On Synthesizing Distributed Firewall Configurations Considering Risk, Usability and Cost Constraints

Bin Zhang
Collage of Computing and Digital Media
School of Computing
DePaul University,
Chicago, IL, USA
bzhang@cs.depaul.edu

Ehab Al-Shaer
Cyber Defense & Network Assurance (CyberDNA) Center
Department of Software & Information Systems
University of North Carolina, Charlotte
Charlotte, NC, USA
ealshaer@uncc.edu

***Abstract*—Firewalls are the most deployed security devices in computer networks. Nevertheless, designing and configuring distributed firewalls, which include determining access control rules and device placement in the network, is still a significantly complex task as it requires balancing between connectivity requirements and the inherent risk and cost. Formal approaches that allow for investigating distributed firewall configuration space systematically are highly needed to optimize decision support under multiple design constraints.**

The objective of this paper is to automatically synthesize the implementation of distributed filtering architecture and configuration that will minimize security risk while considering connectivity requirements, user usability and budget constraints. Our automatic synthesis generates not only the complete rule configuration for each firewall to satisfy risk and connectivity constraints, but also the optimal firewall placement in the networks to minimizes spurious traffic. We define fine-grain risk, usability and cost metrics tunable to match business requirements, and formalize the configuration synthesis as an optimization problem. We then show that distributed firewall synthesis is an NP-hard problem and provide heuristic approximation algorithms. We implemented our approach in a tool called *FireBlanket* that were rigorously evaluated under different network sizes, topologies and budget requirements. Our evaluation study shows that the results obtained by *FireBlanket* are close to the theoretical lower bound and the performance is scalable with the network size.

I. Introduction

Designing a usable distributed filtering architecture and configurations requires careful balancing between number of competing factors: risk, usability satisfaction and cost. As filtering device configurations are hardened to reduce risk, it might be unnecessary too restrictive to achieve satisfiable usability or too expensive from budget or cost-benefit ratio perspective. Therefore, optimum usable security must provide satisfiable usability and minimum risk and cost. The enterprise security operations are mostly expert-based and ad-hoc, which might create instability and insecurity in enterprise networks.

The increasing complexity of designing and managing security configurations due to conflicting factors, larger networks, and longer policies makes configuration errors highly likely. Misconfigurations can cause reachability problems, security violations, and network vulnerabilities. Recent studies have found that more than 62% of network failures today are due to network misconfiguration [2].

Recent research works focus on verifying and hardening network security configuration. For example, attack graphs creation and analysis to block attacks has been presented in various studies such as [13], [4], [8], and [16]. Also, global security policies verification has been studied in [1], [7], [2] and [11].

On the other hand, automatic synthesis of firewall configurations to generate a comprehensive distributed filtering architecture under competing design factors is still unaddressed problem. Reasoning about distributed filtering architecture and configuration in the context of risk, limiting spurious traffic, usability satisfaction, and business and budget requirements is a challenging problem. In this paper, we first identify and formalize security configuration factors in order to quantify usability satisfaction, risk and cost. We then formulate the distributed firewall synthesis problem (DFSP) as an optimization problem. The solution of this problem determines: (1) the optimal locations of firewalls in the network to enforce polices while minimizing spurious traffic, (2) the correct firewall rules that satisfy system requirements while minimizing risk. We show that this problem is NP-hard and present greedy heuristics approximation algorithm that is accurate, scalable and practical. As a result, we present a holistic configuration approach to optimize distributed firewall architecture and configuration by automatically balancing usability satisfaction, risk and cost, while minimizing unwanted spurious traffic in the network. The resulting tool (FireBlanket) can be used as decision support tool to create distributed firewall architecture on clean-slate networks (assuming only connected hosts/services) or to refine existing firewall configurations.

The remainder of this paper is organized as follows. In Section 2, we formalize the distributed firewall synthesis problem and show its complicity. We describe the approximation algorithm implemented in FireBlanket in Section 3. In Section 4, we evaluate the performance of FireBlanket through a set of simulation experiments and a real-life network case studies. Related works and conclusion are discussed in Sections 5 and 6, respectively.

Parameter	Definition
x_{mn}	Decision variable, represents whether link m, n has a firewall deployed.
V_i	The overall vulnerability of node i
r_{ij}^g	The risk of flow from i to j for service g
I_j	The impact if node j is comprised.
S_j	The usability satisfaction of node j.
w_{ij}^g	The importance of traffic between i and j for service g.
y_{ij}^g	Decision variable, represents whether traffic from node i to node j for service g is allowed. y_{ij}^g = 1 means allow.
f_{ij}^g	Service flow from node i to j for service g
q_{ij}^g	Connection requirement, represents whether the service flow from node i to j for service g should be allowed.
d_{ij}^g	Connection demand, represents the users's wish for the flow from i to j for service g, which can be denied for security consideration.
e_{mn}	is the cost of deploying a firewall on edge m, n.
B_{max}	Deployment budget threshold.
$\hat{S}_j$	Usability satisfaction threshold for node j.
o_{ij}^g	The distance the spurious traffic can travel from i to j before it is blocked.

TABLE I
PARAMETERS AND DEFINITIONS IN THE MODEL

II. SYSTEM MODEL AND PROBLEM FORMULATION

We can define the firewall synthesis problem as follows: given the a network topology, deployed network services, business connectivity requirements ($\mathcal{Q}$), connectivity demand $\mathcal{D}$, usability profile S and cost c, the goal is to calculate the *minimum subset of firewalls* that can be places in various links and *subset of permitted flows* ($\mathcal{Q} \cup \mathcal{D}$) to minimize *risk (R)*, while satisfying the *connection requirements ($\mathcal{Q}$)*, the deployment *budget (B)* and *usability satisfaction (S)* and spurious traffic limitation constraints.

A. Synthesis Metrics and Parameters

In our approach, the synthesis of the distributed security configuration is based on number of input parameters that are easy to obtain in practice. These parameters are defined in high-level specification and they include network topology, connectivity requirements, connectivity demands, and usability profiles.

The connectivity requirements ($\mathcal{Q}$) states (1) the required accessibility based on business mission (e.g., "the Web server can always be accessed via the Internet"), (2) service dependencies (e.g., "accessing the Web server requires accessing the DNS server"). The connectivity requirements must be satisfied by FireBlanket solution.

The connectivity demands ($\mathcal{D}$) represents all network flows the users wish to have besides the connection requirements. Unlike the connection requirements, some of the connectivity demands might be granted by FireBlanket based on security constraints. This means the connectivity demands are usually for non-critical network services (e.g., instant messaging). Therefore, allowing $\mathcal{D}$ connections will certainly improve user satisfaction, but denying them will not impact the business operation.

The usability profile ($\mathcal{S}$) is per user or class of users. It describes the services to be deployed and their priorities (i.e., ranking) from the user perspective. For example, for a student machine, Skype might be more important than Secure Shell (SSH), while it could be the opposite for a faculty machine. The user satisfaction in our system is computed based on usability profiles. The goal of FireBlanket is to satisfy the users' satisfaction constraints while minimizing the risk on the global system.

Both $\mathcal{Q}$, $\mathcal{D}$ and $\mathcal{S}$ can be created based on business policies, requirements and network service database, respectively. Moreover, the connection requirements and connection demands can also be defined by high level policy specification language similar as the language proposed in [19]. The budget is simply the maximum amount to be spent on firewall deployment, configuration and maintenance. For risk estimation, users will provide information about the critical assets on their network and the value of estimated damage. Using the estimated damage and service venerability score, risk can be estimated.

B. FireBlanket Framework

In this section, we describe our formulation for each factor considered in FireBlanket.

1) Formalization of Usability Satisfaction: Suppose there are G type of services in the network. Let f_{ij}^g represent the service flow from node i to node j for service $g \in G$. In FireBlanket, each end host j is associated with a usability profile. Each usability profile defines how many services are running at each host and the rank or importance of each service to that host. The higher the rank, the important the service is to the node. Let A_j^g denote the rank of service g in node j which is defined in usability profile, then the weight of service g (W_j^g) can be calculated by normalizing the rank value as follows:

$$W_j^g = \frac{A_j^g}{\sum_g A_j^g} \tag{1}$$

Each usability profile can further define the rank of each flow under each service. Let a_{ij}^g denotes the rank of flow f_{ij}^g in usability profile for service g at node j. The rank value a_{ij}^g can represent the address space of i (how many hosts in the virtual node i), the importance of source node i or the combination of these factors. For example, flows from larger subnet and more important source (the dean's office) should have higher ranking value. However, this can be customized to reflect the requirements of each individual user. Similarly, based on the rank value, the weight of this flow (w_{ij}^g) can be computed as follows:

$$w_{ij}^g = \frac{a_{ij}^g}{\sum_i a_{ij}^g} \tag{2}$$

For each service g running on node j, its service usability satisfaction decreases if service access is denied (e.g. firewall blocking traffic flows). Thus, we define the Service Usability Satisfaction (s_j^g) as the ratio of *granted* access (number of flows) by FireBlanket and the total *requested* access for service g of node j in the connectivity demands, $\mathcal{D}$. For instance, a service will have 100% usability of it can be reached by all nodes as requested in $\mathcal{D}$. We use $d_{ij}^g \in \mathcal{D}$ (0 or 1) to indicates whether the traffic flow of service g from node i to j is *demanded* in $\mathcal{D}$. We also use y_{ij}^g as decision variables to indicates whether the service flow from node i to j is *allowed* (via firewalls) by FireBlanket. We can compute the Service Usability Satisfaction (s_j^g) as follows:

$$s_j^g = \frac{\sum_i y_{ij}^g w_{ij}^g}{\sum_i d_{ij}^g w_{ij}^g} \tag{3}$$

Based to the individual service usability satisfaction, we can calculate the overall service usability satisfaction (S_j) of node j, which is the weighted summation of service usability satisfaction of each service:

$$S_j = \sum_g W_j^g s_j^g \tag{4}$$

2) Formalization of Risk: Risk has many contributing factors and it is sometimes infeasible to consider all of them simultaneously. However, minimizing any subset of these factors can improve security significantly. In our previous work [14], we consider service vulnerability based on history. There are several works have been proposed in the area of risk assessment [6], [14]. In this work, we present a risk metric using well-known factors: (1) the potential damage, I_j, when node j is compromised, (2) the overall vulnerability (V_i) of source node i (3) the vulnerability of service g at node j (v_j^g). I_j can be estimated from business mission. The individual service vulnerability v_j^g comes from Common Vulnerability Scoring System (CVSS) [15], which provides scores in the range 1 to 10, where the higher the value, the severe the vulnerability. The overall vulnerability V_i equals the maximal vulnerability of running service on node i, $V_i = max_g(v_i^g)$.

We can compute the potential damage node i can cause to node j ($\hat{r}_{ij}^g$) due to attack on node j through service g if node i is compromised. Here, $\hat{r}_{ij}^g = V_i v_j^g I_j$. The risk implies by a traffic flow is proportional with the overall vulnerability of source node (V_i) and the destination service vulnerability (v_i^g). Intuitively, V_i represents how possible attack can be originated from node i, and v_j^g represents how possible the attack can succeed at node j running service g. After firewall deployment, the residual risk (r_{ij}^g) can be defined as follows:

$$r_{ij}^g = V_i v_j^g I_j y_{ij}^g \tag{5}$$

In the rest of this paper, risk always means residual risk. Consequently, the aggregated risk on node j can be computed as:

$$R_j = max_{i,g}(r_{ij}^g) \tag{6}$$

3) Formalization of Deployment Cost: Deploying firewall at network link $\{m,n\}$ associates with a deployment cost c_{mn}, which can be computed as follows:

$$c_{mn} = x_{mn} e_{mn} \tag{7}$$

Here the decision variable x_{mn} represents whether there is a firewall deployed on link $\{m,n\}$, where $x_{mn} = 1$ means there is a firewall on link $\{m,n\}$. And e_{mn} stands for the cost of deploying a firewall on link $\{m,n\}$ which can be calculated based on device price, hardware installation and software configuration. Based on the network topology and traffic throughput, different firewalls may be deployed on different links. For example, backbone link should be deployed with enterprise level firewall. On the contrary, the end-user machines should be deployed with host-based firewall.

4) Formalization of Spurious Traffic: Spurious traffic refers to the traffic flow which travel portions of the its path to the destination before it is blocked by a firewall. Spurious traffic consumes network bandwidth and increase the network vulnerability to deny of service (DOS) and other possible attacks embedded in spurious traffic. Ideally, unwanted traffic should not be allowed to get into the network. But in practice, it is impossible to have firewall deployed everywhere. Thus, we need limit amount of spurious traffics in the network. In this work, we control limit the amount of spurious traffic by controlling the distance (hops) that the spurious traffic can travel in the network.We use o_{ij}^g to represent the distance that the spurious traffic can travel from node i to j of service g before it gets blocked by a firewall. We use P_{ij} to represent the set of links from node i to j.

$$P_{ij} = \{p_{ij}^1, p_{ij}^2, \dots, p_{ij}^{M_{ij}}\} \tag{8}$$

Where p_{ij}^τ represent the τth link along the path from node i to j. So o_{ij}^g can be get as follows:

$$o_{ij}^g = (1 - y_{ij}^g) \sum_{h=1}^{M_{ij}} \prod_{\tau=1}^{h} (1 - x_{p_{ij}^\tau}) \tag{9}$$

From equation 9 we can see that o_{ij}^g is calculated by counting how many links the flow can travel before it is blocked by the first firewall along the path. Here, $x_{p_{ij}^\tau}$ represents whether the τth link along the path from i to j has firewall deployed.

5) Usability Satisfaction Control: FireBlanket allows the administrator to control the usability satisfaction at three levels. The most fine-grained usability satisfaction can be achieved by defining the individual threshold ($\hat{S}_j$) for each node/user, $S_j \geq \hat{S}_j$. Based on the role (obligation and functionality) of each user in the organization, the users (service nodes) can be classified into different groups (e.g., faculties, students). Let N_l represents the set of nodes in group l, the administrator can define the service satisfaction threshold ($\hat{S}_{N_l}$) for the minimal usability satisfaction of group l. More generally, the usability satisfaction can be controlled globally across the network by defining threshold ($\hat{S}_{global}$), which is the minimal usability satisfaction for every node. Using these three levels of control, administrators can fine tune the FireBlanket with great flexility to achieve desired security configuration.

6) Formalization of FireBlanket Optimization Problem: Let B_{max} denotes the deployment budget, and $\hat{R}_j$ represents the risk threshold for node j, we can then formalize the FireBlanket Synthesis Problem as follows:

$$minimize \sum_i \sum_j \sum_g r_{ij}^g \tag{10}$$

subject to

$$S_j \geq \hat{S}_j \tag{11}$$

$$S_j \geq \hat{S}_{N_l} \; j \in N_l \tag{12}$$

$$S_j \geq \hat{S}_{global} \tag{13}$$

$$r_{ij}^g \leq \hat{R}_j, \forall \; i, j \tag{14}$$

$$\sum_{mn} c_{mn} < B_{max} \tag{15}$$

$$y_{ij}^g \geq q_{ij}^g \tag{16}$$

$$o_{ij}^g \leq T \tag{17}$$

$$\sum_{\{m,n\} \in P_{ij}} x_{mn} \geq 1 - y_{ij}^g \; for \; d_{ij}^g = 1 \tag{18}$$

Let q_{ij} denotes the connection and security requirements, $q_{ij} = 1$ means service flow from node i must reach node j. Note that satisfying the Connection Requirements avoids unrealistic solutions of FireBlanket synthesis problem such as blocking connection to/from the Internet. The usability satisfaction for individual users, group of users and all users are defined in equations 11, 12 and 13. The cost of deployment firewalls should be less than the deployment budget, which is shown in equation 15. Connection requirements are satisfied by equation 16. Spurious traffic is controlled by equation 17, T is the maximal hops spurious traffic can travel. And equation 18 links decision variable x_{mn} and y_{ij} together which means if traffic between i and j is blocked, then there must be at least one firewall deployed in the path from i to j. Additionally, the user can also define thresholds for many different groups or global minimal usability satisfaction.

C. Computational Complexity

Theorem 1: The optimal distributed filtering architecture and configuration problem (DFSP) is a NP-hard problem.

We can prove theorem 1 via a reduction from knapsack problem. The proof sketch can be found in appendix.

Algorithm 1 Heuristic Greedy Algorithm for DFSP

```
 1: for $f_{ij}^g \in F$ do
 2:     $\bar{s}_{ij}^g == W_j^g \frac{w_{ij}^g}{\sum_i d_{ij}^g w_{ij}^g}$
 3:     $r_{ij}^g = V_i v_j^g I_j y_{ij}^g$
 4: end for
 5: $F_D \leftarrow getMinRiskFlows$
 6: $L \leftarrow \emptyset$
 7: $E' \leftarrow \emptyset$
 8: for $f_{ij} \in F_D$ do
 9:     if for edge$\{m, n\}$, $dist(in) < T$ then
10:         $E' \leftarrow E' \cup \{m, n\}$
11:     end if
12: end for
13: while $F_D \neq \emptyset$ do
14:     choose $l \in E'$ which can maximize $\frac{|f_l \setminus F_B|}{e_l}$
15:     $F_D \leftarrow F_D \setminus f_l$
16:     $L \leftarrow L \cup l$
17:     $E' \leftarrow E' \setminus l$
18: end while
19: return $L$, $F \setminus F_D$
```

III. HEURISTIC ALGORITHMS

In this section, we propose a greedy heuristic algorithm to DFSP which is efficient and practical to implement in large-scale network. The algorithm take the network topology, connectivity demands, vulnerability of each service and overall vulnerability of each node, deployment budget and usability satisfaction threshold as input, output the link for firewall deployment and the service flows which should be allowed. Algorithm 1 illustrates our greedy algorithm. This algorithm solves the DFSP in 3 steps.

A. Heuristic Algorithm for DFSP

Step 1, Preprocessing: Each service flow f_{ij}^g has its contribution to the overall usability satisfaction of node j, represented by $\bar{s}_{ij}^g$. Based on the definition in Section II, $\bar{s}_{ij}^g$ can be calculated by:

$$\bar{s}_{ij}^g = W_j^g \frac{w_{ij}^g}{\sum_i d_{ij}^g w_{ij}^g} \tag{19}$$

This algorithm first calculates $\bar{s}_{ij}^g$ and risk r_{ij}^g for each service flow (line 1-4). Then, it will process the usability satisfaction thresholds(individual, group and global) provided by the user. The priorities of these thresholds in descending order are individual, group, global.

Step 2, Minimal Risk Flows Selection: In this step, the algorithm greedily selects the subset of service flows to be allowed in the network based on the usability satisfaction thresholds (UST). Let F_j^A denote the subset of service flows which will be allowed in the network to achieve minimal risk under usability satisfaction threshold for node j, F_j^D represents the denied subset of flows. Select allowed flows to satisfy the UST and achieve minimal risk is equal to select denied flows to achieve maximal risk reduction and

Algorithm 2 Heuristic Algorithm for Minimal Risk Flows Selection: getMinRiskFlows

```
$F_D \leftarrow \emptyset$
for Each node j do
    if No individual UST then
        if No group and global UST then
            $\hat{S}_j \leftarrow NULL$
        else if No Group UST then
            $\hat{S}_j \leftarrow \hat{S}_{N_l}$
        else
            $\hat{S}_j \leftarrow \hat{S}_{global}$
        end if
    end if
    $F_j^D \leftarrow \emptyset$
    $S_j \leftarrow 0$
    sort $r_{ij}^g$ in descending order
    while $S_j < 1 - \hat{S}_j$ do
        select top flow $f_{ij}^g$ in the list
        $F_j^D \leftarrow F_j^D \cup f_{ij}$
        $S_j \leftarrow S_j + \bar{s}_{ij}^g$
    end while
    $F_D \leftarrow F_j^D \cup F_D$
end for
return $F_D$
```

not violating UST. Thus, for each node, if UST exists, this algorithm sorts all service flows of this node based on their risk in descending order. Then it selects the first service flow from the flow list until there is no room for more flow to be blocked, which means

$$\sum_{f_{ij}^g \in F_j^D} \bar{s}_{ij}^g < 1 - \hat{S}_j \tag{20}$$

The algorithm for minimal risk flows selection is shown in algorithm 2

Step 3, Firewall Deployment: The problem of finding the subset of links to deploy firewalls which can cover all the flows to be blocked under given budget constraints can be proven to be NP-hard by the directly mapping from the minimum weighted hitting set problem [3]. Our approximation algorithm selects the link which has the maximal ratio of new cover flows over deployment cost at each step until all denied flows have been covered or deployment budget has been reached.

IV. Evaluation

In this section, we evaluate FireBlanket using both simulation and real case studies. First, we conducted many simulation experiments to evaluate the accuracy and scalability of our approximation algorithms with respect to the optimal solutions. To evaluate the performance of FireBlanket under real-life network environment, we uses FireBlanket to analyze the campus network at Depaul University and generate the security configurations under different budgets and usability satisfaction thresholds.

A. Accuracy and Scalability of the Heuristic Algorithm

Since our integer programming solver could not compute the optimal solutions for the large-size network, we use the LP relaxation of our problem to compute a crude lower bound on the optimal solution. The LP solution can be computed in reasonable time for relatively large networks by using MATLAB[10]. We conducted the experiments under different network sizes and service flow configurations with different firewall deployment budgets. We used GT-ITM [17] generate network topologies. Unfortunately, the GT-ITM topology does not provide traffic demand matrices for each topology generated. Therefore, we used the technique proposed in [5] to generate service flows. In these experiments, we assume each note has one service running. The service vulnerability is randomly assigned, $[1, 10]$. The impact for each end node is randomly selected from range 1 to 100. In the first set of experiments, we study the accuracy (deviation from optimal) of our approach as the number of service flows increases. The deviation is calculated as follows deviation $= \frac{\text{optimal-heuristic}}{\text{optimal}}\%$ We fix the network size (1000 end hosts) and topology, but change the number of demanded service flows from 200 to 3800. We limit the connectivity requirements to be maximum 10% of the service flows. The deployment budget is the fixed maximal value of device deployment. But the actual deployment cost is determined by the FireBlanket solutions. We should guarantee that every denied flow has one firewall on its path, and the total cost of all firewalls can not be larger than budget. In order to eliminate the impact of the firewall deployment budget on accuracy, we adopt high deployment budget which means we can deploy firewalls to block every service flow. In this experiment, we only set the global UST to be 0.4, without setting group and individual UST. The results in Figure 1-a show that the deviation from the lower bound slightly increases as the number of flows increases. But the deviation ratio is less than 10% for a reasonable large number of service flows. Also the accuracy of our algorithm almost remains constant as the network size changes. This is because our algorithm attempts to select the subset of flows that minimizes the risk, no matter if these flows running on small or large networks. Thus, the number of flows dominates the impact on performance.

In the next set of experiments, we study the impact of deployment budget on the accuracy of our heuristic. For each network topology, we fix the global UST, and flow configuration. We also change the firewall deployment budget levels: low, medium and high. The high deployment budget means for current network topology and service flows, we can deploy firewall to block all service flows. Medium budget means around 10% of the service flows will not be filtered by firewall under any deployment plan based on the budgets. Low budget means 30% of the flows are not filtered by firewalls. We evaluate how our proposed heuristic performs under various service flows configuration ranging from 200 to 3800. The result in Figure 1-b show that low deployment budget can increase the deviation because our heuristic approach allows

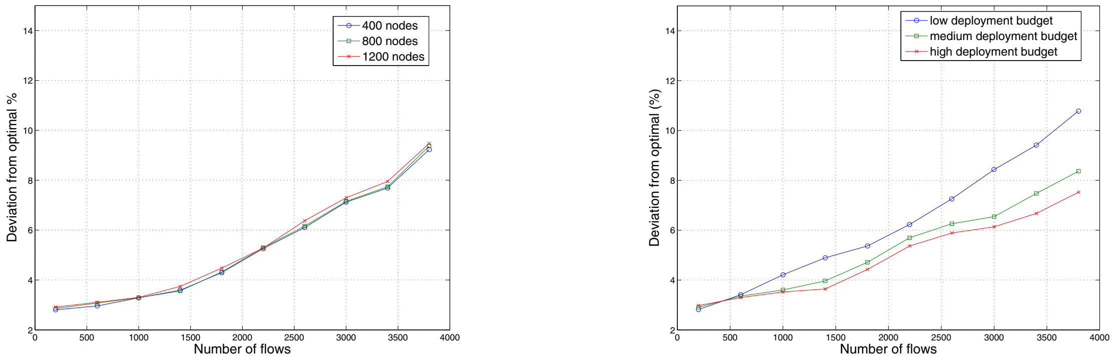

Fig. 1. Heuristic algorithm accuracy versus number of flows (a) the impact of network size on accuracy (b) the impact of deployment budget on accuracy

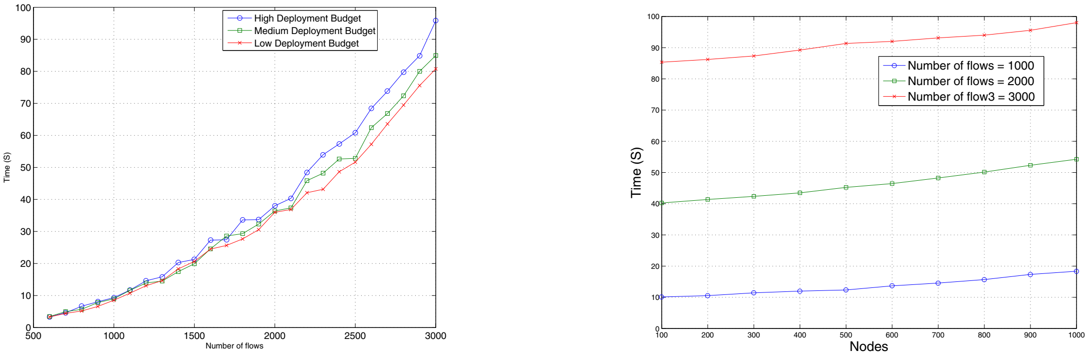

Fig. 2. FireBlanket performance (a) the impact of firewall deployment budget on finishing time (b) The impact of network size on finishing time

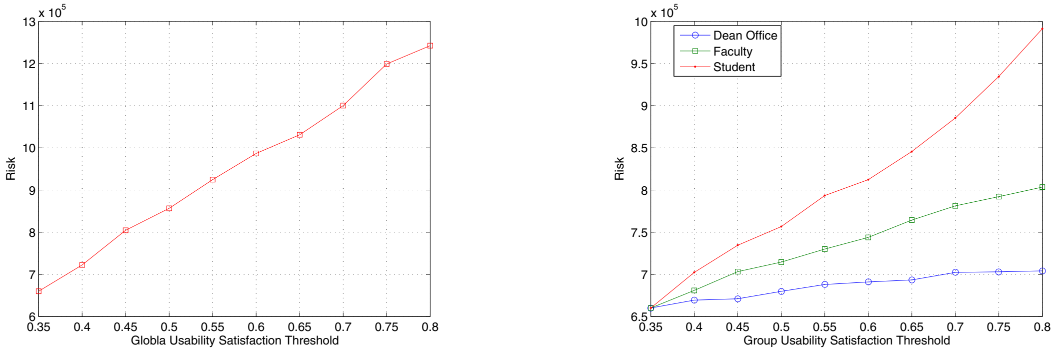

Fig. 3. The impact of usability satisfaction thresholds to risk (a) Global UST (b) Group UST

a set of flows, which should be denied, to transfer on the network due to low deployment budget.

In the third set of experiments, we study the scalability of our greedy heuristic algorithm. We implement FireBlanket using java and conduct the experiments on one PC with P4 2.4G cpu with 2GB memory. We conduct the experiments in a network with 1000 nodes. To study the impact of deployment budget, we fix the global UST and change the deployment budget in three different levels: Low, Medium, High. Figure 2-a shows the algorithm completion time as the number of network flows change. We can see that increasing the firewall deployment budget does not have much impact on algorithm completion time. And as the number of service flow increases, the algorithm can still run in reasonable time. In the fourth set of experiments, we evaluate the impact of network size on completion time. For each network topology, we have three service flow configurations 1000,2000 and 3000. We change the network size from 100 to 1000 end hosts. The result in Figure 2-b show that the execution time varies from few to 90 seconds which is practically a reasonable figure.

B. Case Study

We use the network of school of computing at Depaul University as case study. This network is composed of 12 different

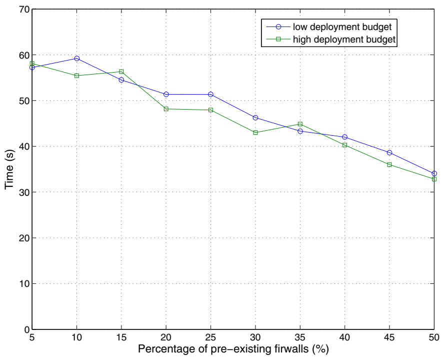

Fig. 4. Completion time versus pre-existing firewalls

subnets and 341 end host machines for faculties, staffs, and students. Based on the importance, responsibility and activity, these 341 end hosts are divided into 11 groups including: faculty, student, research lab, general staff, administrator, dean office, etc. The impact of each host is assigned based on its importance ranging from 1 to 100. We consider 15 common network services in this study and use the base score from CVSS as the vulnerability for each service. The connectivity requirements and demands are generated based on teaching and research activity. We first study the impact of different level of usability satisfaction threshold to the global risk. We set firewall deployment budget high to minimize its impact. We first study the impact of global UST (without group and individual UST) by increasing its value from 0.3 to 0.8 with an increase step of 0.05. The results in Figure 3-a show that the global risk increases as the global UST increases. Since the connectivity requirements must be satisfied, such connections represent the minimal risk and usability value in the system. This is the starting point of the curve in Figure 3-a. We test the impact of group UST to the risk by selecting three groups: faculty, student and dean office. We set the global UST=0.3 and change the each group UST from 0.3 to 0.8. The results in Figure 3-b show that adjusting the UST for different groups has different impact on the global risk. Also, increasing the student usability satisfaction will greatly increase the risk because students demand high risk services such as P2P file sharing, instant messaging with untrusted sites.

In order to evaluate the distributed firewall architecture provided by FireBlanket, we use FireBlanket to generate distributed filtering architecture under different budgets for testing networks while maintaining the usability satisfaction as the same as in the current configurations. We then generate the attack graphs [16] for both the current real network and FireBlanket configuration. We identify and compare the attack scenarios to compromise several critical assets in both the current and FireBlanket architecture. The results in Table II show that FireBlanket reduces the risk significantly by eliminating more than 75% of the attacks scenarios. It also shows that reducing budget might increase risk (attacks) but increasing it may not always result in reducing risk.

Distributed filtering architecture	Attack Scenarios
Current Network	128
FireBlanket (current budget)	31
FireBlanket (90% of current budget)	40
FireBlanket (110% of current budget)	17
FireBlanket (120% of current budget)	17

TABLE II
COMPARISON OF ATTACK SCENARIOS BETWEEN EXISTING ARCHITECTURE AND FIREBLANKET ARCHITECTURE

V. RELATED WORKS

Recently, there is a significant amount of research in the area of security configuration. In this section, we study two group of related works which focus on either security policy management or security configuration hardening. The security policy misconfiguration have been studies extensively in [1], [7], [20], [2]. In these approaches, the formal definition of configuration anomalies and safe deployment in single, multi firewall policies have been proposed and algorithms were presented to discover configuration inconsistency. These approaches can analyze existing policies, but they can not synthesize policies based on business requirement automatically. ConfigAssure [11] is the closest work to our approach. It uses a Requirement Solver that will take security requirements and configuration variables as input, and it produces values for the configuration variables in that make the requirements valid. If no feasible solution, it outputs a proof of unsolvability. This approach transforms the requirement into an equivalent quantifier-free form and relies on Kodkod, a SAT-based model-finder to find the value of each configuration variable. ConfigAssure requires complete and well define properties and it can not reason about the optimal configuration base on risk, usability and cost.

In [12], the author proposes a technique to place intrusion detection system (IDS) sensors and prioritize IDS alarms using attack graph analysis. The attack graphs predict the various possible ways of penetrating a network to reach critical assets. The IDS sensors are placed to cover all these paths. However, this approach does not allow for security hardening under budget and usability constraints. In [4], the authors model the problem of selecting a set of security hardening measures to minimize the residual damages in a predefined attack graph within certain budget. Unlike these approaches, FireBlanket does not require attack graph as input to synthesize configurations and it can be used on a clean slate network. In our previous work [18], [19], we show how to generate firewall rules automatically. However, FLIP [19] requires complete and well-defined filtering specs to generate firewall configurations, that do not consider risk, usability and cost requirements, and SecBuilder [18] focuses on DMZ creation using a preliminary simplified formulation with no control of spurious traffic and usability profiles.

VI. CONCLUSION AND FUTURE WORK

Automating synthesis of security configuration is one of the main standing challenges in today's networks. The manual or ad hoc design and configuration of networks does not only causes serious misconfigurations but it also does not allow for evaluating alternative configurations systemically for provable security. Furthermore, the complexity of distributed security configuration synthesis increases significantly as multiple contradicting constraints such as risk, usability satisfaction and cost are to be considered.

To the best of our knowledge, this work is the first approach to optimize distributed firewall architecture systematically, automate rules configuration synthesis and evaluate configuration alternatives quantitatively. The proposed system, called FireBlanket, determines the firewalls placement and rules to minimize risk while enforcing connection requirements, usability, cost and spurious traffic constraints. We formulate (1) *risk* based on potential propagation of infection, (2) *usability* based on user demand and satisfaction, and (2) *cost* based on deployment effort and then formulate the firewall synthesis problem as an IP optimization problem. We prove that firewall synthesis problem is NP-hard by a reduction from Kanpsack problem. We presented a greedy approximation algorithm for firewall synthesis and conducted rigorous evaluation experiments using both simulation and real-life case studies. Our evaluation study shows that our greedy approximation algorithm deviates from the lower bound by maximum 12%, even with large size network of 1000 end-hosts with 4000 network flows. We also show that the execution time is between order of seconds to few minutes for more than 1000 of end-hosts and many thousands of simultaneous flows.

Our current model only cover firewall deployment and configuration. The future work will focus on extend FireBlanket to IDS, IPSec and host-based security countermeasure such as software upgrade, system and application patching, host-based firewall and virtualization.

REFERENCES

[1] Ehab Al-Shaer, Hazem Hamed, Raouf Boutaba, and Masum Hasan. Conflict classification and analysis of distributed firewall policies. In IEEE Journal on Selected Areas in Communications (JSAC), 2005.

[2] Ehab Al-shaer, Wilfredo Marrero, Adel El-atawy, and Khalid Elbadawi. Network configuration in a box: Towards end-to-end verification of network reachability and security. In *International Conference on Network Protocols*, pages 123–132, 2009.

[3] A. Cincotti, V. Cutello, and F. Pappalardo. An ant-algorithm for the weighted minimum hitting set problem. Swarm Intelligence Symposium, 2003.

[4] Rinku Dewri, Nayot Poolsappasit Indrajit Ray, and Darrell Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. Proceedings of the 14th ACM conference on Computer and communications security, 2007.

[5] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. Proc. IEEE INFOCOM, 2000.

[6] A.K. Ganame and J. Bourgeois. Defining a simple metric for real-time security level evaluation of multi-sites networks. 4th int. Workshop on Security in Systems and Networks (colloaed with IPDPS'08), 2008.

[7] Hazem Hamed, Ehab Al-Shaer, and Will Marrero. Modeling and verification of ipsec and vpn security policies. in Proceedings of IEEE ICNP'2005, November 2005.

[8] John Homer and Xinming Ou. Sat-solving approaches to context-aware enterprise network security management. In IEEE JSAC Special Issue on Network Infrastructure Configuration, To appear.

[9] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons. ISBN 0-471-92420-2., 1990.

[10] Matlab. Mathworks. http://www.mathworks.com/.

[11] Sanjai Narain, Gary Levin, Vikram Kaul, and Sharad Malik. Declarative infrastructure configuration synthesis and debugging. Journal of Network and Systems Management, 2008.

[12] Steven Noel and Sushil Jajodia. Attack graphs for sensor placement, alert prioritization, and attack response. Cyberspace Research Workshop of Air Force Cyberspace Symposium, November 2007.

[13] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security*, 2006.

[14] Mohamed Salim, Ehab Al-Shaer, and Latifur Khan. Integrated risk evaluation for automated security management. Journal of Network and System Management (JNSM), to appear, 2011.

[15] M. Schiffman. A complete guide to the common vulnerability scoring system (cvss). http://www.first.org/cvss/cvss-guide.html, 2009.

[16] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. Proceedings of the IEEE Symposium on Security and Privacy, 2002.

[17] Georgia Tech. Modeling topology of large internetworks. http://www.cc.gatch.edu/fac/Ellen.Zegura/graphs.html.

[18] Bin Zhang and Ehab Al-Shaer. Towards automatic creation of usable security configuration. Proceedings of INFOCOM 2010, 2010.

[19] Bin Zhang, Ehab Al-Shaer, Radha Jagadeesan, James Riely, and Corin Pitcher. Specifications of a high-level conflict-free firewall policy language for multi-domain networks. In Proceedings of 12th ACM Symposium on Access Control Models And Technologies (SACMAT), June 20-22, 2007.

[20] Charles C. Zhang, Marianne Winslett, and Carl A. Gunter. On the safety and efficiency of firewall policy deployment. IEEE Symposium on Security and Privacy, May 2007.

APPENDIX

Proof sketch of Theorem 1:

Proof: The DFSP problem is proven to be NP-hard via a reduction from the knapsack problem [9], which is stated as follows. Given a set of n items, each represented by its volume V_i and price P_i and a knapsack with maximal carrying weight T_{max}, find a set of items with total weight at most T_{max} with maximum possible price. Considering a simplified DFSP problem by relaxing the deployment budget constraint. Then, the problem is to find the subset of service flows to be denied in the network to achieve minimal residual risk (maximal risk reduction) and satisfy usability satisfaction threshold. e can see that minimal global risk can be achieved by minimizing risk for each service, which is the sub problem of the original DFSP problem, we call it SDFSP problem. Given an instance of the knapsack problem, we create an instance of SDFSP problem as follows. First we create a create a bipartite graph G_B, each item i in knapsack instance with volume V_i and price P_i corresponds to one node m_i in one side of bipartite graph. Then we create a service node s in other side of the bipartite graph. We create a service flow between s and each node m_i with weight $w_{si} = P_i$. We set the impact of each node equal the volume of each item, $I_i = V_i$. Then we set the connection demand $d_{si} = 1$ and connection requirement $q_{si} = 0$. The UST reduction threshold $(1 - \hat{S}_j)$ is equal T_{max}. Clearly, through this reduction we can see that there exists a solution of total risk reduction K to the given SDFSP problem if and only if there exists a solution of total price K to the Knapsack problem. ■