Automated Synthesis of Resiliency Configurations for Cyber Networks

Mohammad Ashiqur Rahman*, Abdullah Farooq†, Amarjit Datta*, and Ehab Al-Shaer†
*Department of Computer Science, Tennessee Tech University, USA
†Department of Software and Information Systems, University of North Carolina at Charlotte, USA
Emails: marahman@tntech.edu, afarooq@uncc.edu, adatta42@students.tntech.edu, ealshaer@uncc.com

***Abstract*—Enterprise networks deploy security devices to control access and limit potential threats. Due to the emergence of zero-day attacks, security device based isolation measures like access denial, trusted communication, and payload inspection are often not adequate for the resilient execution of an organization's mission. Diversity between two hosts in terms of operating systems and services running on these hosts is crucial for limiting the attack propagation. Since different software systems have different vulnerabilities, it is important to have the hosts diversified considering the isolation among the hosts as well as the mission requirements. In this paper, we present a formal model for synthesizing network resiliency configurations. The resiliency design integrates isolation and diversity measures. We take the network topology, resiliency requirements, and business constraints as inputs. Then, our proposed model synthesizes cost-effective resiliency configurations satisfying the constraints. The output of the model provides necessary placements of different security devices in the topology and necessary installments of operating systems and services on the hosts. We demonstrate the execution of the proposed model as well as their scalability using simulated experiments.**

***Index Terms*—resiliency configuration; automated synthesis; isolation; diversity; formal model.**

I. Introduction

Resiliency is the state of a system which keeps the damage of the system limited while the attack intensity increases. While a system cannot be secured completely, especially considering zero day vulnerabilities and attacks, the system should be designed in a resilient manner which can keep the mission-critical services running even in a compromised situation. Therefore, the design of resiliency configurations for an enterprise network is crucial. The resiliency of an enterprise network has become crucial in this highly competitive world, while security threats are expanding vastly and cyberattacks are growing extensively. Although most organizations are emphasizing on the enforcement of their resiliency, they need to satisfy organizational usability requirements. Moreover, each organization has limited resources and the deployment cost for ensuring the resiliency cannot exceed the budget. In order to ensure strong resiliency of a network, it is important to explore different resiliency measures. Furthermore, the contention between the resiliency requirements and the business constraints needs to be resolved. Therefore, the resiliency architecture design is a challenging task.

This work focuses on resistance-based resiliency configurations that include two kinds of measures: isolation and diversity. An isolation measure can be firewall, Internet protocol security (IPSec), or intrusion detection device (IDS)-based protection, while a diversity measure indicates a heterogeneity in terms of operating system (OS) or service, in order to resist the propagation of adversarial exploitations. These resiliency measures are often interdependent with one another. Usually, a resiliency design needs to satisfy the organization's business requirements and constraints, which are mostly represented in terms of usability and deployment cost. The usability depends on various factors, including with demands of different service flows. A cost is involved with deploying security devices in the network or installing diverse operating systems (and services) to implement resiliency measures.

In this work, we present a formal framework for the automatic synthesis of network resiliency configurations. In this framework, we take the network topology, resiliency requirements, and business constraints as inputs, and formulate a model to synthesize the resiliency configurations. We apply Satisfiability Modulo Theories (SMT) [1] to implement this framework. The framework can be used as a decision support system to ensure resiliency of a network.

The rest of this paper is organized as follows: The resiliency measures are discussed in Section II. We present the proposed synthesis framework in Section III. We evaluate the framework in Section V, which is followed by a brief discussion on limitations of this framework. The related work is discussed in Section VII. We conclude the paper Section VIII.

II. Background

There are often three kinds of resiliency measures: *resistance*, *deterrence*, and *recovery*. In the case of resistance, static resiliency measures like isolation and diversity are applied to hinder the potential threats. With regards to deterrence, the system's parameters change frequently such that adversaries can not have certain knowledge to launch attacks. A system can be recovered from an attack using alternative means as well as taking rejuvenation steps. In this work, we focus on resistance-based resiliency.

A. Isolation

Isolation is a restriction or resistance imposed on the network communication. The communication between two hosts can be restricted through access control policies by deploying different security devices, such as firewall, IPSec, IDS, etc. For example, a firewall provides isolation by blocking traffic flows, while IPSec offers isolation by ensuring authenticated transmission for the allowed flows.

An isolation pattern is often associated with a security device. For example, "access denial" is associated with a fire-

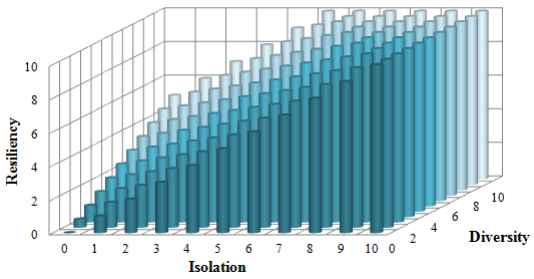


Fig. 1. The figure shows the resiliency of a network with respect to the isolation and diversity measures. All the scores are normalized with respect to a scale of 0 to 10. Resiliency is calculated as a function of isolation and diversity where the diversity measures are weighted as half of the isolation measures in terms of providing resiliency.

wall. An isolation pattern can be of network-level (*e.g.*, access control), host-level (*e.g.*, virtual systems), or application-level (*e.g.*, role-based access control). In this work, our focus is on the network level isolation patterns, particularly "access denial" (firewalls), "trusted communication" (IPSec devices that usually build trusted paths by ensuring authenticated and encrypted communication), and 'Payload inspection" (IDS). In order to deploy correct and cost-effective isolation in a network, it is required to define the level of restriction each isolation pattern can enforce on the flows (*i.e.*, the effectiveness as an isolation measure), and their impact on the usability [2].

B. Diversity

In order to make the network secure, a popular and useful mechanism is patching known vulnerabilities throughout the network. However, a zero day attack can compromise a host, while the attack can further be propagated to compromise other hosts of the network by exploiting the same OS or software vulnerability by using various network services. Diversity is an interesting and useful mechanism to improve network resiliency by resisting attack propagation in the network [3], [4]. Operating systems, applications or programs, routing protocols, and even security measures used in an information system can all be diversified to increase the security and survivability of the system [5]. The basic idea is that the more diverse a system is, the less the correlated risk.

It is shown that multiple software substitutes and the same software on different operating systems do not have same vulnerabilities or cannot be compromised with the same exploit [6]. Vulnerabilities of 11 different OSes are analyzed over a period of 18 years from NIST National Vulnerability Database (NVD) to check how many vulnerabilities affect more than one operating system [7]. It is found that when diverse OSes are used, an attacker needs to spend more time and effort to exploit the system. Although recent works show the potential of diversity as a resiliency mechanism, few efforts have been made to deploy proper diversity by synthesizing necessary configuration parameters for a network with respect to the resiliency, usability, and budget constraints.

C. Objective

The design of a resiliency architecture involves many parameters with regards to the network topology, network hosts, mission goal or connectivity requirements, usability, attack model, and resiliency measures and their deployment cost. These parameters are dependent on one another and it is crucial to consider these interdependencies to design a resiliency architecture. In this work, we formally model the resiliency of a network in terms of different resiliency measures. A solution to this model will provide necessary configurations leading to a resilient system satisfying resiliency requirements and usability and deployment cost constraints. We define the resiliency as a function of isolation and diversity, which intuitively represents the difficulty of launching an attack (attack difficulty). There are different isolation and diversity measures or patterns and we assume a scoring system for each kind. The scores abstract the capabilities of patterns in terms of resiliency. We combine the scores for isolation and diversity to measure the resiliency of the network. Fig. 1 shows an example of scoring resiliency. With regards to the attack model, we consider the multi-step attack paths (*i.e.*, multiple steps of vulnerability exploitation to compromise a host) in synthesizing the resiliency configurations.

III. Resiliency Configuration Synthesis Model

We start this section with the brief description of the resiliency synthesis framework. Then, we describe the formal modeling of this framework.

A. Architecture of the Synthesis Framework

The synthesis framework follows a top-down design approach, in which we take high level requirements and then synthesize fine-grained resiliency configurations accordingly (Fig. 2). Our synthesis framework follows several steps. First, it takes necessary inputs, such as the network topology, existing resiliency configurations (*e.g.*, isolation configurations, deployed security devices, and installed operating systems and services) if there are, resiliency requirements, and business constraints (*e.g.*, minimum usability and maximum deployment cost). The next step includes the formal modeling of the network topology, traffic flows, resiliency measures, their deployment cost, impact on usability, resiliency requirements, and business constraints. In the last step, the security design synthesis problem is encoded into SMT logics. The implementation is solved using an SMT solver that determines appropriate resiliency configurations.

B. Network Model

We define the network model as $\langle \mathbb{N}, \mathbb{L} \rangle$. $\mathbb{N}$ defines a finite set of network nodes including hosts and routers. Thus, $\mathbb{N}$ is a union of two sets: $\mathbb{H}$ and $\mathbb{R}$. $\mathbb{H}$ denotes a finite set of hosts. $\mathbb{R}$ denotes a finite set of routers. Each host is identified by an ID (*e.g.*, IP address). A host may execute one or more services, which are accessed by different hosts. A service is denoted using $s \in \mathbb{S}$, where $\mathbb{S}$ is the set of all services. An operating system (OS) is running on each host. Let $\mathcal{O}$ is the set of possible OSes. The term $s(i, j)$ defines the flow between a pair of hosts $\{i, j\}$, where i is the source and j is the destination, under a service s. $\mathbb{L} \subseteq \mathbb{N} \times \mathbb{N}$ is a finite set of links, which defines the interconnections between the network hosts.

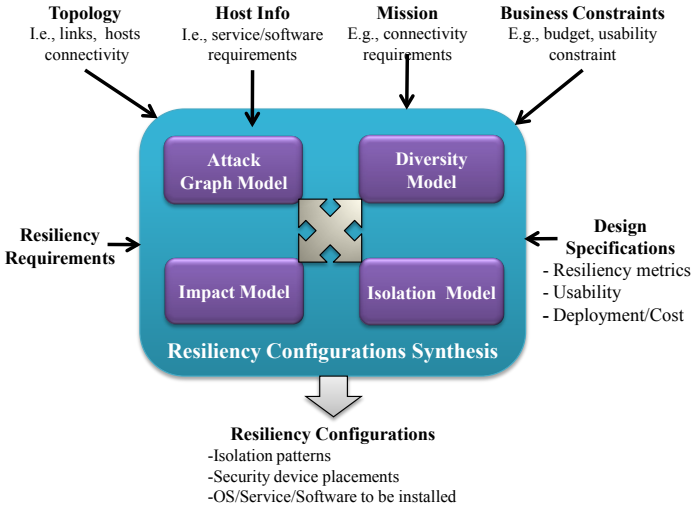


Fig. 2. The framework of the resilient configurations synthesis model.

C. Diversity Model

The diversity between two hosts i and j is denoted by $D_{i,j}$. Let O_i ($O_i \in \mathcal{O}$) denote the operating system (OS) of host i and $\mathcal{S}_i \subset \mathcal{S}$ denote set of services running on host i. $D_{i,j}$ can be calculated as a function of $\mathcal{S}_i$, $\mathcal{S}_j$, O_i, O_j, which valuates $D_{i,j}$ based on different criteria with respect to the similarity between OSes and services running on this host pair. The most similarity offers the least diversity. That is, if $\mathcal{S}_i$ and $\mathcal{S}_j$ are the same and O_i and O_j are the same, $D_{i,j}$ is the minimum (0). When $\mathcal{S}_i$ and $\mathcal{S}_j$ are different as well as O_i and O_j are different, $D_{i,j}$ is the maximum. In this way, we can have four levels or criteria of diversity:

1) $\mathcal{S}_i$ and $\mathcal{S}_j$ are similar and O_i and O_j are the same.
2) $\mathcal{S}_i$ and $\mathcal{S}_j$ are different and O_i and O_j are the same.
3) $\mathcal{S}_i$ and $\mathcal{S}_j$ are the same and O_i and O_j are different.
4) $\mathcal{S}_i$ and $\mathcal{S}_j$ are different and O_i and O_j are different.

Let $E_{i,j}$ represent the diversity criterion of host pair $\{i, j\}$. Then, the minimum level (*i.e.*, criterion 1) of diversity can be formalized as follows:

$$(O_i = O_j) \wedge \ \exists_{s_i \in \mathcal{S}_i, s_j \in \mathcal{S}_j}(s_i = s_j) \ \rightarrow (E_{i,j} = 1) \quad (1)$$

It is also important to include OS families into diversity calculation. For example, two different Windows OSes have more similarities compare to one Windows and one Linux OSes. In this case, the above mentioned criteria will have further classifications. For example, if two OSes are different but they have the same family as well as the similar services running on them, this criterion is better than the first (minimum) one while worse than the second one () in the list of criteria depicted above. As we consider the OS family, the diversity criteria set's size becomes six: for each of the highest two criteria (different OSes), it is considered if the OS families are different or the same. We define $\bar{F}_o$ to denote the family of OS o and F_i to denote the family of the OS installed on host i, and formalize each diversity criterion. The formalization of diversity criterion 3 (*i.e.*, having different OSes and services, but the same OS family), as an example, is presented below:

$$(O_i = o) \rightarrow (F_i = \bar{F}_o) \quad (2)$$

$$(O_i \neq O_j) \wedge (F_i \neq F_j) \wedge \ \exists_{s_i, s_j}(s_i = s_j) \ \rightarrow (E_{i,j} = 2) \quad (3)$$

Different versions of a particular software product often have different vulnerabilities. Therefore, the diversity modeling can also consider the software versions.

A score is associated with each diversity criterion. If $\mathcal{E}$ is the set of these criteria and E_e represents the diversity score for criterion $e \in \mathcal{E}$, we formalize the diversity as follows:

$$\exists_{e \in \mathcal{E}}(E_{i,j} = e) \ \rightarrow (D_{i,j} = E_e) \quad (4)$$

D. Isolation Model

We model the isolation in terms of different network-based security devices, particularly firewall, IPSec, and IDS.

Isolation Pattern and Security Device:

An application of an isolation pattern requires the deployment of one or more security devices. 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):

$$\forall_{i,j,s} \ (y_{i,j}(s) = p) \rightarrow (x_{i,j}(s) = d) \quad (5)$$

Here, $y_{i,j}(s)$ indicates the isolation pattern that is required to be applied between host pair $\{i, j\}$ for service s, while $x_{i,j}(s)$ specifies the security device needs to be deployed for the same traffic. The equation expresses that if pattern p is selected as the isolation measure, device d needs to be deployed between host pair $\{i, j\}$.

Isolation Between a Host Pair:

Parameter $y_{i,j}$ represents the isolation pattern between a pair of hosts $\{i, j\}$ for flow $s(i, j)$. Let W_p denotes the weight (or score) of isolation pattern p. Then, the isolation ($I_{i,j}$) of j with respect to the incoming traffic from i is formalized as:

$$I_{i,j} = \frac{\sum_{s, y_{i,j}(s)=p} W_p}{\sum_s 1} \quad (6)$$

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 $s(i, j)$ is 1 in the scale of $0-1$). We use 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.

E. Resiliency Model

The resiliency of host j with respect to source host i is a function of diversity and isolation:

$$R_{i,j} = f(D_{i,j}, I_{i,j}) \quad (7)$$

In this work, we define this function as a weighted sum of the diversity and isolation:

$$R_{i,j} = \beta D_{i,j} + (1-\beta) I_{i,j} \tag{8}$$

Here, β is the weight given to the diversity, while $1-\beta$ is the weight of the isolation.

Attack Transitivity:

Attack graph or path-based security analysis is crucial to understand the potential exploitation of vulnerabilities [8], [9], [10]. In order to consider transitive exploitations, we generalize the resiliency notation as $R^k_{i,j}$, where k is the number of intermediate hosts in the attack path (*i.e.*, the number of intermediate hosts that need to be compromised to attack the target). Therefore, Equation (8) represents the calculation of $R^0_{i,j}$, where $k=0$. We model $R^k_{i,j}$ as follows:

$$R^k_{i,j} = \frac{\sum_t (R^{k-1}_{i,t} + R^0_{t,j})}{\sum_t 1} \tag{9}$$

Where t is an intermediate host ($i \neq t$ and $j \neq t$). Equation 9 intuitively considers all the paths from host i to host j with k intermediate hosts. If we remove the denominator part of the equation, then it can be elaborated as follows and so on:

$$R^k_{i,j} = \sum_t (\sum_{t'} (\sum_{t''} (R^{k-3}_{i,t''} + R^0_{t'',t'}) + R^0_{t',t}) + R^0_{t,j}) \tag{10}$$

The number of steps (intermediate hosts) has impact on the attack propagation because an attacker needs to exploit multiple hosts one after another till compromising the ultimate target host, which is much expensive compared to one or less steps. Therefore, along with adding the resiliency, we need to consider the number of steps. Thus, we model the resiliency of a host pair by considering each step k and providing a specific weight to each type. Let V_k is the weight of the resiliency when the number of intermediate hosts is k. Since a higher resiliency is required while k is smaller compared to larger values of k, we often assign higher weights to V_k for smaller ks. We use $\hat{R}_{i,j}$ to define the ultimate resiliency between host pair $\{i,j\}$ (source i and destination j). $\hat{R}_{i,j}$ is defined as:

$$\hat{R}_{i,j} = \frac{\sum_k R^k_{i,j} \times V_k}{\sum_k V_k} \tag{11}$$

Here, $0 \leq k \leq K$ and K is the highest number of intermediate hosts considered in attack paths.

Although each intermediate host pair in the sequence of the attack path, from the attack initiating host to the ultimate victim host, may have high diversity, the overall diversity may not be significant when all hosts on this path are considered. This problem has been solved as we consider all possible attack paths with k intermediate hosts to calculate the ultimate resiliency for a host pair. It is worth mentioning that the larger is K and the number of hosts, the bigger is the set of $R^k_{i,j}$s (from Equation 9), while the upper $R^k_{i,j}$s for larger ks depend on $R^k_{i,j}$s for smaller ks.

Constraint on Overall Resiliency:

The resiliency of host i (R_i) is the summation of the resiliency with respect to each incoming traffic flow:

$$R_i = \sum_j \hat{R}_{j,i} \tag{12}$$

The overall resiliency of the network (R) is the summation of the resiliency of each host, as it is specified bellow:

$$R = \sum_i R_i \tag{13}$$

The main resiliency requirement specifies that this overall resiliency must be at least a threshold value, as follows:

$$R \geq T_R \tag{14}$$

There can be further requirements. For example, each host should have a minimum resiliency: $R_i \geq \bar{T}_R$.

F. Deployment Cost

Each host requires an OS, while the implementation of an isolation pattern requires a deployment of one (or more) security device(s). Each installment of an OS or a deployment of a security device incurs costs, while the total deployment cost needs to be affordable.

Diversity Implementation Cost:

In modeling diversity deployment cost, we primarily include the OS installment cost. Different OSes have different installment cost, which mainly covers the license fees. The price per installment often follows a concave curve, *i.e.*, the more are the number of installments the less is the per installment cost.

Let $C_{o,m}$ denote the mth installment cost of OS o. We consider a geometric cost function for $C_{o,m}$, as follows:

$$C_{o,m} = \lambda_o \times \delta^{m-1} \tag{15}$$

We use O_i to denote the OS (to be) installed on host i. If M_o denotes the total number of installations of OS o on the hosts, then $M_o = \sum_{O_i=o} 1$. Therefore, the cost (C_o) for having M_o number of installations of OS o is as follows:

$$C_o = \lambda_o \frac{1-\delta^{M_o-1}}{1-\delta} \tag{16}$$

Then, the total cost of installing OSes on the hosts ($\bar{C}$) is:

$$\bar{C} = \sum_o C_o \tag{17}$$

If different versions of a particular software (or service) is considered to create diversity, associated costs to create such diversified versions needs to be considered as well.

Isolation Implementation Cost:

The number of security devices depends both on the isolation measures and the topology. This is because it is common to have similar types of isolation patterns between multiple host-pairs, and these host-pairs often share one or more links for routing. In this case, placing a single security device at an shared link ensures the desired isolation. If there are more than one routing path between a host-pair, it is crucial to deploy necessary security devices considering each alternative path.

Security Device Placements: We define traffic routing paths between the hosts to determine the placements of the security devices satisfying the isolation measures. We define all

possible routing paths from source i to destination j as $\mathcal{P}_{i,j}$, while $\mathcal{P}_{i,j,z}$ ($\mathcal{P}_{i,j,z} \subset \mathcal{P}_{i,j}$) is z'th path from host i to host j. $\mathcal{P}_{i,j,z}$ is a set of links $\{l_{i,j,z,1}, l_{i,j,z,2}, ...\} \subseteq \mathbb{L}$ that form a route from host i to host j.

The placements of the security devices on the flow routes are modeled from the mapping between the isolation patterns and security devices specified by Equation (5). In this work, we consider three network isolation patterns: "access denial", "trusted path", and "payload inspection". Equation (5) specifies that for implementing an "access denial" isolation pattern for host pair $\{i, j\}$, a firewall needs to be deployed on each routing path from host i to host j. Deploying a firewall on a routing path intuitively means that there should be a firewall deployed at least on a link of each flow route. The deployment of an IDS (in the case of implementing "payload inspection") follows the same intuition. We formalize the placement of a security device d (a firewall or an IDS) for a particular pair of hosts as follows:

$$(x_{i,j}(s) = d) \rightarrow \forall_z \exists_t l^d_{i,j,z,t} \tag{18}$$

Here, $l^d_{i,j,z,t}$ represents that a security device of type d is deployed on link $l_{i,j,z,t}$. The implementation of isolation pattern "trusted communication" is different from that of "access denial" and "payload inspection". To ensure an encrypted tunnel between a host pair, it is required to place two IPSec gateways at the source and destination ends [2].

Device Deployment Cost: 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 $\hat{C}$ is computed as follows:

$$\hat{C} = \sum_{l \in \mathbb{L}} \sum_{d} l^d \times C_d, \text{where } l^d \rightarrow \exists_{i,j,z,t}, l^d_{i,j,z,t} \tag{19}$$

Constraint on Deployment Cost:

The total deployment cost C is the summation of the total diversity deployment cost and the isolation deployment cost:

$$C = \bar{C} + \hat{C} \tag{20}$$

The main deployment cost constraint specifies that this total cost should not cross the budget (a threshold value). The constraint is formalized as follows:

$$C \geq T_C \tag{21}$$

G. Usability

Usability constraints play a significant role in synthesizing resiliency configurations. A higher network isolation or diversity can provide strong defense, but the network usability may reduce to a level which is unacceptable to the organization.

Impact of Diversity on Usability:

Each operating system has different level of usability with respect to its user. The usability of a particular kind of operating systems depends on the user's technical proficiency or experience. Therefore, it is useful to consider each user's preference in computing usability. If we can correspond each host with an user, then the preference can be taken with respect to each host. Let $F_{i,o}$ represent the preference (usability) of host i over OS o, while $\bar{U}_i$ represent the host's usability in terms of deployed diversity. Then:

$$(O_i = o) \rightarrow (\bar{U}_i = F_{i,o}) \tag{22}$$

Although a usability modeling like above would be useful, considering each distinct or individual preference can be inefficient. This is because, collecting information for each individual's preference may not be possible. Moreover, dealing with a large number of preferences may slow down the synthesis process. Since it is possible to get an idea of overall preference over each OS kind with respect to the organization, it is useful to consider usability in terms of OS only (F_o). Therefore, Equation (22) will be reformed as:

$$(O_i = o) \rightarrow (\bar{U}_i = F_o) \tag{23}$$

Impact of Isolation on Usability:

Usability Based on Isolation Pattern: The usability of the network depends on the ranks of the traffic (service) flows between the hosts in the network. The rank ($a_{i,j}(s)$) of a traffic flow ($s(i,j)$) denotes the demand of the flow. These ranks are given as a relative order based on the organizational demand. If no specification is given about the demand of different flows, all flows receive the default rank. The usability of a traffic flow $s(i,j)$ ($\hat{U}_{i,j}(s)$) is formalized as follows:

$$\hat{U}_{i,j}(s) = \sum_{y_{i,j}(s)=p} (b_p \times a_{i,j}(s)) \tag{24}$$

The application of an isolation pattern to a flow can affect the usability of the flow. The parameter b_p represents the usability of the flow $s(i,j)$ due to applying isolation pattern p for host pair $\{i,j\}$. The value of b_p can be determined based on the obvious or prior knowledge about the time or effort required to get a service access under an isolation measure. In the case of "access denial" based isolation, there is no usability at all. Thus, *i.e.*, $\forall_{s(i,j)}, b_1 = 0$, where $p = 1$ for "access denial". Other isolation patterns can be considered as maintaining the same usability, *i.e.*, $\forall_{p \neq 1}, b_p = 1$.

Usability of host j with respect to incoming traffic flows from host j ($\hat{U}_{i,j}$) is calculated as: $\hat{U}_{i,j} = \sum_s \hat{U}_{i,j}(s)$. The usability of host i ($\hat{U}_i$) represents the accumulated usability considering all of the service flows:

$$\hat{U}_i = \sum_j \hat{U}_{j,i} \tag{25}$$

Connectivity Requirements: Every organization usually has a number of service flows that are essential for its mission. Each of these connectivity requirements represents a flow that must be able to communicate. If $\mathcal{V}$ represents the connectivity requirements, each element $s(i,j) \in \mathcal{V}$ is a service flow that must be allowed from host i to host j. This intuitively specifies that the usability cannot be zero. Therefore:

$$s(i,j) \in \mathcal{V} \rightarrow U_{i,j}(s) \neq 0 \tag{26}$$

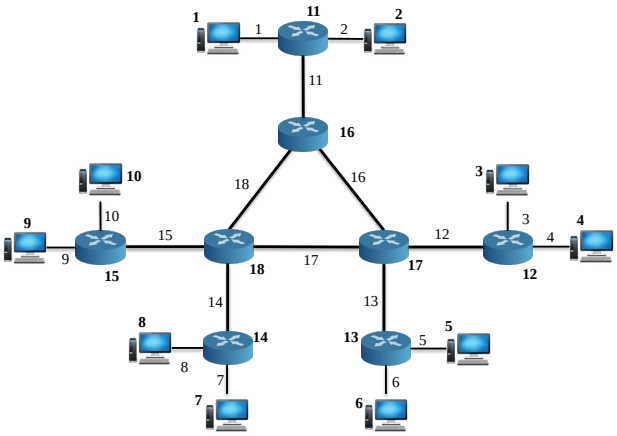


Fig. 3. The network topology considered in the example.

For each flow $s(i, j) \notin \mathcal{V}$), there is no specification for it, *i.e.*, the flow can either be allowed or denied.

Constraint on Usability:

The ultimate usability for host i (U_i) is the weighted sum of the diversity- and isolation-based usability scores:

$$U_i = \bar{U}_i + \hat{U}_i \tag{27}$$

The overall usability of the network is represented as follows:

$$U = \sum_i U_i \tag{28}$$

The main usability constraint specifies that this total usability cannot be less than a threshold usability:

$$U \geq T_U \tag{29}$$

IV. Case Studies

A. An Example

We implement our proposed synthesis model by encoding the system configuration and the constraints into SMT [1]. Here, we present a small example to demonstrate the execution of this model. Fig. 3 shows the network topology that we consider in this example. The input, other than the topology, is partially shown in Table I. The connectivity requirements simply represent traffic flows which must be allowed between different hosts. In this example, we have three primitive isolation patterns (*i.e.*, 'access denial', 'trusted communication', and 'payload inspection') and five operating systems of three OS families (Windows, Linux, and MAC). We also assume a single flow type (*i.e.*, corresponding to a single service) between each pair of hosts. We consider 5 services and, without loss of generality, we assume that all the services can run on a host, while in order to deny a service flow to a host there should be "access denial" for all communication with regards to this service. In this instance, the weights of isolation and diversity in the resiliency calculation are taken as 60% and 40%. The weights corresponding to their impact on usability are considered as the same. The intermediate steps of attack propagation is taken as 2, *i.e.*, a compromised host can compromise another host by compromising two intermediate hosts. The weights of resiliency are assumed as 50%, 33.33%, and 16.67%, respectively for 0 step, 1 step and 2 steps of transitive attacks.

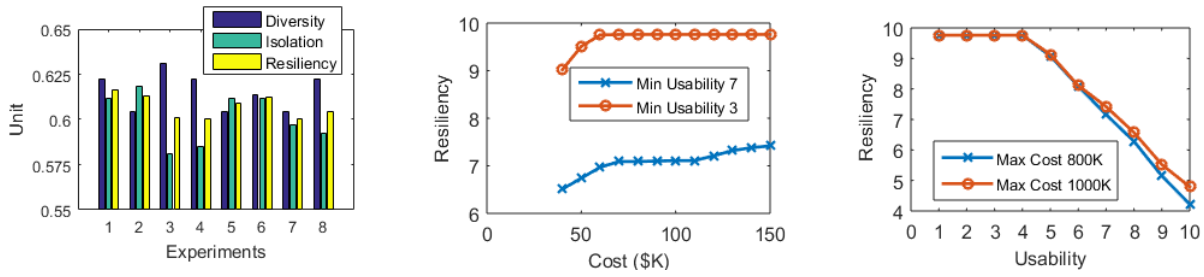
TABLE I
Partial Input to the Example

```
# Number of security devices
3
# Isolation if the isolation pattern is in use (Ordered in Scale 0-3)
3 2 1
# Usability if the isolation pattern is in use (Ordered in Scale 0-3)
0 2 3
# Cost of each isolation device
22000 15000 18000
# Number of operating systems
5
# Diversity scores (Ordered in Scale 0 - 5)
5 4 3 2 1 0
# Usability of each OS (Ordered in Scale 0 - 3)
3 2 1 2 1
#Cost of OS installations ($)
100 150 150 75 200
# Connectivity requirements for each host (comma separated)
# Each entry i represents the hosts that must be reached by host i [ends with 0]
3 0, 4 0, 1 2 0, 2 0, 3 4 0, 3 4 0, 1 2 0, 1 2 0, 1 2 0, 1 2 0
# Rank of the flows (comma separated)
# Each entry includes source and destination of the traffic and the demand
1 2 3, 2 1 3, 2 5 2, 2 6 2, 3 10 3, 4 10 3, 6 8 2
# Number of service options
5
# Service requirements 1 2 3 0, 2 3 0, 0, 0, 3 0, 0, 0, 5 0, 0, 5 0
# Exposure level (vulnerability exploitation steps)
2
# Resiliency requirement and usability and cost constraints
7 6 120000
```

Our formal model corresponding to this example gives a satisfiable result. From the assignments of the decision variables, we find the necessary resiliency configurations (*i.e.*, selection of isolation patterns, security device placements, and diversified OS installations). The results show that firewalls need to be installed at links 15, IPSec gateway devices at links 1, 2, 13, and 14, while IDS devices at links 3 and 5. The OSes are installed on the hosts as follows: Windows 7 on hosts 1 and 6, Windows 8 on hosts 2 and 8, Redhat on hosts 7, Ubuntu on host 4, and MAC on hosts 3, 5, and 9. The corresponding security policy specify that all communication from host 1 to hosts 5, 6, 7, 8, 9, and 10 are denied, host 1 to host 2 are encrypted, to host 3 are payload inspected, and the rest, *i.e.*, all communication to host 4 are allowed without any isolation measure. We can see that the OSes of host 1 and host 4 are highly diversified (different families-Windows and Linux). Host 2 has the similar security policy. All communications from host 3 to hosts 5, 6, 7, 8, 9, and 10 are denied, to hosts 1, 2, and 4 are payload inspected, and to the rest are allowed without any isolation measure. If we observe the security policy for host 5, all communications to hosts 1, 2, and 4 are denied, to hosts 7, 8, 9, 10 are encrypted, and to host 6 are payload inspected. We also see that only a single service flow is allowed to host 3 (even with payload inspection). In total, 220 flows are allowed (many be with "trusted communication" and "payload inspection" isolation measures), while the rest are denied.

B. Application of Isolation and Diversity for Resiliency

We analyze the individual scores of diversity and isolation for a specific resiliency requirement. We consider the example with a little different setup: both resiliency and usability



(a) (b) (c)

Fig. 4. (a) Individual diversity and isolation scores for a specific resiliency score, and (b-c) the impact of the budget and the usability constraint, respectively, on the maximum possible resiliency.

requirement scores are 6 and the budget is \$120K. The model with respect to this problem is solved and we find multiple satisfiable results. In Fig. 4(a), we present the individual scores of diversity and isolation, along with the ultimate resiliency score, for eight arbitrary solutions. We easily observe that when the diversity score is low, a higher isolation score is considered to achieve the required resiliency, and vice versa. That is, when the system is less diversified, a higher isolation is required among the hosts, while a highly diversified system needs to apply lower isolation.

C. Impact of Constraints on Maximum Resiliency

We investigate the impacts of budget (deployment cost constraint) and the usability constraint in achieving the maximum resiliency for a network. Fig. 4(b) and Fig. 4(c) present the results corresponding to this investigation. Fig. 4(b) illustrates how the maximum achievable resiliency depends on the deployment budget. In this investigation, we consider the same topology as in Fig. 3 with the usability requirement scores of 3 and 7 respectively. When the deployment budget is low, the maximum achievable resiliency is low. Resiliency increases as the budget for deploying isolation and diversity increases. The increase in the resiliency gets saturated after a point, *i.e.*, an increase in the budget does not increase the resiliency. As shown in Fig. 4(b), when the usability constraint is 7, the maximum achieved resiliency is significantly low in contrast to the previous case when the usability requirement has been only 3. In fact, a higher resiliency can be achieved when the organization can make a trade-off with the usability. These results motivate us to investigate more on the effects of the usability constraint on the maximum resiliency. The results are illustrated in Fig. 4(c). We again consider two different scenarios with respect to the budget (\$800k and and \$1000k) and investigate the effects of the usability requirement on the maximum resiliency. In both scenarios, the maximum achievable resiliency starts reducing significantly when the usability requirement is over 4.

V. Evaluation

In this section, we investigate the model's scalability in terms of the synthesis time.

A. Methodology

We generated the test networks arbitrarily considering the popular star and tree topologies. The star topology is combined with the ring topology. We usually consider star topology if it is not specified. With regards to the network size, we consider up to 250 hosts and 20 routers. In the test networks, we often randomly choose between 5 to 10 services between a pair of hosts, with a set of 2 to 8 alternative OSes considering 3 OS families. The resiliency and usability constraints are chosen considering a scale of $0-10$, where 0 stands for no resiliency or usability, while 10 specifies complete score. We run our proposed formal framework in a computer equipped with an Intel Core i5 Processor and 16 GB memory.

B. Evaluation Results

Impact of Problem Size on Execution Time: We run experiments under various sizes of networks considering the star topology. The resiliency requirement, the usability constraint, and the maximum budget are set to 5, 5, and \$1.2 million, respectively. Moreover, in these experiments, we assume that an attacker is capable of exploiting attack paths with at most 2 intermediate hosts (*i.e.*, $k = 2$). Fig. 5(a) shows the results and we observe that the synthesis time increases almost quadratically with the increase in the number of hosts. This is because the model size increased with the problem size as shown in Fig. 5(b). In this figure, we can see that the number of clauses or assertions that are made for a network size increases with the number of hosts. This increase is proportional to the number of traffic flows, which is the quadratic order of the hosts. We conduct another set of experiments with the same setup but considering the tree topology. Fig. 5(c) shows the results. We observe the similar exponential time growth as in Fig. 5(a), although the tree topologies take more time for the synthesis, as the core network of a star topology is often smaller and more centered than that of a tree topology.

We investigate the synthesis time by varying the number of routers for both star and tree network topologies. In these experiments, we set the number of hosts to 150. We consider the same usability constraint, the budget, and the maximum number of intermediate hosts in the attack paths, as the previous experiments. As we see in Fig. 5(d) and Fig. 5(e), when the number of routers in the network is higher, more time is needed by the solver to achieve a satisfiable solution. For a higher number of routers (thus, a increased number of links), the solver has more options to search, *i.e.*, there are more clauses in deploying security devices given the resiliency requirements and other constraints. Thus, it often needs more time to synthesize the resiliency configurations. We also evaluate the synthesis time with respect to the number

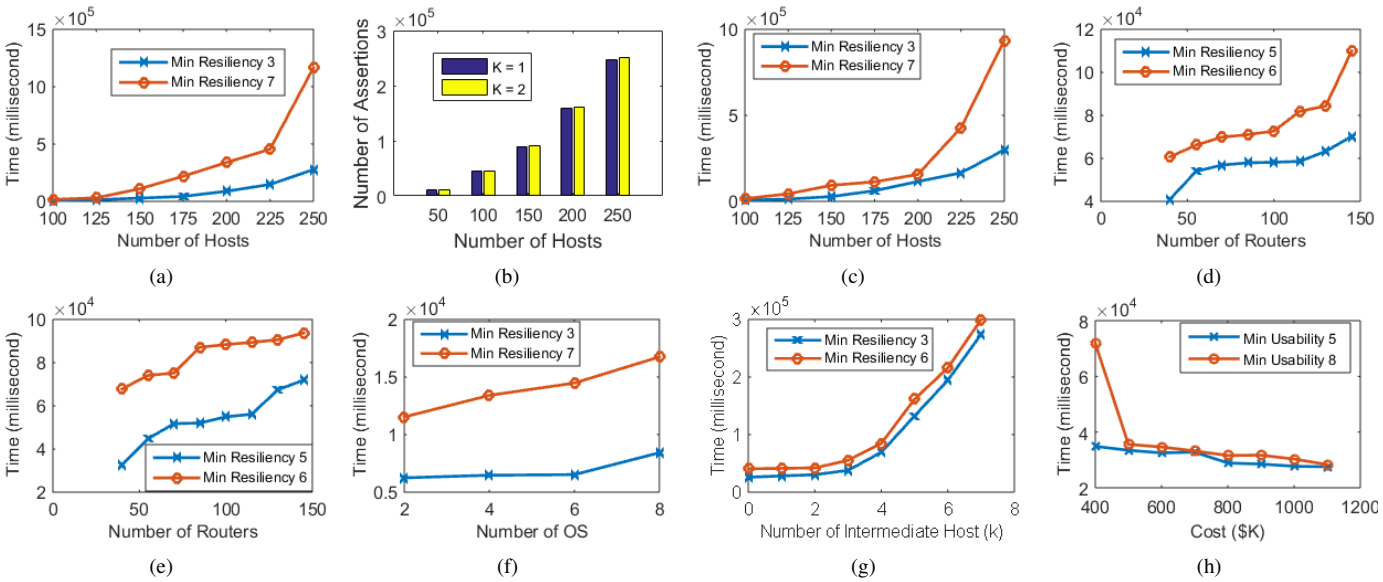


Fig. 5. The synthesis time with respect to (a) the number of hosts (star topology), (c) the number of hosts (tree topology), (d) the number of routers (star topology), (e) the number of routers (tree topology), (f) the number of different operating, (g) the number of intermediate steps or (exploitable) hosts considered in attack pats, and (h) the maximum budget constraint.

of OSes. Fig. 5(f) shows the results, where we observe linear growth in the synthesis time.

Impact of Attack Transitivity Steps on Execution Time: An adversary generally looks for a least effort path to attack a host. It is obvious that having no intermediate hosts to reach the target is the least effort path. However, such a direct path is not always possible and thus adversaries often need to exploit one or more intermediate hosts to attack the target. Therefore, our synthesis model intuitively considers such transitive attacks paths to provide resiliency configurations. We conduct experiments by varying the number of intermediate hosts (K) from 0 to 7. The usability and deployment cost constraints remain the same as the previous. From the experiments, with minimum resiliency requirements of 3 and 6, we observe that the synthesis time increases exponentially with the increase of K (Fig. 5(g)). This is because the number of alternative (attack) paths to a host increases for larger K. In Fig. 5(b), we have seen that the number of assertions increases with K.

Impact of Constraints on Execution Time: In Fig. 5(h), we investigate the effects of maximum budget on the simulation time in two scenarios with respect to the usability requirement (5 and 8). For each data point on both simulation, the resiliency requirement and K is set to 5 and 2, respectively. As we see in the figure, the less the budget is, the more time is required to get a satisfiable solution.

Execution Time in Unsatisfiable Cases: In the cases of very tight constraints, *e.g.*, when the resiliency (and/or usability) requirement is very high or the deployment budget is very limited, there may not be any satisfiable model. Fig. 6(a) and Fig. 6(b) show the execution times in unsatisfiable cases by varying the budget for 50 hosts and 200 hosts, respectively. If the budget is too low, the execution time is low, while the time increases very sharply when the budget increases, although there is still no solution. This is because the SMT solver requires exploring all alternative measures to conclude

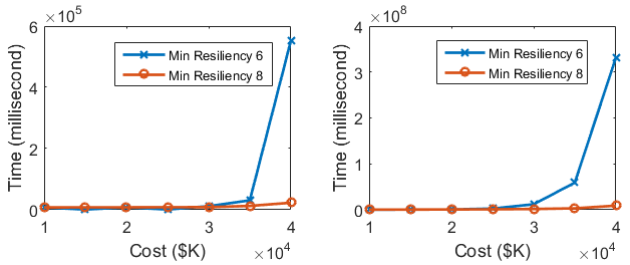


Fig. 6. The synthesis time with respect to in the unsatisfiable cases for (a) 50 hosts and (b) 200 hosts.

that there is no configuration that can satisfy all the constraints. When the budget comes closer to that of a satisfiable result, many of these alternative solution paths need to traverse to a larger extent without a success but increases the time rapidly. This behavior becomes more evident from the observation in Fig. 6 that this sharp increase happens earlier in the case of a low resiliency requirement than that of a higher one.

VI. Discussion

This section specifies some limitations of this work and includes corresponding discussions. We consider a comprehensive list of decision making components that effect the selection of resiliency configurations, many of which need a quantitative measuring of their capabilities in order to compare among alternative measures. The scoring of different resiliency metrics like diversity and isolation, as well as business metrics like usability, follow some relative orders of performance with respect to diversity, isolation, resiliency, and usability. Although such scores do not represent the absolute comparison among different resiliency or usability measures, they provide a relatively comparable notion of resiliency. This capability offers a highly useful basis for the decision support system in order to deploy necessary resiliency configurations. The level of resiliency meant by a particular selection of a value on the scale represents a relative understanding of the resiliency

between the 'complete' and 'no' resiliency scores. To our best of knowledge, there is no universally absolute scoring systems for these metrics which can quantitatively compare them.

We do not consider vulnerability specific to the OS or the services running on a Host. Since we consider threats like zero-day attacks, reachability and exposure-based threat modeling is worthwhile for the decision support system. We model resiliency as a weighted sum of diversity and isolation. However, the computation of resiliency can follow a different methods (*i.e.*, a different function of diversity and isolation), and the proposed framework is flexible enough to model it.

VII. Related Work

Security policy misconfigurations have been studied extensively throughout the last decade [11], [12], [13], [14]. These works follow the traditional bottom-up approach of verifying existing security policies, which cannot be used to automatically synthesize policies. Several studies have been done on attack graph based security configuration analysis [8], [15]. Few works (*e.g.*, [16], [17]) propose to find optimal deployment of security devices using attack graphs in order to block all attack scenarios. These works also cannot automatically find security or resiliency configurations based on the deployment budget or usability constraint.

Risk-based security hardening has received a lot of attention from the researchers. In [18], the authors present a methodology to model the composition of vulnerabilities as attack graphs. Singhal and Xou [19] describe the security metrics to compute the overall risk of a network. Ahmed et al. [20] present a framework of security metrics that objectively quantifies security risk factors. Although these works consider vulnerabilities to assess the network risk, none of them is capable for automatic synthesis of resiliency configurations.

The research on the automated security and resiliency configuration synthesis is still in a premature stage. Narain et al. propose ConfigAssure in [21], which takes security requirements and configuration variables and outputs the values of the configuration variables that satisfies the requirements. Zhang and Ehab [22] present procedural iterative approaches for generating firewall configurations considering the device deployment cost. In our previous work [2], we propose a formal model that can automatically synthesize isolation-based security configurations considering usability and cost together in a single model. Since it is important to consider different resiliency measures together, as well as the potential attack propagation at multiple levels, this work has limited contribution. In this work, we overcome this limitation by integrating diversity with isolation and modeling the interdependency between them, along with usability, deployment cost, and multiple level of potential attack propagation.

VIII. Conclusion

We present a formal framework for automatically synthesizing resiliency configurations, which includes isolation and diversity. We formally model the network configurations, traffic flows, diversity and isolation selections and their interdependency, deployment of security devices, and deployment cost. Then, we formalize the resiliency configuration synthesis process as the conjunction of resiliency requirements and business constraints. We implement the model and run it using an efficient SMT solver. The solution of this model provides resiliency configurations satisfying the requirements and constraints. We evaluate the proposed model using different synthetic networks and find that the proposed solution scales reasonably well with respect to the problem size. In the future, we would also like to consider other resiliency measures like agility, redundancy, and rejuvenation.

References

[1] L. de Moura and N. Bjørner. Satisfiability modulo theories: An appetizer. In *SBMF*, pages 23–36, 2009.

[2] M.A. Rahman and E. Al-Shaer. A formal framework for network security design synthesis. In *IEEE ICDCS*, pages 560–570, 2013.

[3] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese. Modeling network diversity for evaluating the robustness of networks against zero-day attacks. In *ESORICS*, pages 494–511. Springer, 2014.

[4] B. Cox *et al.* N-variant systems: a secretless framework for security through diversity. In *USENIX Security*, volume 6, pages 105–120, 2006.

[5] J. Han. *Novel Techniques of Using Diversity in Software Security and Information Hiding*. 2012.

[6] J. Han, D. Gao, and R.H. Deng. On the effectiveness of software diversity: A systematic study on real-world vulnerabilities. In *DIMVA*, pages 127–146. Springer, 2009.

[7] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro. Analysis of operating system diversity for intrusion tolerance. *Software: Practice and Experience*, 44(6):735–770, 2014.

[8] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*, pages 273–, 2002.

[9] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *ACSAC*, pages 86–95, Dec 2003.

[10] X. Ou, W.F. Boyer, and M.A. McQueen. A scalable approach to attack graph generation. In *ACM CCS*, pages 336–345, 2006.

[11] H. Hamed, E. Al-Shaer, and W. Marrero. Modeling and verification of ipsec and vpn security policies. In *IEEE ICNP*, pages 259–278, 2005.

[12] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, and P. Mohapatra. Fireman: a toolkit for firewall modeling and analysis. In *IEEE Symposium on Security and Privacy*, pages 199–213, May 2006.

[13] C.C. Zhang, M. Winslett, and C.A. Gunter. On the safety and efficiency of firewall policy deployment. In *IEEE Symposium on Security and Privacy*, 2007.

[14] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. Elbadawi. Network configuration in a box: towards end-to-end verification of network reachability and security. In *IEEE ICNP*, pages 123–132, Oct 2009.

[15] S. Noel and S. Jajodia. Attack graphs for sensor placement, alert prioritization, and attack response. In *Cyberspace Research Workshop of Air Force Cyberspace Symposium*, 2007.

[16] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *ACM CCS*, pages 204–213, 2007.

[17] J. Homer and X. Ou. Sat-solving approaches to context-aware enterprise network security management. *IEEE JSAC*, 27(3):315–322, April 2009.

[18] L. Wang, A. Singhal, and S. Jajodia. Measuring the overall security of network configurations using attack graphs. In *IFIP WG 11.3 DBSec*, pages 98–112, 2007.

[19] A. Singhal and X. Ou. Techniques for enterprise network security metrics. In *CSIIRW*, pages 25:1–25:4, 2009.

[20] M.S. Ahmed, E. Al-Shaer, M. Taibah, and L. Khan. Objective risk evaluation for automated security management. *JNSM*, 19(3):343–366, September 2011.

[21] S. Narain, G. Levin, S. Malik, and V. Kaul. Declarative infrastructure configuration synthesis and debugging. *JNSM*, 16(3):235–258, 2008.

[22] B. Zhang and E. Al-Shaer. On synthesizing distributed firewall configurations considering risk, usability and cost constraints. In *CNSM*, pages 1–8, Oct 2011.