Synthesizing Optimal Security Configurations for Enterprise Networks : A Formal Approach

S.K. Majhi[1], P. Bera[1], S. Kumar[2] and Ehab Al-Shaer[3], M. Satpathy[1]
[1]Indian Institute of Technology, Bhubaneswar, INDIA
[2]National Institute of Technology, Durgapur, INDIA
[3]University of North Carolina, Charlotte, USA
Email:{sm20@iitbbs.ac.in,plb@iitbbs.ac.in,swapnilaryan.nitdgp@gmail.com,ealshaer@uncc.com,manoranjan@iitbbs.ac.in }

***Abstract*—In this paper, we present NetSecSlider, an automated framework for synthesizing network configurations exploring various security and safety design alternatives. The design alternatives include distribution of different level of isolations (firewall, IPSec, etc.) and safety enforcement process (e.g. tampering of network flow) in the network. NetSecSlider takes the network topology, organizational security and safety requirements and business constraints as input, and synthesizes a correct and optimal security configuration. Finally, it determines the optimal placement of enabling devices in the network. The framework uses (i) a SMT solver for finding the correct and optimal security configuration and (ii) a method for determining the optimal placement of devices. The framework is evaluated on different networks with varying security and safety requirements.**

I. Introduction

In the recent past, the organizational requirements are becoming more fine-grained to protect resources from newly evolving cyber security/safety threats. In addition, there are business constraints like cost, usability demand etc. Thus, designing a stronger defense-in-depth in a network exploring different alternatives and resolving the contention between the security, safety and business constraints is a challenging problem.

The organizational security requirements include: (i) *connectivity* that defines the service flows between various network devices; and (ii) *isolation* that defines various access control rules as combination of different security enabling devices (firewall[3][8], IPSec[2], IDS[2], NAT[2], Proxy[2] etc.) and their relative order. In addition, the business constraints include *usability* and *cost*. The safety requirements include *tamper proof traffic flow*, *DMZ* and *Fault tolerance*. Deployment of different security devices may affect these constraints. For example, deploying both IPSec and IDS instead of a firewall might cause some usable application inaccessible from a node. Therefore a major challenge is to find a configuration exploring these design alternatives that increases the usability without significant degradation of security and safety.

In this paper, we present NetSecSlider, an automated framework for synthesizing network configurations using constraint satisfaction. The framework can be used as a decision support system to create optimal security and safety configurations on clean-slate networks, thereby providing better defense-in-depth.

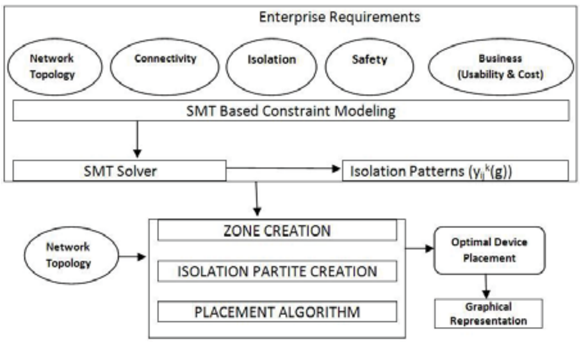


Fig. 1. NetSecSlider Framework

The rest of the paper is organized as follows: Section II presents the functional overview of NetSecSlider. The modeling of network topology and security/ safety requirements are described in Section III. Section IV presents the configuration synthesis problem. Section V presents our method for optimal device placement. Section VI presents our experimental results. Finally, Section VII concludes the paper.

II. NetSecSlider Overview

The NetSecSlider framework is shown in Figure 1. The salient features of our framework are:

- Modelling of network topology, security, safety and business constraints.
- Formalization/encoding of the configuration synthesis problem as a constraint satisfaction problem, which determines valid isolations in different segments of the network.
- Procedural determination of an optimal security device placement.
- Visualization of the synthesized configuration.

III. Modelling of Synthesis Parameters

Definition 1: [Network Topology Model] A network model is 2-tuple, $\langle N, L \rangle$, where,

- $N = NE \bigcup NR$; NE is a finite set of end point devices and NR a finite set of routers. Each node consists of a

set of ports. The ports are mapped to network services $g_1, g_2, .., g_n$ in the node.
- $L \subseteq N \times N$ is a finite set of links which define the interconnections between the network nodes.

We classify the enterprise requirements as: (i) security requirements; (ii) safety requirements and (iii) business constraints.

A. Formalizing Security Requirements

[A.1] Connectivity Requirement: Connectivity is formalized as a set of rules $\{r_1, .., r_n\}$ that represent the allowed services between different nodes. Each r_k defines the mapping between a predicate $P(service, source, destination)$ to a decision variable c (permit/deny), which is represented as :

$$r_k : (src = i) \wedge (dst = j) \wedge (service = g) \Rightarrow c_{ij}(g)$$

Here, $c^g_{ij} = 1$ represents that the service g is allowed between nodes (i, j). The organizational connectivity requirement is represented as:

$$Conn_{Req} \Leftrightarrow \bigwedge_i r_i$$

[A.2] Isolation Requirement: It defines different isolation patterns that can be deployed in the network. Administrators can define various isolation patterns as a single/combination of security device(s) and their relative order. For example, a firewall can be placed to allow/block the traffic flow between a pair of nodes; however, IPSec can be placed to ensure authenticated traffic flow. Moreover, both of these may be required to ensure authenticated and controlled traffic flow.

We consider different isolation primitives as in Table I. Administrators can define isolation patterns considering these primitive isolations. They may exclude some of the device combinations as per the requirements. A relative order between the devices can also be enforced.

We formalize network isolation as a set of rules, $\{ir_1, .., ir_n\}$, where, each ir_p is represented as:

$$ir_p : (src = i) \wedge (dst = j) \wedge (service = g) \Rightarrow y^k_{ij}(g)$$

Here $y^k_{ij}(g) = 1$ means that the kth isolation pattern is required to be deployed between nodes (i, j) under service g. Here, k represents the relative order of the isolation pattern. It shows that $k = 1$ for "firewall Deny" and $k = 3$ for "IPSec based authentication". So, if $y^1_{ij}(g) = 1$, then the service g must be denied between the node pair (i, j) through firewall.

TABLE I
PRIMITIVE ISOLATION LEVELS

Isolation Order	Isolation Pattern	Decision Variable
1	Firewall Deny	$y^1_{ij}(g)$
2	IPSec Encryption	$y^2_{ij}(g)$
3	Payload Inspection (IDS)	$y^3_{ij}(g)$
4	Source Identity Hiding(NAT)	$y^4_{ij}(g)$
5	Traffic Forwarding through PROXY	$y^5_{ij}(g)$

Isolation Composition: The primitive isolations from different classes can be combined to provide better defense-in-depth in the network. This may result in (i) possibility of $n!$ different isolation patterns (under n different primitives), and (ii) defining relative order of the composite isolation patterns. The number of isolation patterns can be reduced by considering a predefined number (say, at most 2) of isolation devices in each pattern and/or adding various constraints that exclude some of the combinations.

A set of invariants need to hold for maintaining the consistency between connectivity and isolation requirements.

[A.2.1] Isolation Invariants: These define the consistency between the connectivity and isolation requirements based on their functional behavior. We model such invariants as:

$$\textbf{[IVC1]:}(\neg c_{ij}(g) \Rightarrow y^1_{ij}(g)) \wedge (c_{ij}(g) \Rightarrow \neg y^1_{ij}(g))$$

$$\textbf{[IVC2]:}(y^1_{ij}(g) \Rightarrow \forall_{\,k \neq 1} \neg y^k_{ij}(g)$$

Here, **IVC1** states that if it is required to deploy "firewall Deny", then the corresponding connectivity must be *false*. **IVC2** states that if the isolation is "firewall Deny" for a specific service then other network isolation patterns must be *false*.

[A.2.2] User-defined Isolation Invariants: These constraints relate to organizational requirements, running services and criticality of the network. Administrators can define isolation between a pair of nodes based on some security policy. For example, isolation between nodes (X, D) can be defined as:

$$\neg y^1_{SD}(g) \Rightarrow y^1_{XS}(g) \wedge \neg(X = Internet)$$

It indicates that the firewall must allow the service g from source S to destination D, if no nodes from Internet is allowed to reach S.

[B] Safety Constraints: The main sources of safety hazard includes injection of physical fault, tampering of traffic flow or equipment and operational errors in the network. We consider the following safety enforcement processes: (i) DMZ (ii) Fault tolerance and (iii) tamper-proof traffic flow using combination of enabling devices. The DMZ enforces access control on specific segment of the enterprise network. If a node is faulty, it may disconnect an important segment of the network. To ensure safety, we introduce path-level redundancy while synthesizing device placement, keeping same functionality along each path.

A user provides a relative order between different isolation patterns. We assign an weight (between 0 and 1) to each isolation pattern accordingly. These weights are calculated by the ratio of isolation index with the total number of isolation patterns. The isolation pattern with maximum index has weight 1 and one with minimum index has weight $1/n$, n being the number of isolations. These weights map ranks of different isolation patterns into Boolean variables. We denote the weight of kth isolation pattern between nodes (i, j) under service g by $L^k_{ij}(g)$.

The isolation decision variables and the associated weights $L^k_{ij}(g)$ are used to define the overall *Isolation* between a given pair of nodes, which is represented as:

$$I_{ij} \Leftrightarrow \sum_{g} \sum_{k} y_{ij}^{k}(g) \times L_{ij}^{k}(g)$$

Note that if the corresponding service is not allowed (i.e., $c_{ij}(g) = 0$), then the effect of the associated isolation on the overall isolation will be zero. In that case, the isolation decision variables will be: $y_{ij}^{1}(g), \neg y_{ij}^{2}, \neg y_{ij}^{3}$ and so on.

B. Organizational Business Constraints

A higher isolation may provide stronger defense-in-depth in the network, but the deployment of such isolations may not be cost-effective. Therefore, resolving the contention between security and business constraints is an important need. The two business constraints are *Usability* and *Cost*.

[C.1 Usability:] The usability of different nodes is dependent on the rank of services and service flows in the network. Each service g in node N_i is given a rank $A_i(g)$. Similarly, each service flow $g(i,j)$ also assigned a rank $a_{ij}(g)$. Next, we introduce *Relative Service Weight* and *Relative Flow Rank* for computing usability.

Definition 2: [Relative Service Weight] The relative weight of a service g_x in a node N_i represents the importance of the service with respect to other services running in that node. This is represented as:

$$W_i(g_x) \Leftrightarrow \frac{A_i(g_x)}{\forall k, k \neq x \sum_{x} A_i(g_k)}$$

Definition 3: [Relative Flow Rank] The relative rank of a flow $g(i,j)$ for node j represents the importance of the flow with respect to the other flows $g(k,j)$ reaching to j under the same service g. This is represented as:

$$w_{ij}(g) \Leftrightarrow \frac{a_{ij}(g)}{\forall k, \sum_{k \epsilon N} a_{kj}(g)}$$

We use these weights, flow demand matrix and isolation decision variables to formalize the service level usability of a node for a specific service.

Definition 4: [Service Level Usability] Service level usability of a service g in a node N_j is represented as:

$$S_j(g) \Leftrightarrow \frac{\sum_{i \epsilon N} \sum_{k} b_{ij}^{k}(g) \times w_{ij}(g)}{\sum_{i \epsilon N} d_{ij}(g) \times w_{ij}(g)}$$

Here, $d_{ij}(g)$ represents the demand of the flow $g(i,j)$. The values of $d_{ij}(g)$ lie between 0 and 1. Note that the service level usability reduces as the isolation level increases. We introduce a weight parameter $b_{ij}^{k}(g)$ to represent the effect of different isolation patterns on usability. An example of this parameter under different primitive isolations could be:
$y_{ij}^{1} \Rightarrow b_{ij}^{1} = 0$; $y_{ij}^{2} \Rightarrow b_{ij}^{2} = 0.7$; $y_{ij}^{3} \Rightarrow b_{ij}^{3} = 1$
The network security administrator can easily determine these weights by calculating the time/effort (no of clicks) required to get a service access under different isolations.

Definition 5: [Node Level Usability] Node level usability is modeled as:

$$S_j \Leftrightarrow \sum_{g} W_j(g) \times S_j(g)$$

It represents the accumulated usability of a node considering all service flows to that node. The usability of the whole network is represented as:

$$U = \sum_{j \epsilon N} S_j$$

One of our goals in synthesizing the security configuration is to maximize usability.

[C.2 Cost:] We calculate the total cost of different isolation patterns for all node pairs in the network as:

$$C = \sum_{i \epsilon N} \sum_{j \epsilon N} \sum_{k} C_{ij}^{k}$$

Here the cost of deploying kth isolation between nodes (i,j) is: $C_{ij}^{k} = \sum_{g} y_{ij}^{k}(g) \times c^{k}$. One of our goals is to minimize C.

IV. Synthesis Problem as Constraint Satisfaction

The goal of our configuration synthesis problem is to maximize the overall isolation in the network satisfying various security requirements and organizational business constraints. Thus, it is represented as:

$$\forall i, \forall j, (I_{ij} > T^{I}) \wedge (Conn_{Req}) \wedge IVC1$$
$$\wedge IVC2 \wedge S_j \wedge (C < T^{C}) \wedge (U > T^{U}) (1)$$

Here, T^{C}, T^{I} and T^{U} represent the threshold on *cost*, *isolation* and *usability demand* respectively. Satisfaction of this predicate produces values of the isolation variables $y_{ij}^{k}(g)$ for all triplets (i,j,g). We solve this configuration synthesis problem using the Yices SMT solver [6].

V. Optimal Placement of Security Devices

The isolation result that we obtain using the Yices solver may not provide us an optimal device placement. For example, the solver may give us the configuration of Figure 2(a) in which there are three identical isolation patterns – identical device placement – between the single source and the three destinations, which is not optimal. One possible optimal configuration has been shown in Figure 2(b) – three destination nodes have been grouped into a single zone and so one isolation device is adequate.

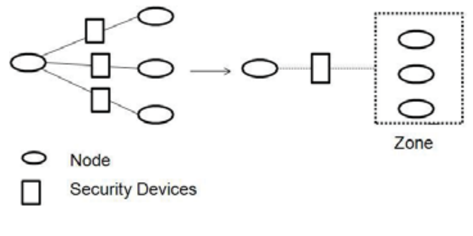


Fig. 2. Placement of Security Devices between node pairs

Here we present a systematic approach for solving the optimal device placement issue. The first step of our approach is to logically combine all nodes j with similar isolation pattern with respect to a specific node i into a single zone.

Definition 6: [Node Isolation Equivalence Zone] An *Isolation Equivalence Zone*, Z_i^k is defined as a logical collection of nodes having same isolation pattern k with respect to node i. This is represented as:

$$Z_i^k = \bigcup_{j \in N} \{j | y_{ij}^k = 1\}$$

We create these zones with respect to each node i in the network. Now, multiple isolation zones may contain overlapping member nodes. Thus, these zones should be further combined to group the nodes based on the notion of *isolation partite classes*.

Definition 7: [Isolation Partite Class] An *Isolation Partite Class*, S_{ix}^k is defined as a logical collection of isolation zones Z_x^k with respect to a zone Z_i^k such that $Z_x^k \subseteq Z_i^k$ and the node x is not in Z_i^k. It is represented as:

$$S_{ix}^k = Z_i^k \bigcup_x \{(Z_x^k | (Z_x^k \subseteq Z_i^k) \wedge \neg(x \in Z_i^k)\}$$

Each *isolation class* has two partites; the suffix sequence (here, $i, x \in N$) of the class S_{ix}^k represents the left partite, and the participating zones of class S_{ix}^k represent the right partite. After creating different partite classes, we procedurally determine the security device placement between different zones under the partite classes. The procedure is presented in Table II.

TABLE II
SECURITY DEVICE PLACEMENT ALGORITHM

```
Input: Zone number ($Z_i^k$), number of nodes ($N$) within the zone and
the isolation pattern between nodes.
Output: Optimal device placement between nodes.
1. done = 0;
2. while (! done)
3.   Create a sorted list $S'(s_1, s_2, .., s_N)$ based on descending order of $|S_{ix}^k|$ such that (each $s_i \equiv S_{x1x2..}$)
4.   Select first element $s_1$ from $S'$
5.   For $k = 1$ to $N$ Loop
6.     Place a $k^{th}$ level security device between the left partite $\{x1x2..\}$ and the right partite zones in $S_{x1x2..}$
7.     Remove the zones $\{Z_{x1}, Z_{x2}..\}$ from $S'(s_1, s_2, .., s_N)$
8.     Order the $k^{th}$ and $(k+1)^{th}$ device as per priority.
       (For sequencing Refer Table I)
9.   End Loop
10.  Remove all the empty sets from $S'$
11.  if ($S' = \emptyset$) then done = 1;
12.  end if
13. end while
```

We first create a sorted list S' of partite classes based on decreasing order of class size $|S_{ix}^k|$. Then, we select the first element s_1 (a partite class $S_{x1x2..}$) from S' and place a security device between the two partites of that class. After the corresponding device placement, we remove from all the classes, the zones that are already considered. This process is iterated until the list S' becomes empty. In this way, for each isolation pattern k, our framework derives an optimal device placement in the network.

We now explain our device placement method through an example. Let our isolation patterns be *firewall, IPSec, IDS* and the combination of *firewall* and *IPSec* (i.e. $k = 1, 2, 3, 4$ respectively). Consider the following isolation equivalent zones under a given isolation configuration:
$Z_1^{123} = \{3, 4\}$: firewall, IPSec and IDS are placed between nodes 1, 3 and 4.
$Z_2^{23} = \{4\}$: IPSec and IDS are placed between nodes 2 and 4.
$Z_3^1 = \{1\}$: only firewall is placed between nodes 3 and 1.
$Z_4^{12} = \{2, 3\}$: firewall as well as IPSec are required to be placed between nodes 4, 2 and 3.
Note that the functionality of IPSec subsumes that of a firewall; therefore, only IPSec device is adequate when both IPSec and firewall are required between the nodes of a zone.

Based on our partite class creation, the derived isolation classes for these zones are as follows:
$S_{12}^{23} = \{Z_1, Z_2\}; S_2^3 = \{Z_2\}; S_3^1 = \{Z_3\}; S_4^2 = \{Z_4\}$.
Now, the different iterations of the device placement procedure are as follows:
Iteration1: $S'(S_{12}, S_4, S_3, S_2)$; $(1, 2) \triangle \{Z_1 Z_2\} \sim (3, 4)$
Iteration2: $S'(S_4, S_3)$; $(4) \triangle \{Z_4\} \sim (2, 3)$
Iteration3: $S'(S_3)$; $(3) \triangle \{Z_3\} \sim (3, 4)$
Here, $x \triangle y$ denotes the placement of security device between x and y. It shows all the connectivity and security device placement of the required pattern. Figure 3(c) shows the graphical representation of the above example. In this way, the efficient device placement can be determined for other isolation patterns. Here, we show one possible security device placement. However, network administrator(s) can choose the isolation classes from the list in Table I based on the topology and different security metrics.

VI. EXPERIMENTATION & EVALUATION

We have used the Yices SMT solver to synthesize the security configuration. The device placement procedure has been implemented in C under Linux platform. GraphViz tool has been used to visualize the placement of security devices in the network.

We evaluate the NetSecSlider framework using various test cases; performance has been analyzed in terms of time complexity. The framework has been tested using 50 different test networks whose size ranges from 50 to 1000 nodes; and the constraints vary from 50 to 500 test requirements. The execution time ranges from 1.5 to 2 seconds.

Tables III and IV show the various security and business requirements respectively which are used to derive our test cases. We also generate numerous random test configurations through simulation with different isolation levels. We verify these configurations using realistic scenarios by measuring the attack surface (no. of machines infected) under each configuration. Further, we cross-validate the configurations

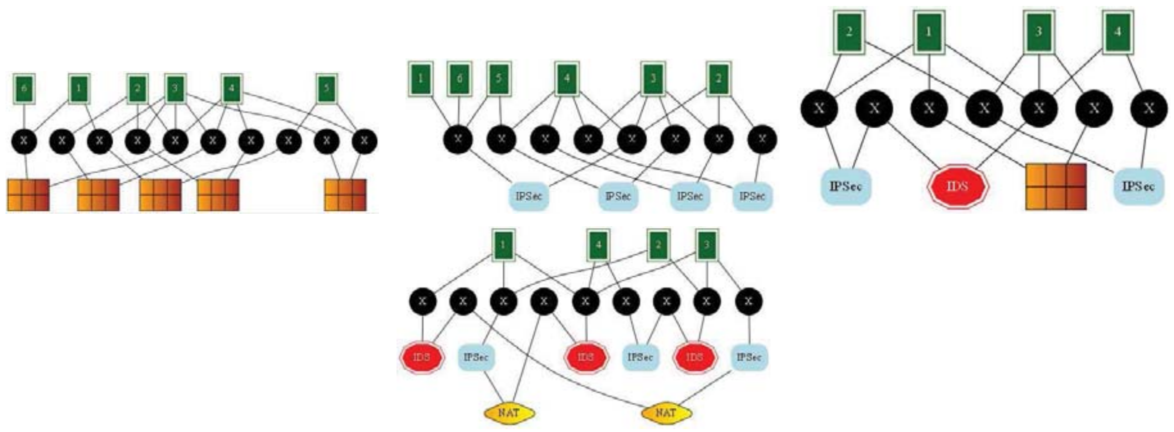


Fig. 3. Placement of Security Devices: (a) Test Case1: Synthesis for Firewall; (b)Test Case1: Synthesis for IPSec (c) Test Case3: combination of firewall, IPSec and IDS; (d) Test Case4: combination of NAT, IPSec, and IDS

TABLE III
ENTERPRISE SECURITY REQUIREMENTS (TEST CASES)

Cases	Connectivity			Isolation		
No.	Src	Dst	Service	Src	Dst	Device
1	1	(2,3,4)	http	1	(2,3,4)	Firewall
	2	(3,4)	ssh	2	(3,4)	Firewall
	3	(4,5)	https	3	(4,5)	Firewall
	4	(2,3)	ftp	4	(2,3)	Firewall
	5	(1,3)	ssl	5	(1,3)	Firewall
	6	3	http	6	3	Firewall
2	1	(2,3,4)	https	1	(2,3,4)	IPSec
	2	(3,4)	TLS	2	(3,4)	IPSec
	3	(4,5)	http	3	(4,5)	IPSec
	4	(2,3)	ssl	4	(2,3)	IPSec
	5	(1,3)	RPC	5	(1,3)	IPSec
	6	3	http	6	3	IPSec
3	1	(3,4)	https	1	(3,4)	Firewall +IPSec+IDS
	2	4	TLS	2	4	IPSec+IDS
	3	1	http	3	1	Firewall
	4	2	ssl	4	2	Firewall +IPSec
4	1	(3,4)	RPC	1	(3,4)	IPSec +IDS
	2	4	ssh	2	4	IDS+NAT
	3	1	http	3	1	Firewall+ IPSec+ IDS+NAT
	4	2	telnet	4	2	Firewall+ IPSec+ IDS

TABLE IV
ENTERPRISE SAFETY AND BUSINESS REQUIREMENTS (TEST CASES)

Cases	Safety		Usability				
No.	DMZ	SFT	Node1	Node2	Node3	Node4	Node5
1	(2,4,5)	5	-	25	-	10	30
2	(3,4)	(1,5)	5	-	15	35	-
3	(1,4)	2	15	30	-	20	-
4	3	(4,2)	-	10	50	30	-

by increasing the isolation levels and measuring the attack surface. Figure 3 presents the different device placements with respect to the test requirements in Table III and IV.

We evaluate the time complexity of our framework over different test networks with varying network size and security requirements.

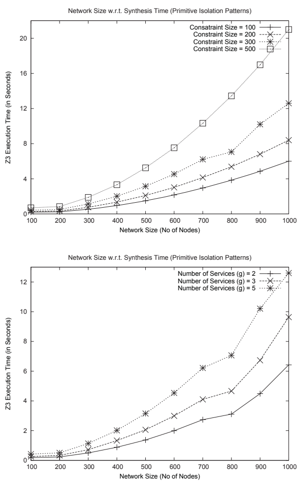
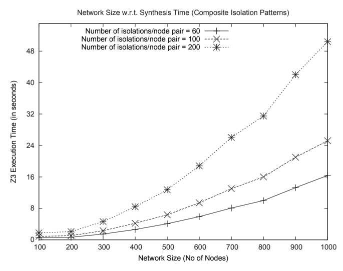


Fig. 4. Analysis under Primitive Isolation: (a) Synthesis Time vs. Network Size [g = 5]; (b) Synthesis Time vs. Network Size [Constraint Size = 300];

Configuration synthesis time (ST) includes model generation time and constraint verification time. ST depends on three parameters: (i) Network size (NS) - number of nodes; (ii) Constraint size (CS) - number of security and business constraints, and (iii) Number of Services (g). Figure 4(a) shows that ST varies quadratically with NS under a fixed CS and g. For a large-scale network (NS = 1000) with CS = 500, the runtime is around 21 secs. Figure 4(a) also shows that the ST varies linearly with CS. Figure 4(b) shows that ST varies linearly with g considering fixed NS and CS.



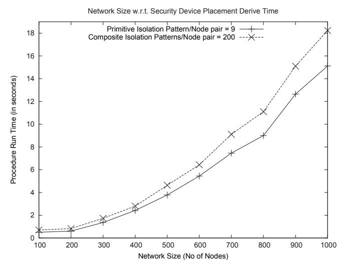


Fig. 5. Analysis under Composite Isolation: (a) Synthesis Time vs. Network Size [Const. size = 300, g = 5]; (b) Time vs. Network Size [g = 3]

Figure 5(a) shows the plot of ST vs. NS keeping CS and g fixed under composite isolation patterns. We observe that, for 200 different isolation patterns for 1000 nodes, 300 constraints and 5 network services, the synthesis time is approximately 51 seconds. This is reasonably good for large scale networks.

The device placement time has three time components: (i) *Equivalence zone creation*; (ii) *Partite class creation*, and (iii) *Placement time*. The complexity of the first component is $O(N * k)$, where, N = NS and k is the number of isolation patterns. Note that k is small with respect to N. On the other hand, the time complexity of the latter two components is $O(N^2)$. Thus, the overall device placement time is quadratically dependent on NS. Figure 5(b) shows that this time increases with the number of isolation patterns.

We also randomly generated different network topologies upto 1000 nodes with varying isolation constraints between different nodes to procedurally evaluate the variation of security devices required. In each of the cases, we found that the number of devices required is 10% of the total number of isolations in the network with a variance factor of $\pm 5\%$.

VII. Related Work

In the last two decades, a significant amount of research [2][3][9][1] have been done in formally verifying the network security configurations mainly in analyzing consistency of firewall configurations. These works follow traditional bottom-up approach of security analysis. However, such analysis sometimes leads to high state-space complexity for large networks. In addition, this approach requires repairing a sequence of misconfigurations iteratively to meet a specific requirement. There are few works on risk based security configuration analysis. For example, risk analysis using attack graphs have been proposed in [7] and [10]. Others have proposed using attack graphs to find optimal deployment of security devices to block all attack scenarios [11][12].

However, the research on security configuration synthesis is in a pre-mature stage. In [5], a procedural approach of generating distributed firewall configurations including device configurations was presented. However, this work only describes generation of firewall policy configurations and does not consider different defense-in-depth measures (firewalls, IPSec, IDS, Proxy, NAT, etc. and their combinations) in the context of security, usability satisfaction, and business constraints. In addition, this work does not show the optimal placement of security devices in a network. ConfigAssure [4] is a requirement solver that takes requirements and a configuration database with variables as input and outputs values of configuration variables that make the requirement true. It uses a SAT-based model finder to find the configuration variables. In [8], a procedural approach for generating access control policy configurations is presented. However, none of these works generate security design architecture or evaluate defense-in-depth. Moreover, these works do not explore various security design alternatives in determining satisfiable and cost-effective security configurations, which is the major thrust of this paper.

VIII. Conclusion and Future Work

We have presented an automated framework for synthesizing correct and cost-effective security configurations satisfying organizational configurations. The framework uses SMT constraint solving to derive the optimal security configuration. The framework also implements a procedure that takes the isolation results as input and determines an efficient placement of security devices based on isolation equivalence between different nodes. The complete framework has been implemented and tested with synthesized test cases. We have also planned to address the extensibility of the framework due to changes in the requirements and network topology.

References

[1] E. S. Al-Shaer and H. H. Hamed. *Discovery of Policy Anomalies in Distributed Firewalls.* INFOCOM, pp.2605-2626, Hong Kong 2004.

[2] E. Al-Shaer, W. Marrero, A. El-Ataway and K. ElBadani. *Network Security Configuration in A Box: End-to-End Security Configuration Verification.* In ICNP'09, pp. 123-132, Princeton, NJ, October 2009.

[3] L. Yuan, J. Mai, Z. Su, H. Chen, C. Chuah, and P. Mohapatra. *FIREMAN: A Framework kit for Firewall Modeling and Analysis.* In IEEE Symp. on Security and Privacy, Oakland, USA, May 2006.

[4] S. Narain, G. Levin, V. Kaul and S. Malik. *Declarative Infrastructure Configuration Synthesis and Debugging.* In Journal of Network System and Management, vol. 16 (3), pp. 235-258, 2008.

[5] B. Zhang and E. Al-Shaer. *Towards Automatic Creation of Usable Security Configuration.* In IEEE INFOCOM'10 Mini-conference.

[6] L. de Moura and N. Bjorner, *Z3: An Efficient SMT Solver*, Conference on TACAS, Budapest, Hungary, 2008.

[7] X. Ou, W. F. Boyer and M. A. McQueen. *A scalable approach to attack graph generation.* In 13th ACM CCS, 2006.

[8] P. Bera, S. Maity and S. K. Ghosh. *Generating Policy based Security Implementations in enterprise network: a formal framework.* In 3rd ACM SafeConfig workshop'10, ACM CCS, Chicago, USA, 2010.

[9] P. Bera, S. K. Ghosh and P. Dasgupta. *Policy based Security Analysis in Enterprise Networks -A formal approach* In IEEE Tr. on Network and Service Management, vol. 7(4), pp. 231-243, 2010.

[10] R. Dewri, N. Poolsappsi, I. Ray and D. Whitley. *Optimal security hardening using multi-objective optimization on attack tree models of networks.* In 14th ACM CCS 2007.

[11] J. Homer and X. Ou. *Sat-solving approaches to context-aware enterprise network security management.* In IEEE JSAC Special Issue on Network Infrastructure Configuration, 2011.

[12] I. Kotenko and M. Stepashkin. *Attack graph based evaluation of network security*. In ICCMS, 2006.