A Formal Framework for Network Security Design Synthesis

Mohammad Ashiqur Rahman and Ehab Al-Shaer
Department of Software and Information Systems, University of North Carolina at Charlotte, United States
Emails: {mrahman4, ealshaer}@uncc.edu

I. Introduction

Today, most of the organizations are not only emphasizing the enforcement of the security requirements but also requiring satisfaction of different business constraints on usability and security deployment cost. The organizational security requirements can be indicated by *isolation* measures between the hosts. The isolation patterns are defined based on different security devices and their capabilities. An isolation pattern signifies the type of security resistance, e.g., traffic filtering, trusted communication, payload traffic inspection, hiding traffic source identity, etc. However, any security design has to satisfy the *business constraints* of the organization, which are represented in terms of *usability* and *deployment cost*. The implementation of isolation measures significantly affects these constraints. For example, the use of the encrypted communication might reduce the usability, while the access deny would give no usability. Moreover, the deployment of a security device has a cost. Hence, it is required to find security configurations by exploring different security design alternatives that maintain the security and usability within an expected level and an affordable cost.

The research on the security design synthesis is in a premature stage. In the work [1], the authors presented procedural approaches of generating firewall configurations. However, this work does not relate security device placements with the topology and cannot do the optimal placements of security devices in the network. In this paper, we present *ConfigSynth*, an automated framework for synthesizing network security configurations and physical placements of security devices, using constraint satisfaction checking. This framework formulates the *security design synthesis problem* from the given network topology, security requirements, and business constraints. ConfigSynth is a novel framework that incorporates the security device placements in the network topology within the design in order to model the deployment cost, which is crucial for a security architecture.

II. Security Design Synthesis Model

ConfigSynth models the network topology as a graph. The network model is defined as $\langle \mathbb{N}, \mathbb{L} \rangle$, where,

- $\mathbb{N}$ defines a finite set of network nodes including hosts and routers. A host may execute one or more services, which are accessed by different hosts. A service is denoted using $g \in \mathbb{G}$, where $\mathbb{G}$ is the set of all services. The term $g(i, j)$ defines the flow between a pair of hosts $\{i, j\}$, where i is the source and j is the destination, under a service g.
- $\mathbb{L} \subseteq \mathbb{N} \times \mathbb{N}$ is a finite set of links, which defines the interconnections between the network hosts.

ConfigSynth formalizes different requirements and constraints which are the building-blocks for formulating the configuration synthesis problem. The requirements can be classified into two categories: (i) security requirements, and (ii) business constraints. There are also invariant and user-defined constraints on security implementations.

A. Formalizing Security Requirements: Isolation

The more a host is isolated from other hosts in the network, the potential threat to security becomes less. We define *isolation* as the *restriction* on the connectivity, i.e., network communication. The communication between two hosts can be restricted applying different security devices or systems, such as firewall, IPSec, IDS, NAT, etc. For example, a firewall can be placed to simply block some traffic flows (i.e., complete isolation), while IPSec can be placed to ensure authenticated transmission for the allowed flows (i.e., restriction based on authorization) in a network segment. Both of these devices are required to ensure authenticated and controlled traffic flow.

Isolation Patterns: Isolation patterns can be network level, host level, or application level. In this research, we consider the network level isolation, which includes the following patterns:

- *Access deny*. This is naturally enforced by a firewall.
- *Trusted communication*, i.e., authenticated and encrypted communication (IPSec devices).
- *Payload inspection*. This is done by an intrusion detection system (IDS).
- *Source identity hiding*. A network address translation (NAT) device is applied in order to use different address (typically a real IP address) instead of the original address. With respect to security, it can give security ensuring one way communication (i.e., outgoing only).
- *Traffic forwarding through Proxy*. For example, a reverse proxy gives a layer of security in terms of traffic filtering in the proxy instead of the server.

ConfigSynth allows network administrators to define isolation patterns considering different security devices (*primitive isolation*) and their combinations (*composite isolation*), along with their relative order based on the capabilities and functionalities of the devices. A set of isolation patterns is

TABLE I
ISOLATION PATTERNS AND SECURITY DEVICES

Id (d)	Device Name	Isolation Pattern	Isolation (k)
1	Firewall	Access Deny	1
2	IPSec	Trusted Communication	2
3	IDS	Payload Inspection	3
4	NAT	Source Identity Hiding	4
2 &4	IPSec & NAT	Traffic Forwarding through Proxy with trusted communication	5

shown in Table I. Each pattern is represented using an ID, k. We formalize the isolation measures (i.e., the *security configurations*) as a set of rules, $\{IR_1, IR_2, .., IR_n\}$, where each isolation rule IR_r is defined as follows:

$$IR_r : y^k_{i,j}(g), \text{where, } i,j \in \mathbb{H} \text{ and } g \in \mathbb{G}$$

The variable, $y^k_{i,j}(g)$ indicates that corresponding k'th isolation pattern is required to be deployed between the host pair $\{i,j\}$ for service g. Note that a host can represent a group of hosts that have the same properties (e.g., OS, services, etc.), the same level of users, and reside in the same subnet.

Isolation Pattern and Security Device: Usually, an isolation pattern is related to a particular type of security device. This one-to-one matching is true for primitive isolation patterns. In case of a composite isolation pattern, it is required to deploy more than one security device. The following equation models the relationship between an isolation pattern and associated security device(s):

$$\forall_{i,j,g}, y^k_{i,j}(g) \Rightarrow x^d_{i,j}(g) \tag{1}$$

Equation (1) specifies that if k'th isolation is selected for $g(i,j)$ flow, the d'th (type of) security device is required to be deployed between the host pair $\{i,j\}$ (i.e., on the route of the flow). A particular value of d denotes a particular type of security device. For example, as shown in Table I, $d = 1$ represents a firewall security device. If k'th pattern is a composite isolation pattern, multiple security devices are required to implement the isolation pattern. Hence, in this case, multiple $x^d_{i,j}(g)$s are true.

Score of an Isolation Pattern: We define the isolation *score* (also named as *rank*) of the kth isolation pattern between a pair of hosts $\{i,j\}$ under the network service g by the parameter $L^k_{i,j}(g)$. The score of an isolation pattern denotes its isolation capability compared to others. The scores can be given by the administrator explicitly or be computed based on the relative orders of the isolation patterns according to their isolation capabilities. In the latter case, a simple formal model is developed based on the given partial order between different isolation patterns. The model generates a complete relative order by assigning a *score* to each isolation pattern. It is plausible to assume the same score (L^k) for a particular isolation pattern irrespective of hosts and services.

The isolation scores are normalized according to a specified range, e.g., a scale of $0-1$. Note that this scoring of isolation patterns is relative and security requirements based on this scoring system reflects the same relative meaning.

Isolation of a Host: The decision variables $y^k_{i,j}(g)$, for all k, represent isolation patterns between a pair of hosts $\{i,j\}$ for the flow $g(i,j)$. These decision variables and associated isolation weights $L^k_{i,j}(g)$ are used to formally define the total isolation ($I_{i,j}$) of j with respect to the incoming traffic from i. $I_{i,j}$ is formalized as follows:

$$I_{i,j} = \frac{\sum_g \sum_k y^k_{i,j}(g) \times L^k_{i,j}(g)}{\sum_g \sum_k y^k_{i,j}(g) \times 1}$$

The equation indicates that the isolation between a pair of hosts $\{i,j\}$ is the sum of the isolation measures taken for different services between these hosts. The equation also indicates that the isolation is normalized by dividing the sum by the maximum possible isolation (i.e., the maximum isolation for a flow $g(i,j)$ is 1 in the scale of $0-1$). We consider the similar normalization throughout the model. For the ease of presenting the equations, we do not show the normalization factors (i.e., the denominators at the right hand side of the equations) for the rest of the paper. The total isolation score of a host j is defined in (2).

$$I_j = \sum_{i \neq j} I_{i,j} \tag{2}$$

Equation (3) represents the *overall isolation in the network* (i.e., the *network isolation*) considering all of the hosts.

$$I = \sum_i I_i \tag{3}$$

B. Formalizing Business Constraints: Usability

Business constraints play a significant role in synthesizing usable and cost-effective security configurations in a network. For example, a higher isolation can provide strong defense in the network, but the usability of the network might reduce to a level which is unacceptable to the organization. In ConfigSynth, we formalize the synthesis problem under two business constraints: (i) usability and (ii) deployment cost. In this subsection, we discuss the formalization of the usability.

The usability of the network depends on the ranks of the service flows between the hosts in the network. The rank of a service flow denotes the demand of the flow. Each service flow $g(i,j)$ is associated with a rank, $a_{i,j}(g)$. These ranks are expected to be given in the form of a relative order by the administrator based on the organizational demand. Partial information can be given, from which a complete relative order can be derived, as it has been shown in the case of the isolation patterns. The usability of a service g in a host j is formalized as follows:

$$S_j(g) = \sum_i \sum_k b^k_{i,j}(g) \times a_{i,j}(g)$$

The application of an isolation pattern to a flow can affect the usability of the flow. The parameter $b^k_{i,j}(g)$ represents the usability of the flow $g(i,j)$ due to applying the k isolation pattern between $\{i,j\}$. We assume that the usability depends on the isolation pattern, not on the host-pair (i.e., $b^k_{i,j}(g) = b^k(g)$). The value of $b^k(g)$ can be determined based on the prior knowledge of network security by considering the time or effort required to get a service access under an

isolation measure. The usability S_j with respect to a host j represents the accumulated usability considering all of the services running in the host.

$$S_j = \sum_g S_j(g)$$

The overall usability of the network (i.e., the *network usability*) is represented by (4).

$$U = \sum_j S_j \tag{4}$$

C. Formalizing Business Constraints: Deployment Cost

The deployment of a security device incurs costs and an organization often has an afford limit for deploying security measures. The number of security devices depends not only on the isolation measures but also on the topology. There are usually similar types of isolation between multiple host-pairs and these host-pairs can share one or more links for communication. In this case, placing a single security device at one of the shared links may ensure the desired isolation. Therefore, modeling correct and optimal placements of the security devices considering the network topology, the isolation patterns, and the budget is very challenging.

Modeling Flow Routes: ConfigSynth requires the flow routes between the hosts for determining the placements of the security devices satisfying the isolation measures. A *flow route*, $F^z_{i,j}$ is defined as a set of links $\{l_{i,j,z,1}, l_{i,j,z,2}, ...\} \subseteq \mathbb{L}$, that form a path from a source i to a destination j. As multiple routes are possible between a pair of hosts, z indicates the index of the flow route (i.e., the z'th route), between the host-pair $\{i, j\}$. The term $|F^z_{i,j}|$ denotes the path length, i.e., the number of hops or links in the path. $F_{i,j}$ denotes all of the flow routes possible from i to j.

$$F_{i,j} \Rightarrow \bigwedge_z F^z_{i,j}$$

ConfigSynth finds the flow routes for a host pair by applying a path searching algorithm on the network topology.

Formalizing Device Placements: Equation (1) specifies the security devices which are required to employ an isolation pattern. The placements of the security devices on the flow routes are modeled from these specifications.

If an isolation pattern, e.g., 'access deny', is selected for the traffic from a host i to a host j, then it is required to block the traffic through all possible flow routes between $\{i, j\}$. Equation (1) specifies a firewall to be deployed for implementing an 'access deny' isolation pattern. Hence, there should be a firewall deployed at least on a link of each flow route. We formalize the placement of a security device d for a particular pair of hosts as follows:

$$x^d_{i,j}(g) \Rightarrow \forall_z \exists_t l^d_{i,j,z,t} \tag{5}$$

In the equation, $l^d_{i,j,z,t}$ represents that a security device of type d is deployed on the link $l_{i,j,z,t}$. Note that if there is a security device, e.g., firewall, on the flow route for a host pair, this does not imply that the flow access between the pair is denied. It is denied only if the 'access deny' isolation pattern is specified for the host pair.

For the deployment of the security devices, the deployment cost is computed as the summation of the costs of all of the devices deployed in different links. We define C_d as the average deployment cost of the security device d. Now, if l^d denotes whether a security device d is deployed on the link $l \in \mathbb{L}$, the total deployment cost C is computed as follows:

$$C = \sum_l \sum_d l^d \times C_d, \text{where } l^d \Rightarrow \exists_{i,j,z,t}, l^d_{i,j,z,t} \tag{6}$$

D. Modeling Constraints

The main objective of our configuration synthesis problem is to maximize the security in the network by satisfying various security requirements as well as the organization's business constraints. Thus, the synthesis problem is formalized as the satisfaction of following three threshold based constraints in selecting the security measures (i.e., isolation patterns) on the network flows.

$$TC : (I \geq Th_I) \wedge (U \geq Th_U) \wedge (C \leq Th_C) \tag{7}$$

In the equation, Th_I, Th_U and Th_C represent the slider values, i.e., the constraints on the network isolation, usability, and deployment cost, respectively. The network isolation and the network usability must be greater than or equal to their respective threshold values, Th_I and Th_U. The deployment cost must also be within the budget, Th_C.

III. Conclusion

In this paper, we present an automated framework, ConfigSynth, for synthesizing correct and cost-effective network security configurations. It formally models the network topology, security requirements in terms of isolation, and the organizational business constraints in terms of usability and deployment cost, along with different invariant and user-defined constraints. Then, the framework formalizes the security design synthesis problem as the conjunction of all the requirements and constraints.

References

[1] B. Zhang and E. Al-Shaer. Synthesizing distributed firewall configurations considering risk, usability and cost constraints. In *CNSM*, 2011.

[2] L. de Moura and N. Bjørner. Satisfiability modulo theories: An appetizer. In *Brazilian Symposium on Formal Methods*, 2009.