A Formal Approach for Network Security Management Based on Qualitative Risk Analysis

Mohammad Ashiqur Rahman and Ehab Al-Shaer
CyberDNA Research Center, University of North Carolina Charlotte, USA
Emails: {mrahman4,ealshaer}@uncc.edu

***Abstract*—The risk analysis is an important process for enforcing and strengthening efficient and effective security. Due to the significant growth of the Internet, application services, and associated security attacks, information professionals face challenges in assessing risk of their networks. The assessment of risk may vary with the enterprise's requirements. Hence, a generic risk analysis technique is suitable. Moreover, configuring a network with correct security policy is a difficult problem. The assessment of risk aids in realizing necessary security policy. Risk is a function of security threat and impact. Security threats depend on the traffic reachability. Security devices like firewalls are used to selectively allow or deny traffic. However, the connection between the network risk and the security policy is not easy to establish. A small modification in the network topology or in the security policy, can change the risk significantly. It is hard to manually follow a systematic process for configuring the network towards security hardening. Hence, an automatic generation of proper security controls, e.g., firewall rules and host placements in the network topology, is crucial to keep the overall security risk low. In this paper, we first present a declarative model for the qualitative risk analysis. We consider transitive reachability, i.e., reachability considering one or more intermediate hosts, in order to compute exposure of vulnerabilities. Next, we formalize our risk analysis model and the security requirements as a constraint satisfaction problem using the satisfiability modulo theories (SMT). A solution to the problem synthesizes necessary firewall policies and host placements. We also evaluate the scalability of the proposed risk analysis technique as well as the synthesis model.**

I. Introduction

Today information security has become very crucial to an organization for the successful operations of different mission critical applications. Risk management plays a critical role in protecting an organization's information assets from security risks [1][2]. Correct risk analysis techniques are important for adopting appropriate countermeasures against different security threats to the organizational resources. The ever-changing views of threats and vulnerabilities make the understanding of risk, its analysis and management difficult. Therefore, it is essential to develop an efficient risk analysis tool that can cope with the changes in risk-understanding and can also assist the IT professionals in sharing simple and comprehensive view of potential risks according to different missions.

Most of the organizations emphasize enforcement of the security constraints, but they suffer from resource limitations. Hence, it is important to find the network configuration within this limited resource, which offers expected security performance. Providing a stronger security in a network by exploring potential risk as well as resolving the contention between the security and business constraints (i.e., connectivity requirements) is an important but challenging problem.

Contribution. Our contribution in this paper is twofold:

- *Qualitative risk analysis*. Given the vulnerability and assets of the host, and the network topology, we present the qualitative risk analysis using declarative logic. We consider reachability between the hosts in order to use attack paths in threat computation.
- *Synthesis of firewall policies and host placements considering constraints on risk*. We We present an approach of synthesizing firewall policies and host placements in the network topology that satisfy reachability requirements and one or more risk-based constraints. A risk-based constraint can set a limit on the overall risk of the network or it can be based on a risk mitigation policy. We formulate the synthesis problem as a constraint satisfaction problem using the satisfiability modulo theories (SMT).

The justification of using declarative logic and SMT in this research follows from their utilities. Declarative logic (e.g., Prolog [14]) gained its popularity because of two reasons: flexibility and expressiveness. In order to have the ability to cope with the changes of risk concept, declarative language is an easy choice for the modeling. Over a decade, SMT has proved very useful for modelbased testing and automated synthesis and planning. Modern SMT solvers can check formulas with thousands of variables, and millions of logical clauses [16]. *Uninterpreted function* is one of the major attractions of SMT. In most cases, the predicates used in declarative logic are easy to convert in SMT using uninterpreted functions.

The rest of this paper is organized as follows. In Section II, we present the declarative risk analysis. In that section, we also present the evaluation of the risk analysis technique. In Section III, we describe the synthesis of firewall policies and host placements. In Section IV, we discuss the related work. In Section V, we conclude the paper.

II. Risk Analysis

The *risk* of a host is considered as a function of *impact* and *threat* [1][2]. Potential security threat that a host may face depends on the exposure of its *vulnerabilities*. We use the term *exposure* for characterizing possible attack paths to a host. A number of matrices are used to qualitatively define different parameters towards the risk assessment. The use of these matrices in the risk analysis process is shown in Fig. 1.We

978-3-901882-50-0

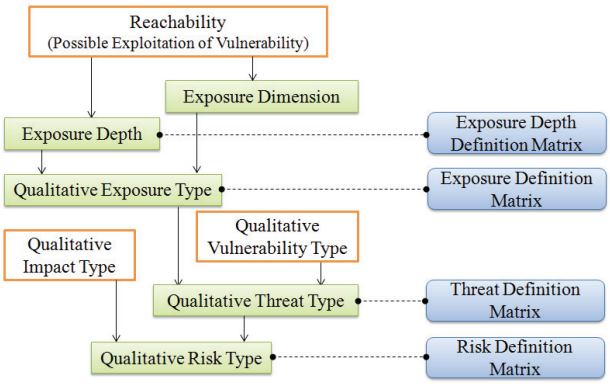


Fig. 1. The uses of matrices in the risk analysis process.

start this section with a very brief discussion about the traffic reachability model, which is similar to [3].

A. Network configuration

Each participant in a network is a node. Usually, a node can be either a host or a router. We define two predicates: `host(N)` and `router(N)`, to denote a node `N` as a host or a router, respectively. The predicate `link(N,N2)` signifies that a node `N` is physically connected to a node `N2`. We write a traffic flow in the form of '`<source,destination,service>`' (i.e., `<S,D,Sv>`). Each router maintains a routing table and the predicate `forward(N,S,D,Sv,T)` specifies that the router `N` forwards the traffic `<S,D,Sv>` to the next hop `T`. We consider firewalls as security enforcement devices in the network. A firewall checks the incoming traffic and forwards the traffic to the next hop according to its policy. The predicate `firewall(N)` denotes that the node `N` is a firewall. The predicate `firewallAllow(N,S,D,Sv)` specifies that the firewall `N` allows the traffic `<S,D,Sv>` to its destination. The predicate `canReach(S,D,Sv)` finds whether the traffic `<S,D,Sv>` is reachable from its source to the destination. We also define `reachable(S,D)` from `canReach(S,D,_)` to find the traffic reachability from the source to the destination irrespective of the service.

B. Impact and Vulnerability

We define the vulnerability of a host qualitatively, e.g., high, medium, and low, according to the CVSS scores of the host's vulnerabilities [9]. The CVSS scores are given in an integer scale of $0-10$. A host may have different vulnerabilities which usually have different CVSS scores. We take the highest score for defining the vulnerability of the host.

The impact of a host refers to the potential magnitude of the damage if an attack is launched against it. Here, we measure impact, simply, based on the asset value of a host. The asset value of a host represents its importance based on the organizational requirements. These asset values are assumed to be normalized in an integer scale of $0-10$. We define the predicate `hostInfo(N,V,A)` to express the highest vulnerability score (`V`) and the asset value (`A`) of a host `N`.

A vulnerability type `T` is defined using the predicate `vulnModel(T,Mn,Mx)`, i.e, if the vulnerability score is within a semi-open range [`VMn`, `VMx`), the vulnerability type is `T`. The predicate `vulnerability(N,T)` finds the vulnerability type `T` of a host `N`. The model for vulnerability computation is presented in Listing 1, where we take `high`, `medium` and `low` as the vulnerability types. The predicate `vulnModel(T,VMn,VMx)` is used to map the vulnerability score to a qualitative type.

```
vulnModel(high,7,10).
vulnModel(medium,4,6).
vulnModel(low,0,3).

vulnerability(N, T):-
  vulnModel(T,Mn,Mx), hostInfo(N,V,_), V>=Mn, V<Mx.
```

Listing 1. Vulnerability measure

Similar to the vulnerability modeling, we measure the impact type `T` of a host `N` using the predicate `impact(N,T)` from the asset value. We use the predicate `impactModel` to map the asset value to a qualitative type.

C. Exposure

We derive a query to find the *transitive exploitation of a host's vulnerability* from another host based on reachability. It means that exploitation of a vulnerability of a host can cause exploitation of a vulnerability of another host.

```
findVulnTransition(S,D,_,K,[D]):-
  K>=0, malicious(S), reachable(S,D).
findVulnTransition(S,D,PF,K,[D|P]):-
  K>0, not(reachable(S,D)), host(N),
  not(member(N,PF)), reachable(N,D), K2 is K-1,
  findVulnTransition(N,D,[N|PF],K2,P).

vulnTransition(S,D,K,P):-
  findVulnTransition(S,D,[S,D],K,P).
```

Listing 2. Transitive exploitation of vulnerability

We define transitive exploitation of vulnerability (see Listing 2) from a host `S` to a host `D` as '`S` can reach `D` in one or more steps. For example, `S` can exploit a vulnerability in a host `N`, while `N` can exploit a vulnerability in `D`; i.e., `S` can transitively exploit a vulnerability in `D` in two steps. The query `vulnTransition(S,D,K,P)` finds if `S` can exploit a vulnerability in `D` taking at most `K` steps. `P` shows the attack paths. Our query implementation is loop-free. Predicate `findVulnTransition` captures the transitive path excluding the hosts in `PF` and considering remaining steps `K`. Query `vulnTransition` invokes `findVulnTransition` with `PF` as empty. The query `allVulnSrc(D,K,SL)` finds the list of sources `SL` that can exploit vulnerabilities in `D`, taking at most `K` steps.

The *exposure* of a host is the qualitative measurement of the extent to which the host is exposed to, i.e., reachable by other (potential malicious/compromised) hosts (e.g. the Internet). The potential propagation of exploitation from a host to another host is expressed by the exposure. The number of steps (i.e., the number of intermediate hosts) required for

TABLE I
AN EXAMPLE OF EXPOSURE DEPTH DEFINITION MATRIX

		Path security		
		ipsec	ids	none
Qualitative steps of vulnerability transition	low	low	medium	high
	medium	low	low	medium
	high	low	low	low

TABLE II
AN EXAMPLE OF QUALITATIVE EXPOSURE DEFINITION MATRIX

		Qualitative exposure dimension		
		high	medium	low
Qualitative exposure depth	high	high	high	high
	medium	high	medium	medium
	low	medium	low	low

TABLE III
AN EXAMPLE OF QUALITATIVE THREAT DEFINITION MATRIX

		Qualitative vulnerability		
		high	medium	low
Qualitative exposure	high	high	high	medium
	medium	medium	medium	low
	low	medium	low	low

TABLE IV
AN EXAMPLE OF QUALITATIVE RISK DEFINITION MATRIX

		Qualitative impact		
		high	medium	low
Qualitative threat	high	high	high	medium
	medium	high	medium	low
	low	medium	low	low

the vulnerability transition has an effect on the exposure, i.e., *the more steps that are required by a malicious host, the less is the possibility of exploitation*. We calculate the exposure of a host considering two metrics: *exposure depth* and *exposure dimension*. Exposure depth of a host is computed from two parameters: (i) the qualitative steps that a malicious host requires to reach it, and (ii) the security properties (i.e., existence of IPSec, IDS, etc.) of the communication path between them. The definition of an exposure depth type follows from the matrix as shown in Table I. Exposure dimension of a host is the number of other hosts that can reach this host. An exposure type is defined from the qualitative types of exposure depth and exposure dimension. We define `exposure` using the predicate `exposureModel`, which resembles each element of the matrix, as shown in Table II, generated from exposure depth types (in rows) and exposure dimension types (in columns). We take the same qualitative types, `high`, `medium`, and `low`, for all of these metrics.

```
getExposureDimension(N,ES):-
  maxExposureIndirection(K);
  allVulnSrc(N,K,L),length(L,ES).

getExposureDepth(N,EE,P):-
  maxExposureIndirection(K),EE<=K,
  malicious(S),vulnTransition(S,N,EE,P),
  not(getExposureDepth(N,EE2),EE2<EE).

hasExposure(N,EDT,EST):-
  getExposureDimension(N,ES),
  getExposureDepth(N,EE,P),
  exposureStepModel(EET,EEMn,EEMx),
  EE>=EEMn,EE<EEMx,pathSecurity(P,PT),
  exposureDepthModel(EDT,EET,PT),
  exposureDimensionModel(EST,ESMn,ESMx),
  ES>=ESMn,ES<ESMx.

exposure(N,T):-
  exposureModel(T,EDT,EST),
  hasExposure(N,EDT,EST).
```

Listing 3. Qualitative exposure measure

The predicate `exposureModel(T,EDT,EST)` specifies that when the exposure depth type is `EDT` and the exposure dimension type is `EST`, the exposure type is `T`. The predicate `hasExposure(N,EDT,EST)` finds `EDT` and `EST` of the host `N`. The predicate `getExposureDimension(N,ES)` finds the number of hosts `ES` that can reach `N`. The predicate `getExposureDepth(N,EE,P)` finds the exploitation steps `EE` and the path quality `P`. The maximum number of steps to be considered for vulnerability transition is bounded by `maxExposureStep`. The predicate `exposure(N,T)` finds the exposure type `T` of a host `N`.

D. Threat

A threat type is defined from the qualitative exposure type and the qualitative vulnerability type of a host. The predicate `threatModel(T,ET,VT)` defines the threat type `T` based on the exposure type `ET` and the vulnerability type `VT`. The definition of a threat type follows from the *threat definition matrix* (as shown in Table III) which is formed by taking exposure types in rows and vulnerability types in columns. We define `threat(N,T)` to find the potential threat type of a host `N`. This predicate invokes the predicates `exposure` and `vulnerability` and identifies the threat type utilizing the predicate `threatModel`.

```
threat(N,T):-
  threatModel(T,ET,VT),
  exposure(N,ET),vulnerability(N,VT).

risk(N,T):-
  riskModel(T,TT,IT),
  threat(N,TT),impact(N,IT).
```

Listing 4. Threat and risk measure

E. Risk

The risk of a host is defined as a function of its potential threat (exploitation of its vulnerability) and the resulting impact of that adverse event. In order to define the risk types, we utilize the *risk definition matrix* by taking threat types in rows and impact types in columns. We show an example of such a matrix in Table IV. The predicate `riskModel(T,TT,IT)` (as shown in Listing 4) states that if the threat type is `TT` and the impact type is `IT`, the risk type is `T`. The predicate `risk(N,T)` finds the risk type `T` of the host `N`.

The query `allRisk(HL,T)` finds all the hosts `L` that have the same risk type `T`. This query is useful for an administrator to know the risk types of different hosts in a network. The query `riskScore` finds the risk score of the network by considering a (quantitative) risk value for each host based on the host's (qualitative) risk type. For example, the risk scores of the high, medium, and low risk types can be taken as 1, 0.5,

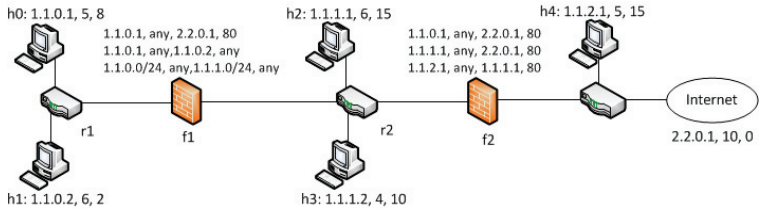
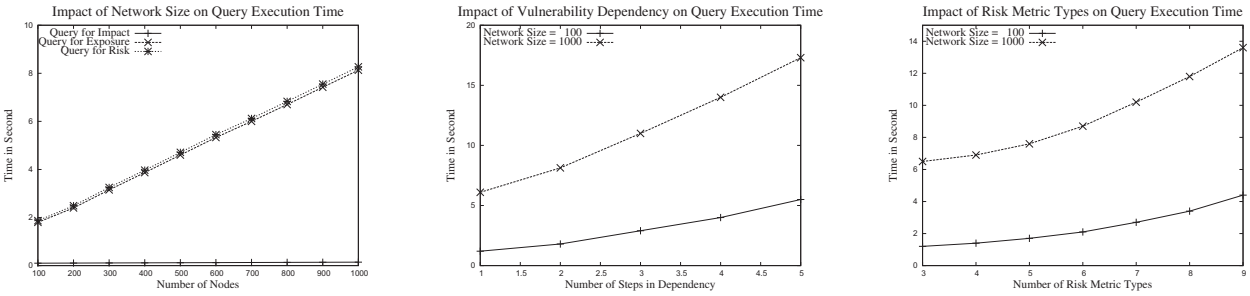


Fig. 2. An example network topology



(a) (b) (c)

Fig. 3. Impact of (a) network size, (b) the number of steps (transitive exploitation), and (c) the number of risk metric types over analysis time.

and 0, respectively. The query finds all the hosts with different risk types, sums the risk scores of these hosts, and normalizes the aggregated value based on maximum risk value (when all the hosts have high risk). This query specifies the risk that resides in the network, which promotes the administrator to do necessary reconfigurations in the policy or topology to keep the risk within an accepted level.

F. Example

Here, we show an example of the application of our proposed risk analysis model using a small test network as shown in Fig. 2. The hosts' information (i.e., the IP address, vulnerability score and asset value of each host) and the firewall rules (allow rules only) are shown in the figure. We consider the same aforementioned risk model. With this setting, the query `allRisk(L,high)` returns the list `[h2,h4]` as `L`; i.e., the hosts `h2` and `h4` have high risk. The invocations of `allRisk(L,medium)` and `allRisk(L,low)` return `[h0,h3]` and `[h1]`, respectively.

G. Evaluation of Declarative Risk Model

We write a program to generate the predicates for a given network topology and security policies. We randomly create the core network of routers and firewalls, as shown in Fig. 2. Each router is connected with one or more subnets. Each subnet consists of a number of hosts. The routing entries of each router are generated by following the shortest path from the router to the destination. Firewall policies are generated randomly. We run our queries using SWI Prolog [14] in a machine equipped with an Intel Core i3 Processor and 4 GB memory under Windows 7 OS platform.

We evaluate the impact of the network size on the execution time of different queries in Fig. 3(a). We consider the network size as the number of hosts. In the experiments, we vary the number of hosts, while the core network remains the same. Core network consists of 20 routers. We consider three qualitative types for each risk definition metric and at most three steps (i.e., `K` is 3) for transitive exploitation. We take three queries for evaluation: `vuln`, `exposure`, and `risk`. We observe from the graphs that the increase in the execution times in the cases of the last two queries are linear with the increase of the network size, while the time in the case of the first query almost remains the same. The computing time of the exposure is almost equal to that of the risk, i.e., the exposure computation time covers most of the risk computation time.

The risk analysis time is evaluated with respect to the maximum steps (i.e., `K`) that considered for transitive exploitation. Fig. 3(b) shows that if the number of steps increases, execution time increases almost linearly. We also evaluate the analysis time changing the number of risk metric types. The results are shown in Fig. 3(c). We find that execution time increases quadratically, if the number of metric types increases. This is due to the reason that with the increase of metric types (i.e., the rows and columns of different metrics' definition matrices), the number of predicates increases quadratically.

III. Risk-Based Security Policy Synthesis

We apply our proposed risk analysis model for synthesizing the network security policy. In this synthesis process, we consider a number of constraints that keep the risk in the network low.

A. Synthesis Goal

Our security policy synthesis has two parts: the firewall policy synthesis and the host placement synthesis. In the firewall policy synthesis, firewall rules are generated for each firewall based on the satisfaction of given constraints. All possible traffics between the hosts of the network are considered in the firewall policy. Our firewall policy synthesis goal is to assign actions (i.e., 'allow' or 'deny') to the firewall rules. Placements of hosts in the core network are synthesized depending on the topology. We define the core network topology as a collection

TABLE V
AN EXAMPLE OF REACHABILITY REQUIREMENT MATRIX

		Host		
		h0	h1	h2
Host	h0	-	yes	no
	h1	no	-	no
	h2	no	yes	-

TABLE VI
AN EXAMPLE OF RISK MITIGATION MATRIX

		Impact		
		high	medium	low
Threat	high	no	no	yes
	medium	no	yes	yes
	low	yes	yes	yes

of routers and firewalls, and links between them. It is assumed that firewalls are placed in the topology in such a way that helps to form a defense-in-depth. Synthesizing the placement of a host means selecting a router (i.e., associated subnet) to which the host can be connected, so that the constraints are satisfied. Note that an administrator can easily apply the synthesized placements of hosts with the help of virtual LANs.

B. User-driven Constraints

It is obvious that if there is no connectivity between the hosts, the network would be risk-free. However, a computer network is for connectivity, although connectivity between each pair of hosts is not essential. Therefore, the *reachability requirement* (or *connectivity requirement*) is very crucial as a constraint. We introduce a matrix named *Reachability Requirement Matrix* (RRM), which represents all reachability requirements (i.e., allowed traffics) in the organization. An example of such a matrix is shown in Table V taking 3 hosts (`h0`, `h1`, and `h2`) into consideration. In RRM, the hosts in rows are the traffic sources, while the hosts in columns are the traffic destinations. A traffic indicated by 'yes' must be able to successfully reach its destination from its source. A 'no' indication for a traffic means that the reachability of the traffic is not required.

The synthesis process is constrained by the limit that is set on the risk. The limit stands for the maximum overall risk score of the network (refer to Section II-E). We can have specific constraints on the potential risk of a host. For example, a particular host can not have high risk, or the host should have low risk. Moreover, there can be security hardening policy based on the source and destination of a traffic. Since a host having high threat has higher chance to be compromised and a host having high impact has more loss due to an attack, we consider the threat of the communicating host (the source) and the impact of the destination host for risk mitigation. We define the *Risk Mitigation Matrix* (RMM) that represents a relational matrix between threat and impact showing whether a host having a particular threat is allowed to communicate with a host having a particular impact. A host, for example, with high threat should not be allowed to communicate with a high impact host. Each connectivity requirement must be satisfied without violating the risk mitigation constraints. An example of RMM is shown in Table VI. We take the entries with the 'yes' value as the constraints.

TABLE VII
FORMALIZATIONS OF HOST PLACEMENTS AND FIREWALL POLICIES

Formalization of the host network addresses

```
; A host can have any of the set of IP addresses.
(assert (forall ((x Int))
   (⇒ (isHost x)
      (or (= (nAddr x) (getIP 152 15 0 1))
          (= (nAddr x) (getIP 152 15 0 2))
          . . . . . .

; No two hosts cannot have the same IP address.
(assert (forall ((x Int) (y Int))
   (⇒ (and (isHost x) (isHost y))
      (and (not (= (nAddr x) (nAddr y)))))))
```

Formalization of firewall policies

```
; Default firewall rules
; Alow rules from a higher secure zone to a lower secure zone
(assert (= (frSrc f0 0) (getIP 150 15 0 0)))
(assert (= (frSMask f0 0) ((_ int2bv 32) 24)))
(assert (= (frDest f0 0) (getIP 150 15 1 0)))
(assert (= (frDMask f0 0) ((_ int2bv 32) 24)))
(assert (frAct f0 0))
. . . . . .

; Synthesized firewall rules
; Rules from a lower secure zone to a higher secure zone)
(assert (= (frSrc f0 0) (getIP 150 15 1 1)))
(assert (= (frSMask f0 0) ((_ int2bv 32) 32)))
(assert (= (frDest f0 0) (getIP 150 15 0 1)))
(assert (= (frDMask f0 0) ((_ int2bv 32) 32)))
; Firewall action (frAct) is for synthesis
. . . . . .
```

C. Synthesis Model

We formalize the synthesis model using SMT [15]. The risk analysis model follows the declarative logic model described in Section II. We apply *uninterpreted function* over *integer* terms to formalize the risk analysis model and corresponding network reachability model. In this section, we describe the formalizations of some parts of the model, which are crucial for understanding the synthesis of the targeted configurations. The formalizations follow the *SMTLIB V-2* syntax [17].

1) Host's Address Synthesis Model: We synthesize the addresses of the hosts according to the given constraints. There are two invariant constraints for assigning a network address to a host: (i) a host can take an address only from a set of addresses, and (ii) no two hosts can have the same address. The set of addresses corresponds to the subnets of the network. Each subnet is connected to a particular router. A router can have more than one subnet connected to it. In Table VII, the formalizations of these constraints are shown, which are followed from a small example. Fig. 4 shows the network of the example. In this example, we consider only seven hosts along with the Internet. The core network consists of three routers and two firewalls. The placements of the hosts in the network are to be synthesized according to the constraints.

2) Firewall Policy Synthesis Model: Our firewall policy synthesis problem is the assignments of actions to the firewall rules. In the firewall policy, we consider one rule for each possible traffic in the network. The action of the rule is kept undefined. As all traffics are considered individually in the policy, there is no rule which is superset or subset of another

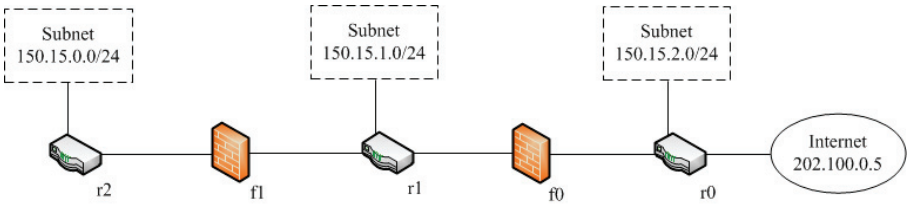


Fig. 4. A core network with three routers and two firewalls.

TABLE VIII
FORMALIZATION OF LINKS

```
; The links (which are bidirectional) of the core network are fixed
(assert (and (link r0 f0) (link f0 r0)
       (link f0 r1)
. . . . . .

; Modeling of the relation/link between a host and a router

; A router (e.g., r0) is connected to each host of a subnet
(assert (= (rHostSubnet r0 0) (getIP 150 15 0 0)))
(assert (= (rHostSMask r0 0) ((_ int2bv 32) 24)))
. . . . . .

; inSubnet finds whether the host x is under the s'th subnet (the router r)
(declare-fun inSubnet ((x Int) (r Int) (s Int)) Bool
. . . . . .

; isLink finds if there is a link between two nodes
(assert (forall ((x Int) (y Int))
    (⇒ (isLink x y)
       (or (and (isRouter x) (isRouter y) (link x y))
          (and (isRouter x) (isHost y) (= x r0) (inSubnet x r0 0)
. . . . . .
```

TABLE IX
SYNTHESIS MODELING: FORMALIZATION OF CONSTRAINTS

```
; Reachability requirement constraints
(assert (and (reachable internet h2)
    (reachable internet h6)
    (reachable h2 h0) (reachable h0 h1)
    (reachable h1 h0) (reachable h1 h4)
    (reachable h4 h3) (reachable h5 h6)
    (reachable h5 internet)
    (reachable h1 internet)))

; Risk mitigation constraints
(assert (forall ((x Int) (y Int))
    (⇒ (reachable x y)
       (or (= (threat x) 0)
          (= (impact y) 0)
          (and (= (threat x) 1) (= (impact y) 1))))))

; Constraint for risk
(assert (= totalRisk (+ (risk h0) (risk h1) (risk h2) · · · (risk h6))))
(assert (< totalRisk 6)) ; Max possible risk is 14 (all hosts with high risk)
```

rule. However, the optimization of the synthesized firewall rules can be done later for better efficiency. We consider an invariant constraint for each firewall that any traffic other than the rules in the policy should be denied. This is the default deny constraint. The partial model of the firewall policy is shown in Table VII.

3) Modeling of Links and Routing Rules: As the core network is fixed, the links between the core network components (i.e., routers and firewalls) are defined. The placements of hosts in the network is undefined, i.e., unset, and as a result the links between a host and a router can not be asserted (i.e., fixed). Most importantly, the routers cannot have a fixed routing policy based on these undefined placements. We model this issue considering fixed subnets associated to each router. Since the core network is fixed, the subnets connected to a router are also fixed. We assume that if a host belongs to a subnet of a router, there should be a link between the host and the router. As the subnets are known, the routing policy is now easy to define. The formalization of the links are presented in Table VIII. The function *link* is used to denote a link. The links of the core network are asserted explicitly. The links from the hosts to the routers are modeled as a rule, which states that if a host belongs to a subnet of a router, there is a link between the host and the router. The function *inSubnet* checks whether a host's address is subsumed by a router's subnet.

4) Modeling of Constraints: The modeling of the constraints is shown in Table IX. The reachability requirement constraints are modeled using the function *reachable*. We compute the overall risk (*totalRisk*) of the network as the summation of the risks of the hosts. Then, we set a constraint that the total risk should be less than or equal to a (given) threshold value. In the process of risk computation, we need to calculate the exposure (refer to Section II). In our synthesis model, we simplify the definition of exposure by considering the *exposure depth* only. The constraint based on the risk mitigation policy (refer to the *risk mitigation matrix*) is formalized as: if the reachability is true for a pair of source and destination, the risk mitigation policy should allow the traffic.

TABLE X
FIREWALL POLICIES

Firewall	Source Address	Destination Address	Action
f1	150.15.1.2	150.15.0.3	Deny
f1	150.15.1.3	150.15.0.3	Deny
f1	150.15.2.1	150.15.0.3	Deny
f1	. . . . .	. . . . .	Allow
f0	150.15.2.2	150.15.0.3	Deny
f0	Internet	150.15.0.2	Deny
f0	Internet	150.15.0.3	Deny
f0	. . . . .	. . . . .	Allow

5) Example Output: If we verify the satisfiability of the formalization of our example using a SMT solver (i.e., Z3), we the solver can returns SAT (satisfiability) or UNSAT (unsatisfiability). If the result is SAT, a model is also given as output. This model represents the assignments of variables that satisfy the given constraints. An UNSAT result specifies that one or more constraints is not possible to satisfy. Hence, there is no possible model in this case. The synthesized firewall rules (partial) in the case of this example is shown in Table X.

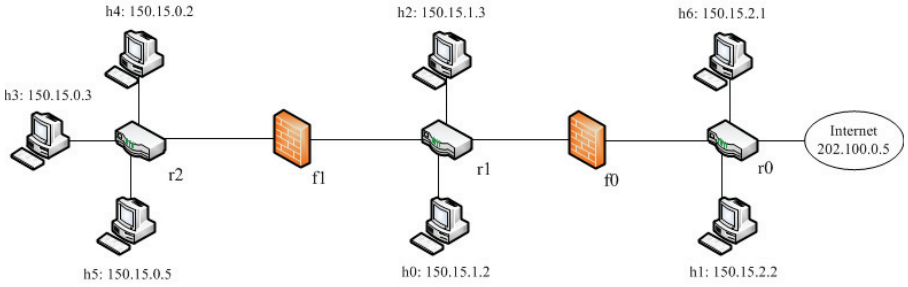
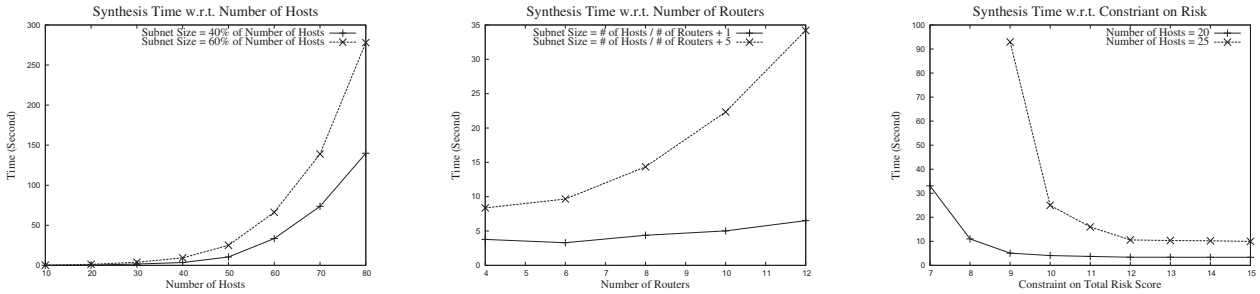


Fig. 5. The placements of the hosts in the network.



(a) (b) (c)

Fig. 6. Synthesis time with respect to (a) the number of hosts, (b) the number of routers (core network size) and (c) the constraint on the network risk (i.e., total risks of the hosts).

The placements of the hosts in the network according to the output model is shown in Fig. 5. We find that the hosts $h3$, $h4$, and $h5$ are connected to the router $r2$, the hosts $h0$ and $h2$ are connected to the router $r1$, and $h1$, $h5$, and the Internet are connected to the router $r0$.

D. Evaluation

The goal of our evaluation is to verify the scalability of our model. We evaluate the execution time of our synthesis model by varying the number of hosts, the number of routers (i.e., the levels of defense-in-depth), and the constraint on the overall network risk.

Impact of the number of hosts. The core network that we consider in this analysis includes three routers and two firewalls (as shown in Fig. 4). We do the analysis in two different scenarios: in one scenario each subnet size (i.e., the maximum possible number of hosts residing in a single subnet) is 40% of the total number of hosts, while in the second scenario the size is 60%. The evaluation results are shown in Fig. 6(a). We observe that the synthesis time increases quadratically with respect to the number of hosts. The number of possible flows between the hosts depends on their number (considering a fixed number of services) and it is $O(N^2)$, where N is the number of hosts. The subnet size has a significant impact on the time. When the number of potential addresses in a subnet increases, the number of potential choices for the address of a host increases. As a result, the synthesis time increases. Fig. 6(a) justifies this argument.

Impact of the number of routers. The core network that we consider in this analysis varies by increasing the number of routers. We consider a firewall between each two routers (as shown in Fig. 4). We do the analysis in two different scenarios. In one scenario each subnet size (i.e., the address space for the hosts) is the ratio of the total number of hosts and the total number of routers, plus 1. In the second scenario, each subnet size is the same ratio plus 5. The evaluation results are shown in Fig. 6(b). We observe that the synthesis time increases with respect to the number of routers. When the number of routers increases, the number of firewall as well as the total number of firewall rules increases. As a result, the number of potential choices for the placement (subnet and the address within the address) of a host increases. Hence, the synthesis time increases.

Impact of the constraint on the network risk. In this analysis we vary the constraint on the overall network risk (i.e., the accumulation of the risk scores of all the hosts in the network). The core network that we consider in this analysis includes three routers and two firewalls. We do the analysis in two different scenarios: in one scenario the number of hosts is 10, while in the second scenario the number is 20. In the experiments, we vary the risk constraint from 7 to 15. Here, we consider a quantitative value for each risk type, i.e., 1, 0.5, 0 for high, medium, and low risk, respectively (refer to Section II-E). Hence, if a network has 20 hosts, among which 4 hosts have high risk, 8 hosts have medium risk, and the rests have low risk, the total risk value is $(4\times1+8\times0.5+8\times0=8)$. Hence, if the risk constraint is 10, it specifies that the total risk of the hosts cannot be more than 10.

The evaluation results are shown in Fig. 6(c). We observe that the synthesis time decreases with the increase of the constraint value (i.e., when the constraint is relaxed). We observe sharp decrease if we increase the constraint. However, this decreasing rate of the synthesis time reduces rapidly if the constraint is relaxed more. This is due to the reason that

the number of possible satisfiable solutions in tight constraint (lower value for the maximum acceptable network risk) are few. As a result, the required time is high. The more the constraint is relaxed, the more solutions are possible. Hence, the synthesis time (the searching time of a satisfiable solution) reduces. If the constraint is relaxed more and more, it does not increase possible solutions significantly due to the constraints like the number of subnets, the reachability requirement, etc.

IV. Related Work

Jaquith in [4] discusses the technical security metrics and gives an emphasis on keeping a history for the measurements in order to track the progress of any failure. The author also explains the problem of "lack of proper visualization". Minimum-cost network hardening is one of the first efforts done in the area of quantitative network security analysis [5]. This work has limitations of taking all the given network resources equally important and the attack resistance as impossible or trivial. In the work [7], authors present a methodology to model the composition of vulnerabilities as attack graphs obtainable from *topological vulnerability analysis* (TVA) system. By analyzing attack graph, they explore different concepts and issues on a metric to quantify potential attacks. Singhal and Xou describe the security metrics to compute the overall risk in an enterprise system in [8]. They present an attack graph based system architecture and security metrics for an enterprise network in order to quantify the overall risk, which are essential for the decision makers in taking sensible security management. In our research work, we consider the reachability as the attack paths. Our risk analysis expression is quite simple and generic. The authors present a security analysis tool for analyzing misconfiguration problems in existing policies [11]. But the tool cannot compute potential risk of a host. It is also incapable to synthesize policies.

The research on security configuration synthesis is in a premature stage. Dewri et al. present a systematic approach to solve the optimization problem of security hardening in limited budget using evolutionary algorithm [12]. Their model is developed on an attack tree model. Homer and Ou propose a logical framework using attack graphs to find optimal deployment of security devices to block all attack scenarios [13]. There are few works on risk based security configuration analysis. In [10], the authors propose a framework for automatic creation of network security architecture including configuration rules and device placements in order to minimize risk while satisfying the business requirements. The framework intends to automate the creation of external and internal Demilitarized Zones (DMZ) to improve security by increasing isolation.They formalize this as an optimization problem. However, the formalization takes a very simple risk definition, which does not consider exposure properly. Moreover, the solution to the problem is done by taking heuristic approximations. In our work, we see the problem in a different and novel dimension, i.e., as a constraint satisfaction problem. Here, we give emphasis on a proper risk analysis and an automated solution for the risk-based security hardening.

V. Conclusion

In this paper, we present a formal approach for the network risk analysis. We use declarative logics to model the risk of a host. The proposed risk analysis approach is a novel and significant addition to the arena of risk assessment. We apply different matrices in the process of the host's risk computation. These matrices make the risk definition flexible and extensible, which is very useful in the dynamic concept of security. In the evaluation of our declarative risk analysis technique, we observed that the technique takes less than 10 seconds for 1000 hosts. We also show the application of risk-based constraints in the synthesis of firewall policies as well as host placements for securing the network. We model the synthesis task as a constraint satisfaction problem and apply SMT to formalize these constraints. Though our synthesis model does not consider constraints like budget, it is the first step to implement an automatic synthesizer of network configurations and firewall policies for the risk-based security hardening. The evaluation results show that the model can solve a synthesis problem with 50 hosts in around 10 seconds. In future, we will extend our model to solve the problem of synthesizing security policies for other security devices like IPSec and IDS. We will also do more study in order to improve the efficiency of our synthesis technique.

References

[1] G. Stoneburner, A. Goguen and A. Feringa, *Risk Management Guide for Information Technology Systems*, NIST Publication 800-30, July, 2002.

[2] Steve Elky, *An Introduction to Information System Risk Management*, SANS Institute InfoSec Reading Room, May, 2006.

[3] M. A. Rahman and Ehab Al-Shaer, *A Declarative Approach for Modeling and Verification of Network Access Control Policies*, In 12th IM, 2011.

[4] Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*, Addison Wesley, 2007.

[5] S. Noel et al., *Efficient Minimum-cost Network Hardening via Exploit Dependency Graphs*, The 19th Annual Computer Security Applications Conference, 2003.

[6] Xinming Ou, S. Govindavajhala, and A. W. Appel, *MulVAL: A logic-based network security analyzer*, USENIX Security Symposium, 2005.

[7] Lingyu Wang, Anoop Singhal and Sushil Jajodia, *Measuring the Overall Security of Network Configurations Using Attack Graphs*, In the 21st IFIP WG 11.3 Working Conf. on Data and Applications Security, 2007.

[8] A. Singhal and S. Xou, *Techniques for Enterprise Network Security Metrics*, In Cyber Security and Information Intelligence Research Workshop, Oakridge National Labs, Oarkridge, April, 2009.

[9] *CVSS: Common Vulnerability Scoring System*, Available in http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2.

[10] B. Zhang and E. Al-Shaer, *On Synthesizing Distributed Filtering Configuration Considering Risk, Usability and Cost Constraints*, CNSM, France , 2011.

[11] E. Al-shaer et al., *Network Configuration in A Box: Towards End-to-End Verification of Network Reachability and Security*. ICNP, 2009.

[12] R. Dewri et al., *Optimal security hardening using multi-objective optimization on attack tree models of networks*, In 14th ACM CCS 2007.

[13] 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.

[14] *SWI-Prolog*, Available in http://www.swi-prolog.org/.

[15] Leonardo De Moura and Nikolaj Bjrner, *Z3: An Efficient SMT Solver*, TACAS, Budapest, Hungary, 2008

[16] L. de Moura and N. Bjørner, *Satisfiability Modulo Theories: An Appetizer*, In Brazilian Symposium on Formal Methods, 2009.

[17] Clark Barrett, Aaron Stump, and Cesare Tinelli, "The SMT-LIB Standard: Version 2.0", In the 8th International Workshop on Satisfiability Modulo Theories, Scotland, UK, July 2010.