13. Guttman, J.D.: Filtering postures: Local enforcement for global policies. In: IEEE Symposium on Security and Privacy, pp. 120–129 (1997)
14. Jeffrey, A., Samak, T.: Model checking firewall policy configurations. In: IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 60–67 (2009)
15. Le, F., Xie, G., Zhang, H.: Understanding route redistribution. In: IEEE International Conference on Network Protocols, ICNP 2007, pp. 81–92 (2007)
16. Liu, A.X.: Formal verification of firewall policies. In: Proceedings of the 2008 IEEE International Conference on Communications (ICC), Beijing, China (May 2008)
17. Mitre: Common Vulnerabilities and Exposures Database, http://cve.mitre.org/ (accessed on February 2008)
18. Pozo, S., Ceballos, R., Gasca, R.: Fast algorithms for consistency-based diagnosis of firewalls rule sets. In: Proceedings of the 3rd International Conference on Availability, Reliability and Security, ARES (2008)
19. Ritchey, R.W., Ammann, P.: Using model checking to analyze network vulnerabilities. In: IEEE Symposium on Security and Privacy, Washington, USA (2000)
20. Shahriari, H.R., Sadoddin, R., Jalili, R., Zakeri, R., Omidian, A.R.: Network Vulnerability Analysis through Vulnerability Take-Grant Model (VTG). In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 256–268. Springer, Heidelberg (2005), citeseer.ist.psu.edu/749214.html
21. de Silva, G., Sveda, M., Matousek, P., Rysavy, O.: Formal analysis approach on networks with dynamic behaviours. In: Proceeding of the 2nd International Workshop on Reliable Networks Design and Modeling (2010)
22. Sveda, M., Rysavy, O., Matousek, P., Rab, J., Cejka, R.: Security analysis of tcp/ip networks – an approach to automatic analysis of network security properties. In: Proceedings of the International Conference on Data Communication Networking 2010, pp. 5–11. Institute for Systems and Technologies of Information, Control and Communication (2010)

23. Tidwell, T., Larson, R., Fitch, K., Hale, J.: Modeling Internet attacks. In: Proc. of the IEEE Workshop on Information Assurance and Security, West Point, NY (2001)
24. Čejka, R., Matoušek, P., Ráb, J., Ryšavý, O., Švéda, M.: A formal approach to network security analysis. Tech. rep. (2008), http://www.fit.vutbr.cz/research/view_pub.php?id=8572
25. Wool, A.: A quantitative study of firewall configuration errors. Computer 37, 62–67 (2004)
26. Xie, G.G., Zhan, J., Maltz, D.A., Zhang, H., Greenberg, A., Hjalmtysson, G., Rexford, J.: On static reachability analysis of ip networks. In: Proc. IEEE INFOCOM (2005)
27. Xie, G., Zhan, J., Maltz, D., Zhang, H., Greenberg, A., Hjalmtysson, G., Rexford, J.: On static reachability analysis of ip networks. In: INFOCOM, pp. 2170–2183 (2005)
28. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: A logic-based network security analyzer. In: Proc. of the 14th USENIX Security Symposium, Baltimore (2005), citeseer.ist.psu.edu/article/bartal99firmato.html
29. Yuan, L., Chen, H.: Fireman: a toolkit for firewall modeling and analysis. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 199–213 (2006)
30. Zakeri, R., Shahriari, H., Jalili, R., Sadoddin, R.: Modeling TCP/IP Networks Topology for Network Vulnerability Analysis. In: 2nd Int. Symposium of Telecommunications, pp. 653–658 (2005), citeseer.ist.psu.edu/749214.html