A Formal Approach to Network Security Analysis

Petr Matoušek, Ondřej Ryšavý, Jaroslav Ráb, Miroslav Švéda, Rudolf Čejka

Brno University of Technology, Faculty of Information Technology

Božetěchova 2, 612 66 Brno, Czech Republic

{matousp,rysavy,rabj,sveda,cejkar}@fit.vutbr.cz

Abstract

This paper deals with an approach to security analysis of TCP/IP-based computer networks. The method developed stems from a formal model of network topology with changing link states, and deploys bounded model checking of network security properties supported by SAT-based decision procedure. Its implementation consists of a set of tools that provide automatic analysis of router configurations, network topologies, and states with respect to checked properties. While the paper aims at supporting a real practice, its form strives to be exact enough to explain the principles of the method in detail.

1 Introduction

1.1 Motivation

Suppose a small organization running a web server that provides information to their customers. The server is placed in the local network equipped with three routers. A path to the Web server goes through router R2 that filters traffic by ACL1 (Access Control List) in its input, see Figure 1.

There is a backup line between routers R_1 and R_3 with higher costs (lower priority). However, when the link between R_2 and R_3 goes down, the traffic is not filtered any more

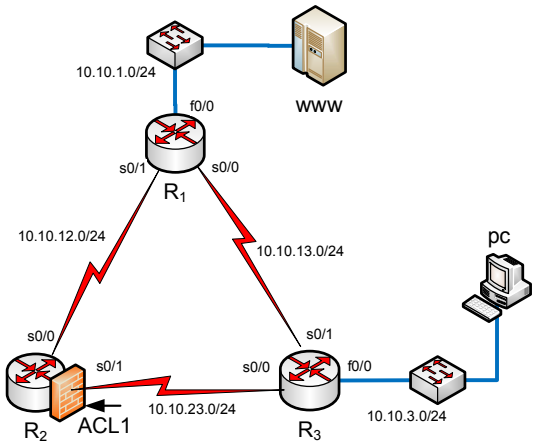


Figure 1: Example of a small network

and the web server can be attacked from the outside network.

In another scenario, the priority line appears between routers R1 and R3. This line enables to access the Web site from the PC, see Figure 1. When the link goes down, traffic is redirected by routing tables trough R2. However, R2 entry interface is filtered by ACL1. The connection from PC to Web server is filtered out and the Web services are no longer available.

These two scenarios present typical situation of a real-world network with dynamic behaviour. This paper shows an approach how to deal with such situation in order to ensure security, safety and availability of network resources.

Our approach 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 behaviour of the network, we can ask questions like "Is this network protected against P2P connections?", "What packets can be delivered to the given host?", or "Is this WWW service accessible under every configuration of the network?".

These questions can be partially answered by scanning and testing tools (ping, nmap), or vulnerability assessment tools (Nessus). However, testing can analyse the network only in immediate state, that means, for fixed configuration in practice. When the topology changes, the response of the network can be different as shown in the example above. In our work we explore how security and safety properties can be verified under every

network configuration using model checking [5]. The model checking 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 ACLs. The specification of network model is given in section 2. The network security properties are expressed in the form of modal logics formulas as constraints over states and execution paths. If a property is not satisfied, the model checker generates a counterexample that reveals a state of the network that violates the property. If the property is proved, it means, that the property is valid in every state of the systems.

In this paper we primarily focus on the automatic analysis of network security properties of the network. The challenges that we are addressing in the paper include: (a) automatic generation of the network model using routers configuration files, (b) creating templates for specification of network security properties, and (c) a combination of tools that verify given properties over the model using model checking.

1.2 State of the Art

In this section, we survey common approaches to network analysis with focus on vulnerability analysis and network security.

Dependability [14] is that property of a system that allows reliance to be justifiably placed on the service it delivers. Dependability measures consist namely of reliability, availability, security, safety and survivability. Availability is the ability to deliver shared service under given conditions for a given time, which means namely elimination of denial-of-service vulnerabilities. Security [7] is the ability to deliver service under given conditions without unauthorized disclosure or alteration of sensitive information. It includes privacy as assurances about disclosure and authenticity of senders and recipients. Security attributes add requirements to detect and avoid intentional faults. Safety [10] is the ability to deliver service under given conditions with no catastrophic affects. Safety attributes add requirements to detect and avoid catastrophic failures.

A failure occurs when the delivered service deviates from the specified service. The

failure occurred because the system was erroneous: an error is that part of the system state which is liable to lead to failure. The cause of an error is a fault. Failures can be classified according to consequences upon the environment of the system. While for benign failures the consequences are of the same order of magnitude (e.g. cost) as those of the service delivered in the absence of failure, for malign or catastrophic failures the consequences are not comparable. A mishap is an unplanned event (e.g. failure or deliberate violation of maintenance procedures) or series of events that results in damage to or loss of property or equipment. A hazard is a set of conditions within a state from which there is a path to a mishap.

Obviously, design of any safe system requires deploying security to avoid intentional catastrophic failures. And vice versa, system's security can be attacked using a safety flaw. The greater the assurance, the greater the confidence that a security system will protect against threads, with an acceptable level of risk. The above statement deals with trust, which is assured reliance on the character, ability, strength, or truth of someone or something [11]. In frame of network systems, trust is a complex subject that should be managed. Trust management entails collecting the information necessary to establish a trust relationship, and dynamically monitoring and adjusting the existing trust relationship.

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 [20], [23], or [18]. Most works follow the similar scheme: (i) Network is modelled as an entity that includes hosts, connections, user privileges, OS types, running services, and individual vulnerabilities of hosts. (ii) Host vulnerabilities are revealed by external automatic tools like Nessus, or by OVAL scanner [22]. Then, detected vulnerabilities are expressed in the language of precondition and postcondition assertions, or rules. (iii) An important step is to determine the attacker goal – security violation (e.g., root access on the web server). The goal is often expressed by a predicate. (iv) Having these, vulnerability analysis follows. It includes an application of derivation rules based on the initial assumptions (i.e., network

configuration) in order to prove a predicate (i.e., security violation). If the predicate is true then the deduction path corresponds to the possible attack scenario.

Despite the statement of authors in [18] that "this model lets *automatically* verify and proof network safety and vulnerability against the attack" (emphasis added), the method of logic deduction and proving requires good knowledge of logics and deductive systems, since the proof is constructive and it is made by human.

In [22], 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 OVAL scanner and CVE database [15].

Another approach is an automatic generation of network protection in the form of firewall rules as shown in [2]. The security policy is modelled 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 Amman [17] shows how model checking can be used to analyse network vulnerabilities. They build a network security model of hosts, connections, an attacker and exploits to be misused by the attacker. Security properties are described by temporal logics and verified using 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.

From the approaches mentioned above we can take following conclusions that are important for network security analysis: (1) a model of a network includes specification of hosts, their configurations, network topology, description of vulnerabilities; (2) a list of host vulnerabilities and network threats can be downloaded from open databases, or specified manually; (3) analysis can be manual or automatic, based on deductive systems or by model checking, respectively; (4) results of the analysis can either show specific vulnerabilities that require intervention of an administrator, generate a new safe configuration for network devices, or prove that the property is valid under every condition of the network.

1.3 Contribution

Many papers in this area deal with static network configuration. If network configuration or topology changes, a model of the network has to be rebuilt. Our approach deals with networks with dynamic behaviour. Dynamics is modelled by routing protocols, e.g., RIP or OSPF. Our goal is to automatically verify network security properties in a network model. The network model is constructed accroding configuration files of network devices and the network topology.

Our approach is close to the work of G. Xie [21], and J. Burns [6]. Unlike these works we build a model that includes both static and dynamic behaviour, i.e. firewall rules and routing information, see [13]. In this model, the verification of reachability properties can be made. In comparison to Ritchey's work [17] we do not focus on hosts vulnerability and their resistance to attacks but on stability of services in dynamic networks.

Main contributions of this paper consists of (i) the creation of a network transition system that models dynamic behaviour of the network, (ii) the definition of security properties using modal logics, (iii) the algorithm for verification of specified properties using bounded model checking and SAT-solver, and (iv) the presentation of the idea on a small example.

1.4 Structure of the paper

The paper is composed of five basic parts. After introduction, we present a formal model of network topology in section 2. The model was originally published in our paper [13]. In this paper we describe in more detail how dynamic changes can be modelled using a network transition system where transitions reflect the changes of links in the original network (link down, up). Section 3 describes common security properties and the way these properties can be expressed by modal logics. Section 4 shows what security properties can be verified using model checking. SAT-solver. The section ends with the example explaining how network security properties are verified using the proposed approach. Last section summarizes the results and proposes future progress of network-wide security analysis.

2 Network Model

The aim of this section is to provide a formal model of a network topology that allows us to specify a set of attributes for security analysis. This model was originally defined and published in [13]. In this paper, we show a brief introduction to our graph-based formal network model with packet filter predicates classifying the message flow and thus constraining the reachability of the entities in the network. For the rest of the paper we refer to the example of the network topology as given in Figure 1.

2.1 Formal description of the network

One of the contributions of this paper is the formal model of the network from point of view of routing processes and filtering. Our approach combines techniques introduced in [21] and [4]. The network is modeled as a directed graph where vertices are routing devices and edges are communication channels that form abstraction of communication links. Each communication link is modeled by a pair of unidirectional communication channel.

Definition 1 (Network) A network *is a tuple* $N = \langle \mathcal{R}_N, \mathcal{L}_N, \mathcal{F}_N \rangle$, *where*

- $\mathcal{R}_N$ *is a finite set of network devices,*
- $\mathcal{L}_N \subseteq \mathcal{R}_N \times \mathcal{R}_N$ *is a finite set of links between routers, such that for every physical link between* R_1, R_2 *there is a pair of channels* $l_{12} = \langle R_1, R_2 \rangle$, $l_{21} = \langle R_2, R_1 \rangle$, *and*
- $\mathcal{F}_N = \{f : \mathcal{P} \rightarrow \{true, false\}\}$ *is a finite set of filtering predicates and* $\mathcal{P}$ *is a set of all possible packets.*

Because filters can be applied in both directions of the link, we suppose that the set $\mathcal{L}$ contains for every link two items $\langle R_i, R_j \rangle$ and $\langle R_j, R_i \rangle$. 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 default gateway. According to the previous definition the network model for our running example is a graph as shown in Figure 2.

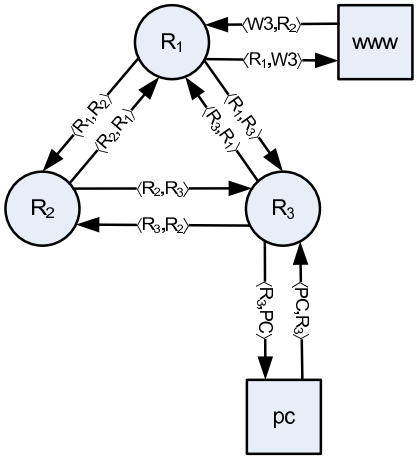


Figure 2: Model of Network N

End-point devices are distinguished from routers by different symbol only for the clarity of the presentation.

Geoffrey G.Xie et al. in [21] show 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. We analyze the network states in section 4 of this paper.

A filtering predicate $f(p) \in \mathcal{F}_N$ is able to determine whether a packet p is allowed to be send. This function is defined so that it uniformly represents the interpretation of Access Control List (ACL) and routing table information adequate to link l. A simple example is a filter, $f(p) = \neg(p.proto = Tcp \wedge p.dstPort = 80)$, that denies all web traffic, i.e. TCP packets with destination port 80. Note that dot notation is used to refer to attributes of the current packet. Both ACL and routing information of a network node can be translated to a filtering predicate.

A network link state s consists of a vector of boolean values representing condition of links in the network and a function that maps the state-specific filtering predicate to each link.

Definition 2 (Network Configuration State) *A set of network configuration states of a network N is a set $\mathcal{S}_N = \{\langle b, \delta_b \rangle | b \in 2^{|\mathcal{L}_N|}, \delta_b : \mathcal{L}_N \rightarrow \mathcal{F}_N\}$, where*

- *b is a link state vector, e.g. $b = (1, 1, 1)$. $b[i] = 1$, means the i-th link is up, 0 means it is down.*

- $\mathcal{F}$ *is a set of filtering predicates of network* N*, and*

- δ_b *is a mapping function that assigns a filtering predicate to each link of network* N*. If the link* l_i *is down then filtering predicate is false for every packet, i.e. for* $b[i] = 0$ *then* $\forall p \in \mathcal{P}.\delta_b(l_i)(p) = false$*.*

For the network N with $|\mathcal{L}_N|$ links its state vector consists of $|\mathcal{L}_N|$-bits. Number of different states is given by all possible combinations of link states, that is $2^{|\mathcal{L}_N|}$. Suppose for our running example from Fig.1, for brevity, only links between routers. We consider a network with three volatile links, represented by three pairs of connections in the corresponding network graph (see Fig.2). In this network, there are following network states: $b_8 = (1, 1, 1)$ (all links are up, network is fully connected), $b_5 = (1, 0, 1)$ (one link is down, in particular, link represented by pair $\langle R_2, R_3\rangle, \langle R_3, R_2\rangle$), etc. The number of network states is $2^3 = 8$.

Definition 3 (Network Transition System) *Behaviour of a network* $N = \langle \mathcal{R}, \mathcal{L}, \mathcal{F}\rangle$*, from point of view of topology changes, can be defined by* network transition system $\mathcal{T}_N = (\mathcal{S}_N, \{\stackrel{a}{\longrightarrow} | a \in \mathcal{A}\})$*, where*

- $\mathcal{S}_N$ *is a finite set of network states,*

- $\mathcal{A} \subseteq \{\{\uparrow \langle r_1, r_2\rangle\} \cup \{\downarrow \langle r_1, r_2\rangle\} | \langle r_1, r_2\rangle \in \mathcal{L}\}$ *is a non-empty set of actions, such that for each* $a \in \mathcal{A}$*,* $\stackrel{a}{\longrightarrow} \subseteq \mathcal{S}_N \times \mathcal{S}_N$*,*

- $\stackrel{a}{\longrightarrow}$ *is a transition relation between network states that reflects the change of one link going down or up. Formally, for link* l_k*:*

  a. $s_i \stackrel{a}{\longrightarrow} s_j, a = \downarrow l_k$ *if* $b_i[k] = 1, b_j[k] = 0$*, and* $b_i[n] = b_j[n], \forall n \neq k$*.*

  b. $s_i \stackrel{a}{\longrightarrow} s_j, a = \uparrow l_k$ *if* $b_i[k] = 0, b_j[k] = 1$*, and* $b_i[n] = b_j[n], \forall n \neq k$*.*

In Figure 3 a transitions system for the network model from Figure 2 is depicted.

By configuration, we mean the state of the network that comprises actual states of links, content of routing information bases, and filtering functions. Note that content of routing information bases depends on the actual states of links, i.e. active topology of the

network. Contrary, we can assume that filtering functions does not vary with the change of network state. A label $a \in \mathcal{A}$ is an information telling us that a link went down or up. For instance, $a = \downarrow \langle R_1, R_2 \rangle$ tells us that connection from R_1 to R_2 is no longer available.

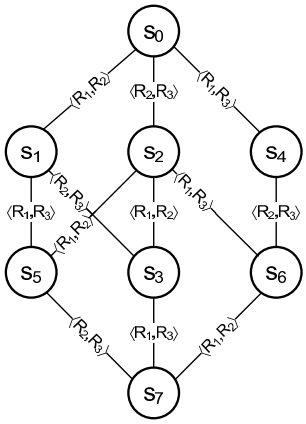


Figure 3: Example of Network Transition System

3 Classification of Security Properties

In the previous part of the paper, we introduced a formal model of the network. The model is automatically taken from configuration files of network devices – routers. In our research, we concentrate on Cisco IOS operating system and Cisco configuration files. However, the approach can be applied to any other configurations. In this paper, we work only with IP addressing (IP addresses, routing) and filtering rules. Other parameters and settings are abstracted away for the sack of simplicity.

Having a formal model of the network, we show in this section, how security properties can be formally expressed and then exploited for verification. Verification using model checking technique is explained in the following section.

Description of network security properties is related to the classification of threats and intrusion. There are plenty of different network security problems, like HTTP attacks, spams, TCP flooding, DoS attacks, Web server misuse, spoofing and sniffing etc. How to grasp the list of security problems in order to formalize security properties for the analysis? We decided to set several categories of network security problems that can be solved by our system.

There are several classifications of security intrusions, threats, vulnerabilities, or at-

Property	Definitions and examples (specified in modal logics[1])
availability	Ability of the system to provide service at any given time.
	Ex.: Any external node A has access to a mail server.
	$SMTP \in NetReach_{\varphi}(A)$, $\varphi = p.proto = Tcp \wedge p.dstPort = 25$
safety	Ability of the system to delivery service under given conditions.
	Ex.: Web server is accessible from A even if link L is down
	$WWW \in NetReach_{\psi}(A) \implies [\downarrow L]WWW \in NetReach_{\psi}(A)$, $\psi = p.proto = Tcp \wedge p.dstPort = 80$
security	Ability of the system to protect resources resources.
	Ex.: Payroll server is not accessible from any external hosts A
	$Payroll \notin NetReach_{\varphi}(A), \varphi = \neg(p.srcIp = local \wedge p.proto = IP)$

[1] syntax of modal logics and function *NetReach*() is explained in section 4 in details

Table 1: Classification of network security properties

tacks. In [8] we can find classification of abstract signatures. Neumann and Parker [16] propose nine categories of misuse techniques (external misuse, hardware misuse, masquerading, etc.). Lindquist and Jonsson [12] describe intrusion in two dimensions – intrusion technique and intrusion result. On-line databases of vulnerabilities have their own categories: CVE (Common Vulnerabilities and Exposures) [15] defines many categories like buffer errors, code injection, configuration. credentials, cross-site scripting, etc. Intrusion detection database Snort [1] classifies violation rules mostly with respect to applications – chat, nntp, mysql, pop, icmp, imap, web, etc.

Our network model deals with only IP addresses and services (ports). Therefore the analysis does not reflects hardware or OS attacks. We also don't examine the contents of TCP/UDP packets. Our primary goal is safety or resistance of the network with respect to dynamic behaviour of the network.

Our classification is based on the taxonomy systems mentioned above. It includes three basic categories of network security properties as described in Table 1. Predicate φ defines packet property that is verified on the model. Since it includes typical fields from IP, TCP, or UDP headers (source/destination IP address, service/port) [13] it allows us to specify wide range of different communications to be analyzed in the network.

4 Model-Checking of Security Properties

In this section, we define a framework suitable for implemention of a model-checking algorithm based on the intepretation of a dynamic network as a state system with transitions induced changes of link conditions. The employed query language is propositional modal logic [19]. For the computation of the model and evaluation of properties in this model we assume a representation of ACLs and routing rules in the form of filtering predicates $\mathcal{F}_N$ for network N.

We define state predicates that can be interpreted in each state of the model and state functions that can be evaluated in each state of the model. In our case, we use $NetReach_{\varphi}(R)$ function that determines a set of routers reachable from router R under packet property φ. Evaluating this function in network states s_1, s_2 can give different results because dynamic routing information varies with network topology.

We can put restrictions on the path between two routers, for example, we verify if there exists a path between two routers for Web traffic. This is called network path under packet property.

Definition 4 (Network Path under packet property) *A path under packet property φ between two routers $R, R' \in \mathcal{R}$ on the network $N = \langle \mathcal{R}, \mathcal{L}, \mathcal{F} \rangle$ in state $s = \langle b, \delta_b \rangle \in \mathcal{S}_N$ is a sequence of routers $r_i \in \mathcal{R}$, links $l_i \in \mathcal{L}$ with filters $F_i = \delta_b(l_i) \in \mathcal{F}$ as follows:*

$$\begin{aligned} \pi^s_{\varphi}(R, R') &= (R, \langle R, R_1 \rangle, R_1, \langle R_1, R_2 \rangle, R_2, \ldots, R_k, \langle R_k, R' \rangle, R') \\ \textit{where} &\quad F_{\langle R,R_1 \rangle}(p) \wedge \ldots \wedge F_{\langle R_k,R' \rangle}(p) \wedge \varphi(p) \textit{ for some packet } p \textit{ holds.} \end{aligned}$$

The definition above restricts the set of possible paths from R to R' to those paths where packet property φ is satisfied on every link of the path in network state s.

Definition 5 (Network Reachability under packet property) *Network Reachability under packet property φ on the network $N = \langle \mathcal{R}, \mathcal{L}, \mathcal{F} \rangle$ in network state s, $NetReach^s_{\varphi}(R)$,*

is a set of routers reachable from router R for packet satisfying property φ:

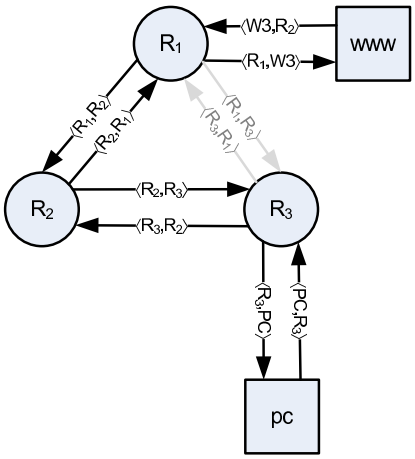
$$NetReach^{s}_{\varphi}(R) = \{R' \in \mathcal{R} \mid \exists \pi^{s}_{\varphi}(R, R_k), R_k = R'\}$$

Network reachability under packet property can be computed by a least fixed-point algorithm. We use a language of modal logic to express security properties. Modal logic allows us to reason with validity of packet properties (protocol = TCP, port = 80) in different network states. For example, a statement in modal logic describes properties in different network states where links can change their states, which cannot be expressed by basic propositional logic. In modal logic, a network property can be specified using box and diamond operators. Modalized formula $[a]\varphi$ is valid in network state s_1, $s_1 \models [a]\varphi$, if packet property φ is valid in network state s_2, $s_2 \models \varphi$, and $s_1 \xrightarrow{a} s_2$. The dual operator called diamond is defined as $\langle a \rangle \varphi \stackrel{def}{=} \neg[a]\neg\varphi$. A language of modal logic enables us to specify and verify various network properties. For instance, network property ψ says that the problem with link between routers R_1 and R_2 has no influence on the web traffic between host PC_1 and web server WWW.

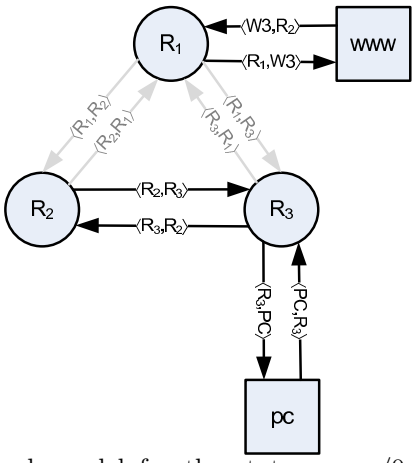
$$\psi \stackrel{def}{=} WWW \in NetReach_{\varphi}(PC) \implies [\downarrow \langle R_1, R_2 \rangle] WWW \in NetReach_{\varphi}(PC)$$
$$\varphi \stackrel{def}{=} proto = Tcp \wedge dstPort = 80$$

Figure 4 shows a case in which property ψ obviously does not hold.

A formula of modal language is interpreted in network transition system $\mathcal{T}_N$. For non-modal fragments we need to interpret atomic sentences. For model checking we define a modal model on the network transition system $\mathcal{T}_N$ as a pair $\mathcal{M}_N = \langle \mathcal{T}_N, \mathcal{V}_N \rangle$ where $\mathcal{V}_N$ is a valuation which assigns to each atomic sentence Q a subset of states of $\mathcal{T}_N$. Truth at state s of an arbitrary formula ψ under $\mathcal{M}_N$ is inductively defined using the notation



Network model for the state $s_3 = \langle 0, 1, 1 \rangle$. Property $WWW \in NetReach_{\varphi}(PC)$ holds in this state because of path via router R_2.



Network model for the state $s_2 = \langle 0, 1, 0 \rangle$. Property $WWW \in NetReach_{\varphi}(PC)$ does not hold in this state as there is not path between the two end devices.

Figure 4: Counterexample of property

$s \models_{\mathcal{M}} \psi$.

$$
\begin{aligned}
s &\models_{\mathcal{M}_N} Q && \text{iff} \quad s \in \mathcal{V}_N(Q) \\
s &\models_{\mathcal{M}_N} \neg\psi && \text{iff} \quad s \not\models_{\mathcal{M}_N} \psi \\
s &\models_{\mathcal{M}_N} \psi_1 \wedge \psi_2 && \text{iff} \quad s \models_{\mathcal{M}_N} \psi_1 \text{ and } s \models_{\mathcal{M}_N} \psi_2 \\
s &\models_{\mathcal{M}_N} [a]\psi && \text{iff} \quad \forall s'.\text{if } s \xrightarrow{s} s' \text{ then } s' \models_{\mathcal{M}_N} \psi
\end{aligned}
$$

For simplification, we consider $Q = R \in NetReach_{\varphi}(R')$.

4.1 Decision procedure

In this section, we aim to explain how to construct a general decision procedure for modal model in the realm of the network transition system $\mathcal{T}_N$. The small modal property of the logic guarantees the decidability of the procedure that tests the satisfiability of a formula [**?**]. We adopted bounded model checking [3] that limits state space by reachability diameter. In our case, the reachability diameter equals to the number of links. In the rest of this section, the translation of the model-checking problem to SAT based problem is shown.

We have modal-model $\mathcal{M}_N$ and modal-formula ψ. The bound k is given by number of links. As shown above, we are able to evaluate expression $R \in NetReach^{s}_{\varphi}(R')$ in every

state s. SAT-based decision procedure evaluates propositional formula $[\![\mathcal{M}_N, \psi]\!]_k$. This formula consists of a propositional representation of transition system $\mathcal{T}_N$ and network property ψ:

$$[\![\mathcal{M}_N, \psi]\!]_k \stackrel{def}{=} [\![\mathcal{M}_N]\!]_k \wedge [\![\psi]\!]_k^0$$

The interpretation of the formula is as follows:

$$\begin{array}{lcl}
[\![\mathcal{M}_N]\!]_k & \stackrel{def}{=} & \bigwedge_{i=0}^{k-1}(s_i \stackrel{a}{\longrightarrow} s_j) \\
[\![Q]\!]_k^i & \stackrel{def}{=} & s_i \in \mathcal{V}(Q) \\
[\![\neg Q]\!]_k^i & \stackrel{def}{=} & s_i \notin \mathcal{V}(Q) \\
[\![\varphi \wedge \psi]\!]_k^i & \stackrel{def}{=} & [\![\varphi]\!]_k^i \wedge [\![\psi]\!]_k^i \\
[\![[a]\varphi]\!]_k^i & \stackrel{def}{=} & (s_i \stackrel{a}{\longrightarrow} s_j) \implies [\![\varphi]\!]_k^j
\end{array}$$

4.2 Example

Assume the network model drawn in Figure 2, network transition system shown in Figure 3 and routing rules that generate filtering predicates as listed in Figure 5. Then the decision procedure that evaluates property ψ as defined in this section works as follow.

Using Kleene's Algorithm (see, for instance, [9, p.66]), the reachability matrix for the network at any given state is computed.

b	WWW	R_1	R_2	R_3	PC
WWW	$\top$	$\top$	$f_{1,2} \vee (f_{1,3} \wedge f_{3,2})$	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$
R_1	$\top$	$\top$	$f_{1,2} \vee (f_{1,3} \wedge f_{3,2})$	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$
R_2	$f_{1,2} \vee (f_{1,3} \wedge f_{3,2})$	$f_{1,2} \vee (f_{1,3} \wedge f_{3,2})$	$\top$	$f_{2,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{2,3} \vee (f_{1,2} \wedge f_{2,3})$
R_3	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{3,2} \vee (f_{3,1} \wedge f_{1,2})$	$\top$	$\top$
PC	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{1,3} \vee (f_{1,2} \wedge f_{2,3})$	$f_{3,2} \vee (f_{3,1} \wedge f_{1,2})$	$\top$	$\top$

Each cell of the reachability matrix defines a formula consisting of filtering predicates. The formula specifies overall filtering predicate for the whole path. Therefore, proposition $WWW \in NetReach_{\varphi}(PC)$ is translated to $(f_{1,3} \vee (f_{1,2} \wedge f_{2,3})) \wedge \varphi$. The network transition system is translated into the following propositional formula:

$$M \stackrel{def}{=} s_7 \stackrel{\downarrow\langle R_1,R_2\rangle}{\longrightarrow} s_6 \wedge s_7 \stackrel{\downarrow\langle R_1,R_3\rangle}{\longrightarrow} s_3 \wedge \ldots s_1 \stackrel{\downarrow\langle R_1,R_2\rangle}{\longrightarrow} s_0 \wedge s_4 \stackrel{\downarrow\langle R_1,R_3\rangle}{\longrightarrow} s_0$$

Mapping function δ assigns to each link a label as defined in the following table.

δ	R_1	R_2	R_3
R_1	$\perp$	$f_{1,2}$	$f_{1,3}$
R_2	$f_{2,1}$	$\perp$	$f_{2,3}$
R_3	$f_{3,1}$	$f_{3,2}$	$\perp$

Routing tables of devices are altered with each topology change. The following table shows a dynamic content of routing tables for selected network states.

state	R_1	R_2	R_3
$\langle 1,1,1\rangle$	$10.10.3.0/24 \rightarrow R_3$ $10.10.23.0/24 \rightarrow R_3$ $10.10.23.0/24 \rightarrow R_2$	$10.10.3.0/24 \rightarrow R_3$ $10.10.13.0/24 \rightarrow R_3$ $10.10.13.0/24 \rightarrow R_1$ $10.10.1.0/24 \rightarrow R_1$	$10.10.1.0/24 \rightarrow R_1$ $10.10.12.0/24 \rightarrow R_1$ $10.10.12.0/24 \rightarrow R_2$
$\langle 1,1,0\rangle$	$10.10.3.0/24 \rightarrow R_3$ $10.10.23.0/24 \rightarrow R_3$	$10.10.3.0/24 \rightarrow R_3$ $10.10.13.0/24 \rightarrow R_3$ $10.10.1.0/24 \rightarrow R_3$	$10.10.1.0/24 \rightarrow R_1$
$\langle 0,1,1\rangle$	$10.10.3.0/24 \rightarrow R_2$ $10.10.23.0/24 \rightarrow R_3$ $10.10.23.0/24 \rightarrow R_2$	$10.10.3.0/24 \rightarrow R_3$ $10.10.1.0/24 \rightarrow R_1$	$10.10.1.0/24 \rightarrow R_2$ $10.10.12.0/24 \rightarrow R_2$

Mapping δ_b for some state b defines the assignment of filtering predicates to labels f_i assigned to links of the network as listed bellow:

$b = \langle 1,1,1\rangle$ (All links are up)
$f_{1,2} = dstIp \in 10.10.12.0/24 \vee dstIp \in 10.10.23.0/24$ $f_{1,3} = dstIp \in 10.10.13.0/24 \vee dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.3.0/24$ $f_{2,1} = dstIp \in 10.10.12.0/24 \vee dstIp \in 10.10.1.0/24 \vee dstIp \in 10.10.13.0/24$ $f_{2,3} = dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.3.0/24 \vee dstIp \in 10.10.13.0/24$ $f_{3,1} = dstIp \in 10.10.13.0/24 \vee dstIp \in 10.10.12.0/24 \vee dstIp \in 10.10.1.0/24$ $f_{3,2} = dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.12.0/24$
$b = \langle 1,1,0\rangle$ (Link between R_1 and R_2 is down)
$f_{1,2} = \perp$ $f_{1,3} = dstIp \in 10.10.13.0/24 \vee dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.3.0/24$ $f_{2,1} = \perp$ $f_{2,3} = dstIp \in 10.10.1.0/24 \vee dstIp \in 10.10.3.0/24 \vee dstIp \in 10.10.13.0/24$ $f_{3,1} = dstIp \in 10.10.13.0/24 \vee dstIp \in 10.10.1.0/24$ $f_{3,2} = dstIp \in 10.10.23.0/24$
$b = \langle 0,1,1\rangle$ (Link between R_1 and R_3 is down)
$f_{1,2} = dstIp \in 10.10.12.0/24 \vee dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.3.0/24$ $f_{1,3} = \perp$ $f_{2,1} = dstIp \in 10.10.12.0/24 \vee dstIp \in 10.10.1.0/24$ $f_{2,3} = dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.3.0/24$ $f_{3,1} = \perp$ $f_{3,2} = dstIp \in 10.10.23.0/24 \vee dstIp \in 10.10.12.0/24 \vee dstIp \in 10.10.1.0/24$

Figure 5: Routing information and filtering for selected network states

Variable	Expression
v_1	$dstIp \in 10.10.1.0/24$
v_3	$dstIp \in 10.10.3.0/24$
v_{12}	$dstIp \in 10.10.12.0/24$
v_{13}	$dstIp \in 10.10.13.0/24$
v_{23}	$dstIp \in 10.10.23.0/24$
v_4	$proto = Tcp$
v_5	$dstPort = 80$

Variable	Transition
$t_{7,6}$	$s_7 \overset{\downarrow\langle R_1,R_2\rangle}{\longrightarrow} s_6$
$t_{7,3}$	$s_7 \overset{\downarrow\langle R_1,R_3\rangle}{\longrightarrow} s_3$
$t_{7,5}$	$s_7 \overset{\downarrow\langle R_2,R_3\rangle}{\longrightarrow} s_5$
$\vdots$	$\vdots$
$t_{4,0}$	$s_4 \overset{\downarrow\langle R_1,R_3\rangle}{\longrightarrow} s_0$

Figure 6: Assignment of Boolean variables to predicates

Next, the property ψ is translated into a propositional formula:

$$P_i^j \overset{def}{=} (f_{1,3}^{s_i} \vee (f_{1,2}^{s_i} \wedge f_{2,3}^{s_i})) \wedge \varphi^i \implies \left((s_i \overset{\downarrow\langle R_1,R_2\rangle}{\longrightarrow} s_j) \implies (f_{1,3}^{s_j} \vee (f_{1,2}^{s_j} \wedge f_{2,3}^{s_j})) \wedge \varphi^j\right)$$

The formula $M \wedge P_i^j$ for every $i, j \in \{0 \dots 7\}$ is a result of translation procedure. SAT solver takes a boolean proposition and attempts to find a valuation of the boolean variables such that the formula is satisfied. To allow application of SAT solver the predicates need to be represented as boolean variables. This is defined by tables in Figure 6. The propositional formula for the network model and a fragment of the formula specifying the property ψ are written as follows:

$$M' \overset{def}{=} t_{7,6} \wedge t_{7,3} \wedge t_{7,5} \wedge t_{5,1} \wedge t_{5,2} \wedge t_{3,1} \wedge t_{3,2} \wedge t_{6,2} \wedge t_{6,4} \wedge t_{1,0} \wedge t_{2,0} \wedge t_{4,0}$$

$$\begin{aligned} P'^3_7 \overset{def}{=} \; & \Big(\big((v_{13} \vee v_{23} \vee v_3) \vee (v_{12} \vee v_{23}) \wedge (v_{23} \vee v_3 \vee v_{13})\big) \wedge v_4 \wedge v_5\Big) \implies \\ & \Big(t_{7,3} \implies \big(((v_{12} \vee v_{23} \vee v_3) \wedge (v_{23} \vee v_3)) \wedge v_4 \wedge v_5\big)\Big) \end{aligned}$$

Finally, the formula $M' \wedge \bigwedge_{i,j=0}^{7} P'^j_i$ is evaluated by SAT procedure with result either true or false.

5 Conclusions

5.1 Summary

The presented verification method aims at validating network design against the absence of security and configuration flaws. The method does not require the deployment of the

network. On the other hand it is not capable of revealing security problems that are incurred by the presence of hardware errors or bugs in software. The network model allows us to describe the effect of static and dynamic routing and access control lists configured on the network devices. The verification technique based on the bounded model checking counts for varying link conditions checks whether a given property holds in the network model. It was shown that the method is able to deal with various class of properties. In particular, properties were classified into availability, safety and security classes. In all cases, a language of modal logic was used to express the property formally and served as an input to the model checking algorithm. Although we did not explicitly stated the complexity of the problem it can be ssen that the application of this technique is feasible for a large class of network models and properties.

5.2 Future Work

In this paper we demonstrated the problem of automatic security analysis of TCP/IP-based computer networks. It was shown that bounded model checking is a useful method in this area. The developed experimental tools provided reasonable data convincing us that the method is applicable in practice. Nevertheless, the experiments with real-size models are still in progress at the moment of writing this paper and further analysis is required to fully evaluate the method. There are also a lot of possible extensions to the method. First, the specification language is rather minimalist and it is challenging to show whether all important security properties can be specified in it. In the case of negative answer the further work should be oriented on refined classification of properties and proposing an adequate extension of the language and the verification method. Second, a lot can be done in the area of optimization of the method. It requires deeper understanding of the relation of dynamic routing protocols behavior to the network transition system for various network topologies. From the simple example provided it is evident that netowork states share much common information. Therefore it is possible to avoid recomputing all filtering predicates for each state. Finally, for conducting practical experiments it is necessary to implement reliable and effective tools.

References

[1] Snort network intrusion and prevention system. Available from http://www.snort.org/; accessed on Feb 2008.

[2] Y. Bartal, A.J. Mayer, K. Nissim, and A. Wool. Firmato: A Novel Firewall Management Toolkit. In *IEEE Symposium on Security and Privacy*, pages 17–31, 1999.

[3] A. Biere, A. Cinnatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 2003.

[4] M. Christiansen and E. Fleury. An Interval Decision Diagram Based Firewall. In *3rd International Conference on Networking (ICN'04)*. IEEE, February 2004.

[5] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.

[6] J. Burns et al. Automatic management of network security policy. In *DARPA Information Survivability Conference and Exposition*, pages 1012–1026, 2001.

[7] Il-Gon Kim and et. Formal Verification of Security Model Using SPR Tool. *Computers and Artificial Intelligence*, 25(5), 2006.

[8] S. Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue, IN, 1995.

[9] Jonathen l. Gross and Jay Yellen, editors. *Handbook of Graph Theory*. CRC Press, 2004.

[10] N.G. Leveson. Software Safety in Computer-Controlled Systems. *IEEE Computer*, pages 48–55, February 1984.

[11] H. Li and M. Singhal. Trust Management in Distributed Systems. *Computer*, 40(2):45–53, 2007.

[12] U. Lindqvist and E. Jonsson. How to Systematically Classify Computer Security Intrusions. In *IEEE Symposium on Security and Privacy*, Washington, 1997.

[13] Petr Matousek, Jaroslav Rab, Ondrej Rysavy, and Miroslav Sveda. A formal model for network-wide security analysis. In *In the 15th IEEE Symposium and Workshop on ECBS*, 2008.

[14] B. E. Melhart and S. White. Issues in Defining, Analyzing, Refining, and Specifying System Dependability Requirements. In *ECBS*, pages 334–340, 2000.

[15] Mitre. Common Vulnerabilities and Exposures Database. Available from http://cve.mitre.org/; accessed on Feb 2008.

[16] P.G. Neumann and D.B. Parker. A Summary of Computer Misuse Techniques. In *Proc. 12th National Computer Security Conference*, pages 396–407, 1989.

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

[18] H.R. Shahriari and R.Jalili. Modeling and Analyzing Network Vulnerabilities via a Logic-Based Approach. In *2nd Int. Symposium of Telecommunications*, pages 13–18, 2005.

[19] C. Stirling. *Modal and temporal logics*, pages 477–563. Oxford University Press, Inc., New York, NY, USA, 1992.

[20] T. Tidwell, R. Larson, K. Fitch, and J. Hale. Modeling Internet attacks. In *Proc. of the IEEE Workshop on Information Assurance and Security*, West Point, NY, 2001.

[21] Geoffrey G. Xie, Jibin Zhan, David A. Maltz, Hui Zhang, Albert G. Greenberg, Gísli Hjálmtýsson, and Jennifer Rexford. On static reachability analysis of ip networks. In *INFOCOM*, pages 2170–2183, 2005.

[22] X.Ou, S.Govindavajhala, and A.W.Appel. MulVAL: A logic-based network security analyzer. In *In Proc. of the 14th USENIX Security Symposium*, Baltimore, 2005.

[23] R. Zakeri, H.R. Shahriari, R.Jalili, and R.Sadoddin. Modeling TCP/IP Networks Topology for Network Vulnerability Analysis. In *2nd Int. Symposium of Telecommunications*, pages 653–658, 2005.