J Netw Syst Manage (2011) 19:343–366
DOI 10.1007/s10922-010-9177-6

Objective Risk Evaluation for Automated Security Management

Mohammad Salim Ahmed · Ehab Al-Shaer · Mohamed Taibah · Latifur Khan

Received: 12 February 2009 / Accepted: 16 September 2010 / Published online: 30 October 2010

Abstract Network security depends on a number of factors. And a common characteristic of these factors is that they are dynamic in nature. Such factors include new vulnerabilities and threats, the network policy structure and traffic. These factors can be divided into two broad categories. Network risk and service risk. As the name implies, the former one corresponds to risk associated with the network policy whereas the later one depends on the services and software running on the system. Therefore, evaluating security from both the service and policy perspective can allow the management system to make decisions regarding how a system should be changed to enhance security as par the management objective. Such decision making includes choosing between alternative security architectures, designing security countermeasures, and to systematically modify security configurations to improve security. As there may be real time changes to the network threat, this evaluation must be done dynamically to handle such changes. In this paper, we provide a security metric framework that quantifies objectively the most significant security risk factors, which include existing vulnerabilities, historical trend of vulnerabilities of the remotely accessible services, prediction of potential vulnerabilities for these services and their estimated severity, unused address space

M. S. Ahmed (✉) · L. Khan
University of Texas at Dallas, Richardson, TX 75080, USA
e-mail: salimahmed@utdallas.edu

L. Khan
e-mail: lkhan@utdallas.edu

E. Al-Shaer
University of North Carolina Charlotte, Charlotte, NC 28223, USA
e-mail: ealshaer@uncc.edu

M. Taibah
DePaul University, Chicago, IL 60604, USA
e-mail: mtaibah@cs.depaul.edu

and finally propagation of an attack within the network. These factors cover both the service aspect and the network aspect of risk toward a system. We have implemented this framework as a user-friendly tool called *Risk based prOactive seCurity cOnfiguration maNAger (ROCONA)* and showed how this tool simplifies security configuration management of services and policies in a system using risk measurement and mitigation. We also combine all the components into one single metric and present validation experiments using real-life vulnerability data from *National Vulnerability Database (NVD)* and show comparison with two existing risk measurement tools.

Keywords Security evaluation · Risk prediction · Vulnerability measure · Attack propagation · Attack immunity · Quality of protection

1 Introduction

In order to formulate a security management framework, it is imperative that we have a security measurement scheme with which we can compare different network security policies. And, to measure security, we have to first decide on how we should consider a computer network. A *computer network system* can be visualized from both physical and security perspectives. From a physical perspective, such a *system* can be considered as consisting of a number of computers, servers and other system components interconnected via high speed LAN or WAN. However, when the same *system* is visualized from a security perspective, it can be divided into its *service* and *network* part. In this paper, we have formulated our security measurement and management framework using this security perspective.

The network part of a *computer network system* allows data to come into the system, some of which may be generated with the intent of an attack and are malicious in nature and some, on the other hand, are benign for the system. After such malicious data makes its way into the system, its impact depends on which services or software in the system are affected by it. Therefore, following this notion, a *system* can be considered as a combination of multiple services that interact with each other and also with other network systems using its network communication.

Since, we can model a system in such a way, risk evaluation of individual services can help in identifying services that pose higher risk. This in turn, allows the system administrators to pay extra attention to such services. We can then incorporate this information in our *network security policy* or *access options* to enhance security. In this paper, *network security policy* is defined as a set of objectives, rules of behavior for users and administrators, list of active services and corresponding software and finally, requirements for system configuration and management. Therefore, it also specifies which services are part of the system. But, meaningful changes to security policy can only happen if we have a security measure of all the services. If the risk of a single service can be quantified, then existing aggregating methods can be used to evaluate the security of all the services in a system. Along with the service risk measurement, integrating it with the network/policy risk measurement can provide a complete risk analysis of the

system. This can allow the management to employ resources for risk mitigation based on the severity and importance of each of these two aspects of risk.

The effectiveness of a security metric depends on the security measurement techniques and tools that enable network administrators to analyze and evaluate network security. Our proposed new framework and user-friendly implementation for network security evaluation can quantitatively measure the security of a network based on these two critical risk aspects - the risk of having a successful attack due to the services and the risk of this attack being propagated within the network due to the policy. Finally all these components are integrated into a single metric called *Quality of Protection Metric (QoPM)* to portray the overall security level of the network for the decision makers.

We have a number of contributions in this paper. First, our proposed tool, based on our framework, can help in comparing services and security policies with each other to determine which ones are more secure. It is also possible to judge the effect of a change to the policy by comparing the security metrics before and after the change. And, this comparison can be done, in aggregated form (e.g., weighted average) or component wise, based on user needs. Second, our framework and its implementation is also an important step toward *adaptive security systems*, in which, networks can be evaluated and automatically hardened accordingly by constantly monitoring changes in the network and service vulnerabilities. We used the Java Programming Language to implement the proposed metrics in one graphical user-interface called *ROCONA*. This tool simplifies periodic risk measurement and management. Third, the underlying framework of our tool is more advanced with respect to the existing tools and previous research works. Previous research include those that scan or analyze a given network to find out allowed traffic patterns and vulnerabilities in the services. However, most of them do not predict the future state of the security of the network, consider the policy immunity or spurious risk. Research that do try to predict future vulnerabilities like [1] is limited to predicting software bugs and it requires inside knowledge of the studied software. In this respect, our prediction model is general and can work using only publicly available data. Finally, using the *National Vulnerability Database* published by *NIST* [2], we performed extensive evaluation of our proposed model. The results show high level of accuracy in the evaluation and a new direction towards security measurement.

The organization of the paper is as follows. First, a detailed description of the entire framework is given in Sect. 2 followed by our implementation of the framework, deployment and a case study in Sect. 3. Then, we present our strategy of combining all the scores in Sect. 4, followed by our experiments, results and discussion on the experimental results in Sect. 5. Next, we discuss the related works in Sect. 6. And finally, we present our conclusion and some directions of future research in Sect. 7.

2 Security Risk Evaluation Framework

As can be seen from Fig. 1, we have modeled our framework as a combination of two parts. The first part measures the security level of the services within the computer network system based on vulnerability analysis. This analysis considers the presence

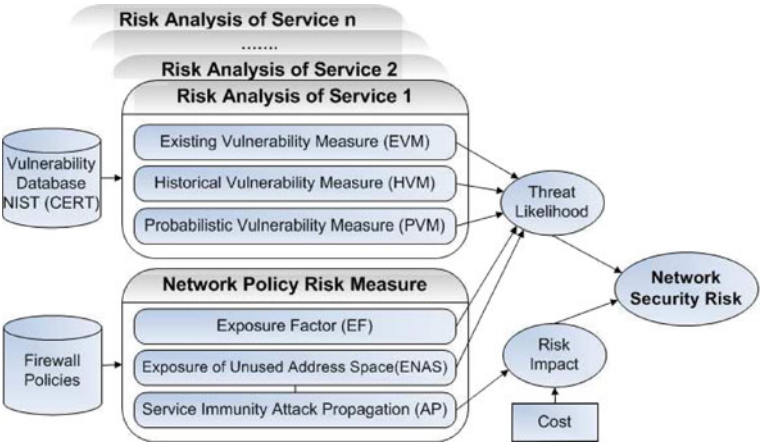

Fig. 1 Network risk measurement framework

of existing vulnerabilities, the dormant risk based on previous vulnerabilities and the potential future risk. In the second part, the service exposure to outside network by filtering policies, the degree of penetration or impact of successful attacks against the network and the risk due to traffic destined for unused address space are measured from a network security policy perspective. The three service risk components, the exposure factor and the unused address space exposure, together give us the threat likelihood and at the same time the attack propagation provides us with the risk impact to the network of interest. When the risk impact is combined with the cost of the damage, we get the contribution of network policies in the total risk. In this paper, we use the terms vulnerability and risk interchangeably. If a service has vulnerabilities, we consider it as posing a risk towards the corresponding system. Also, the terms security and risk are regarded as opposites. Therefore, when we consider security, high values indicate high level of security and low values indicate low security. On the other hand, when we use the term vulnerability, high values indicate high vulnerability or risk (i.e., low security) and vice versa. Also, we shall refer to a *computer network system* (with both its service and network part) simply as a *system* from this point forward.

2.1 Service Risk Measurement

In this section, we describe and discuss in detail the calculation method of our vulnerability analysis for service risk measurement. This analysis comprises of *Historical Vulnerability Measure*, *Probabilistic Vulnerability Measure* and *Existing Vulnerability Measure*.

2.1.1 Historical Vulnerability Measure

Using the vulnerability history of a service, the *Historical Vulnerability Measure (HVM)* quantifies how vulnerability prone a given service has been in the past. Considering both the frequency and age of the vulnerabilities, we combine the

severity scores of past vulnerabilities so that a service with a high frequency of vulnerabilities in the near past has a high *HVM*.

Let the set of services in the system be S. We divide the set of vulnerabilities $HV(s_i)$ of each service $s_i \in S$ into three groups—$HV_H(s_i)$, $HV_M(s_i)$ and $HV_L(s_i)$ for vulnerabilities that pose high, medium and low risks to the system. In evaluating a service, the vulnerabilities discovered a long time ago should carry smaller weight, because with time these would be analyzed, understood and patched. And our analysis of vulnerabilities found the service vulnerability to be less dependent on past vulnerabilities. So, we regard the relationship between a vulnerability and its age as nonlinear and we apply an exponential decay function of the age of the vulnerability. The age of a vulnerability indicates how much time has passed since its discovery and is measured in number of days. The parameter β in the decay function controls how fast the factor decays with age. In computing the *HVM* of individual services, we sum up the decayed scores in each class, and take their weighted sum. Finally, we take its natural logarithm to bring it to a more manageable magnitude. Before taking the logarithm, we add 1 to the sum so that a sum of 0 will not result in ∞, and the result is always positive. The *HVM* of service s_i, $HVM(s_i)$ is calculated as follows:

$$HVM(s_i) = \ln\left(1 + \sum_{X\in\{H,M,L\}} w_X . \sum_{v_j \in HV_X(s_i)} SS(v_j).e^{-\beta Age(v_j)}\right) \quad (1)$$

In this equation, v_j is a vulnerability of service s_i, $Age(v_j)$ can have a range of (0, ∞) and $SS(v_j)$ is its severity. Finally, the *Aggregate Historical Vulnerability Measure* of the whole system, *AHVM(S)*, is calculated as

$$AHVM(S) = \ln\left(\sum_{s_i \in S} e^{HVM(s_i)}\right) \quad (2)$$

This equation is designed to be dominated by the highest *HVM* of the services exposed by the policy. We take the exponential average of all the *HVM*s so that the *AHVM* score will be at least equal to the highest *HVM*, and will increase with the *HVM*'s of the other services. If arithmetic average was taken, then the risk of the most vulnerable services would have been undermined. Our formalization using the exponential average is validated through our conducted experiments.

2.1.2 Probabilistic Vulnerability Measure

The *Probabilistic Vulnerability Measure (PVM)* combines the probability of a vulnerability being discovered in the next period of time and the expected severity of that vulnerability. This measure, therefore, gives an indication of the risk faced by the system in the future.

Using the vulnerability history of a service, we can calculate the probability of at least one new vulnerability being published in a given period of time. From the vulnerability history, we can also compute the *Expected Severity* of the predicted vulnerabilities.

We calculate *Expected Risk (ER)* for a service as the product of the probability of at least one new vulnerability affecting the service in the next period of time and the expected severity of the vulnerabilities. We can compare two services using this measure—a higher value of the measure will indicate a higher chance of that service being vulnerable in the near future.

For each service, we construct a list of the interarrival times between each pair of consecutive vulnerabilities published for that service. Then we compute the probability of the interarrival time being less than or equal to a given period of time, T. Let P_{s_i} be the probability that d_{s_i}, the number of days before the next vulnerability of the service s_i is exposed, is less than or equal to a given time interval, T, i.e.,

$$P_{s_i} = P(d_{s_i} \leq T) \tag{3}$$

To compute the expected severity, first we build the probability distribution of the severities of the vulnerabilities affecting the service in the past. Let X be the random variable corresponding to the severities, and x_1, x_2, ... be the values taken by X. Then, the expectation of X_{s_i} for service s_i is given by (4):

$$E[X_{s_i}] = \sum_{i=1, x_j \in s_i}^{\infty} x_j P(x_j) \tag{4}$$

where $P(x_j)$ is the probability that a vulnerability with severity x_j occurs for service s_i. Finally, we can define the *Expected Risk (ER)*, of a service s_i as follows.

$$ER(s_i) = P_{s_i} \times E[X_{s_i}] \tag{5}$$

If S is the set of services exposed by the policy, we can combine the probabilities of all the services exposed by the policy to compute the *PVM* of the system as in (6).

$$PVM(S) = \ln \sum_{s_i \in S} e^{ER(s_i)} \tag{6}$$

For combining the expected risks into one single measure, we are using the exponential average method like the *AHVM*, so that the *PVM* of a system is at least as high as the highest expected risk of a service in the system. Therefore, from definition perspective, *PVM* is similar to *AHVM* as both measures correspond to a collection of services whereas *HVM* and *ER* are similar as they correspond to a single service.

In order to calculate *PVM*, we have looked into different methods to model the interarrival times. The first method that we have analyzed is *Exponential Distribution*. Interarrival times usually follow exponential distribution [3]. In order to evaluate (3) for a given service, we can fit the interarrival times to an exponential distribution, and we can then find the required probability from the *Cumulative Distribution Function (CDF)*. If λ is the mean interarrival time of service s_i, then the interarrival times of service s_i, d_{s_i}, will be distributed exponentially with the following CDF:

$$P_{s_i} = P(d_{s_i} \leq T) = F_{d_{s_i}}(T) = 1 - e^{-\lambda T} \tag{7}$$

The next method that have been analyzed is *Empirical Distribution*. We can model the distribution of the interarrival times of the data using empirical distribution. The frequency distribution is used to construct a *CDF* for this purpose.

In this case, the empirical *CDF* of the interarrival time, d_{s_i}, will be:

$$P_{s_i} = P(d_{s_i} \leq T) = F_{d_{s_i}}(T) = \frac{\sum_{x \leq T} f_i(x)}{\sum f_i(x)} \tag{8}$$

Finally, exponential smoothing, a *Time Series Analysis* technique was also used for identifying the underlying trend of the data and finding the probabilities.

2.1.3 Existing Vulnerability Measure

Existing vulnerability is important as sometimes the services in the system are left unpatched or, in cases where there are no known patches available. Also, when a vulnerability is discovered, it takes time before a patch is introduced for it. During that time, the network and services are vulnerable to outside attack. The *Existing Vulnerability Measure (EVM)* measures this risk. *EVM* has been studied and formalized in our previous work [4].

We again use the *exponential average* to quantify the worst case scenario so that the score is always at least as great as the maximum vulnerability value in the data. Let $EV(S)$ be the set of vulnerabilities that currently exist in the system with service set S, and $SS(v_j)$ be the severity score (i.e., *CVSS*) of a vulnerability v_j. We divide $EV(S)$ into two sets—$EV_P(S)$ containing the vulnerabilities with existing solutions or patches, and $EV_U(S)$ containing those vulnerabilities that do not have existing solutions or patches. Mathematically,

$$EVM(S) = \alpha_1 . \ln \sum_{v_j^p \in EV_P(S)} e^{SS(v_j^p)} + \alpha_2 . \ln \sum_{v_j^u \in EV_U(S)} e^{SS(v_j^u)} \tag{9}$$

Here, the weight factors α_1 and α_2 are used to model the difference in security risks posed by those two classes of vulnerabilities. Details can be found in [4]. However, finding the risk to fully patched services based on historical trend and future prediction is one of the major challenges where we contribute in this paper.

2.1.4 HVM, PVM and EVM Based Risk Mitigation

There will always be some risks present in the system. But the vulnerability measures can allow the system administrator to take steps in mitigating system risk. As mentioned previously, *EVM* indicates whether there are existing vulnerabilities present in the system of interest. The vulnerabilities that may be present in the system can be divided into two categories. One category has known solutions i.e., patches available and the other category does not have any solutions. In order to minimize risk present in the system, *ROCONA* (1) Finds out which vulnerabilities have solutions and alerts the user with the the list of patches available for install, (2) Services having higher vulnerability measure than user defined threshold are listed to the user with the options: (i) block the service completely (ii) limit the traffic

toward the service by inserting firewall rules (iii) minimize traffic by inserting new rules in IDS (iv) place the service in DMZ area and (v) Manual. (3) Recalculate *EVM* to show score for updated settings of the system. The administrator can set a low value as threshold to always keep track of the services that contribute most in making the system vulnerable.

As can be seen from our definition of *HVM*, it gives us the historical profile of a software and *PVM* score gives us the latent risk toward the system. Naturally we would like to use a software that does not have a long history of vulnerabilities. In such cases, a software with low *HVM* and *PVM* score providing the same service should be preferred because low scores for them indicates low risk as mentioned previously. Our implemented tool performs the following steps to mitigate the *HVM* and *PVM* risk. (1) Calculate the *HVM* and *PVM* scores of the services (2) Compare the scores to the user defined threshold values. (3) If scores are below the threshold then strengthen layered protection with options just like in case of *EVM*. (4) Recalculate the *HVM* and *PVM* scores of the services (5) If scores still below the threshold, then show recommendations to the system administrator. Recommendations include (1) Isolation of the service using Virtual LANs (2) Increase weight (i.e., importance) to the alerts originating from these services even if false alarms increase. (3) Replace the service providing software with a different one.

The administrators can use our tool to measure the *EVM*, *HVM* and *PVM* scores of similar service providing softwares from the *NVD* database and choose the best possible solution. But for all of them, a cost-benefit analysis must precede such a decision making. It should be clarified here that these three measures described here just provide us the risk of the system from a service perspective. If the network policy is so strong that no malicious data (i.e., attacks) can penetrate into the system, then these scores may not pose high importance. But our assumption here is that a system will be connected to outside systems through the network and there will always be scope of malicious data to gain access within the system. That is why we believe that these vulnerability scores will be useful for decision making.

2.2 Network Risk Measurement

The network policies determine the exposure of the network to outside world as well as the extensiveness of an attack on the network (i.e., how widespread the attack is). In order to quantify network risk, we have focused on three factors. These three factors are *Attack Propagation (AP)*, *Exposure of Unused Address Spaces (ENAS)* and *Exposure Factor (EF)*.

2.2.1 Attack Propagation Metric

The degree to which a policy allows an attack to spread within the system is given by the *Attack Propagation (AP)* metric. The *Attack Propagation* measure assesses how difficult it is for an attacker to propagate an attack through a network across the system, using service vulnerabilities as well as security policy vulnerabilities.

Attack Immunity: For the purpose of these measures, we define a general system having N hosts and protected by k firewalls. Since we analyze this measure using a

graph, we consider each host as a node in a graph where the graph indicates the whole system. The edges between nodes indicate network connectivity. Let each node be denoted as d and D be the set of nodes that are running at least one service. We also denote the set of services in d as S_d and p_{s_i} as the vulnerability score for service s_i in S_d. In this case, however, p_{s_i} has the range (0, 1).

For our analysis, we define another measure, I_{s_i}, that assesses the *Attack Immunity* of a given service, s_i, to vulnerabilities based on that service's *EVM*, *HVM* and *PVM*. I_{s_i} is directly calculated from p_{s_i}, the combined vulnerability measure of service s_i, as:

$$I_{s_i} = -\ln(p_{s_i}) \tag{10}$$

I_{s_i} has a range of $[0, \infty)$, and will be used to measure the ease with which an attacker can propagate an attack from one host to the other using service s_i. Thus, if host m can connect to host n using service s_i exclusively, then I_{s_i} measures the immunity of host n to an attack initiating from host m. In case of a connection with multiple services, we define a measure of combined attack immunity. Before we give the definition of the combined attack immunity, we need to provide one more definition. We define S_{mn} as the set of services that host m can use to connect to host n.

$$S_{mn} = \{s_i | \textit{host } m \textit{ can connect to } n \textit{ using service } s_i\} \tag{11}$$

Now, assuming that the combined vulnerability measure is independent for different services, we can calculate a combined vulnerability measure $p_{s_{mn}}$. Finally, using this measure, we can define the combined attack immunity $I_{s_{mn}}$ as:

$$I_{s_{mn}} = -\ln(p_{s_{mn}}) \times PL \tag{12}$$

Here *PL* is the protection level. For protection using firewall, the protection is 1. For protection using *IDS*, the protection is a value between 0 and 1 and is equal to the false negative rate of the *IDS*. It is assumed that firewall will provide the highest immunity and hence *PL* is assigned a value of 1. In case of *IDS* the immunity decreases by its level of misses to attacks. This ensures that our *Attack Immunity* measure considers the absence of firewalls and *IDS*. From the definition of *AP*, it is obvious that high immunity indicates low risk to the system.

Service Connectivity Graph: To extract a measure of security for a given network, we map the network of interest to a *Service Connectivity Graph (SCG)*. The *SCG* is a directed graph that represents each host in the network by a vertex, and represents each member of S_{mn} by a directed edge from m to n for each pair $m, n \in N$. Each arch between two hosts is labeled with the corresponding *Attack Immunity* measure. It should be noted that if an edge represents more than one service, then the weight of the edge correspond to the combined attack immunity of the services connecting that pair of hosts. Considering that the number of services running on a host cannot exceed the number of ports (and is practically much lower than that), it is straight forward to find the time complexity of this algorithm to be $O(n^2)$. We assume that S_{mn} is calculated off-line before the algorithm is run. A path through the network having a low combined historical immunity value represents a security issue. And, if such a path exists, the system administrator needs to consider possible risk mitigation strategies focusing on that path.

Attack Propagation Calculation: For each node d in D, we want to find how difficult it is for an attacker to compromise all the hosts within reach from d. To do that, we build a minimum spanning tree for d for its segment of the *SCG*. The weight of this minimum spanning tree will represent how vulnerable this segment is to an attack. The more vulnerable the system is, the higher this weight will be. There are several algorithms for finding a minimum spanning tree in a directed graph [5] running in $O(n^2)$ time in the worst case. We define *Service Breach Effect* (SBE_d) to be the weight of the tree rooted at d in D. It actually denotes the damage possible through d. We use the following equation to calculate the SBE_d

$$SBE_d = \sum_{n \in T} \left(\prod_{m \in \text{ nodes from } d \text{ to } n} p_{s_{dm}} \right) \times Cost_n \tag{13}$$

Here $Cost_n$ indicates the cost of damage when host n is compromised and T is the set of hosts present in the spanning tree rooted at host d. Finally, the attack propagation metric of the network is:

$$AP(D) = \sum_{d \in D} P(d) \times SBE_d \tag{14}$$

where, $P(d)$ denotes the probability of the existence of a vulnerability in host d. If the service s_d present in host d is patched then this probability can be derived directly from the expected risk $ER(s_d)$ defined previously in (5). If the service is unpatched and there is existing vulnerabilities in service s_d of host d, then the value of $P(d)$ is 1. The equation is formulated such that it provides us with the expected cost as a result of attack propagation within the network.

2.2.2 Exposure of Unused Address Spaces (ENAS)

Policies should not allow spurious traffic, i.e., traffic destined for *unused* IP addresses or port numbers, to flow inside the network, because this spurious traffic consumes bandwidth and increases the possibility of a DDoS attacks.[1] In our previous work [6, 7], we show how to identify automatically the rules in the security policy that allow for spurious traffic, and once we identify the possible spurious flows, we can readily compute the spurious residual risk for the policy.

Although spurious traffic may not exploit vulnerabilities, it can potentially cause flooding and DDoS attacks and therefore poses a security threat toward the network. In order to accurately estimate the risk of the spurious traffic, we must consider how much spurious traffic can reach the internal network, the ratio of the spurious traffic to the total capacity and the average available bandwidth used in each internal subnet. Assuming that maximum per-flow bandwidth allowed by the firewall policer

[1] We exclude here spurious traffic that might be forwarded only to Honeynets or sandboxes for analysis purpose.

is M_{l_i} Mbps for link l_i of capacity C_{l_i}, and F_{l_i} is the set of all spurious flows passing through link l_i, we can estimate the *Residual Capacity (RC)* for link l_i as follows:

$$RC(l_i) = 1 - \frac{\sum_{f_j \in F_{l_i}} M_{l_i}}{C_{l_i}} \tag{15}$$

The residual capacity has a range [0, 1]. We can now calculate the *Spurious Risk (SPR)* of host d and then sum this measure for all the hosts in network A to get the *SPR* for the entire network :

$$SPR(A) = \sum_{d \in N} \left(c(d) \times \max_{l_i \in L} (1 - RC(l_i)) \right) \tag{16}$$

where $c(d)$ is the weight (i.e., importance or cost) associated with the host d in the network with a range [0,1] and L is the set of links connected to host d. We take the minimum of all the *Residual Capacity* associated with host d to reflect the maximum amount of spurious traffic entering the network and therefore measuring the worst case scenario. This is the spurious residual risk after considering the use of per-flow traffic policing as a counter-measure and assuming that each of the hosts within the network has a different cost value associated with it.

2.2.3 Considering The Network Exposure Factor

In general, the risk of an attack may increase with the exposure of a service to the outside network. When we are using the severity score of a vulnerability, it should be multiplied by the *EF* to take into account for the exposure of the service to the network. This exposure, therefore, influences *EVM*, *HVM* and *PVM* calculation. We call it as *Network Exposure Factor (EF)* and it is calculated as the fraction of the total address space to which a service is exposed to. A service that is exposed to the total possible address space (e.g. *.*.*.*/*ANY*) has a much higher probability of being attacked and exploited than one that is exposed only to the address space of the internal network. The *Exposure Factor (EF)*, of a service s_i considers the number of IP Addresses served by s_i, $IP(s_i)$, the number of ports served by s_i, $PORTS(s_i)$, the total number of IP addresses, 2^{32} and the total number of ports, 2^{16} . Thus the range of this factor will start from the minimum value of 1 for the service totally hidden from the network, and will reach the maximum value of 2 for the service totally exposed to the whole of the address space. Mathematically,

$$EF(s_i) = 1 + \frac{\log_2(IP(s_i) \times PORTS(s_i))}{\log_2(2^{32} \times 2^{16})} \tag{17}$$

assuming that the risk presented by the outside network is distributed uniformly. Of course, if we can identify which part of the address space is more risky than the others, we can modify the numerator of (17) to be a weighted sum. For the purpose of multiplying vulnerability severity score with *EF*, we can redefine the severity score as $SS(v_i)$ = Severity Score of the vulnerability $v_i \times EF(s_{v_i})$, where s_{v_i} is the service affected by the vulnerability v_i. This redefinition allows us to incorporate the

EF in all the equations, where the severity of a vulnerability is being used, i.e., (1), (5) and (9).

2.2.4 AP and SPR Based Risk Mitigation

AP indicates the penetrability of the network. Therefore, if this metric has a high value, it indicates that the network should be partitioned to minimize communication within the network and attack propagation as well. The possible risk mitigation measures that can be taken include (1) Network redesigning (introducing virtual LANs). (2) Rearrange the network so that machines having equivalent risk are in the same partition. (3) Increase the number of enforcement points (Firewalls, IDS). (4) Increase the security around the hot spots in the network and (5) Strengthen MAC sensitivity labels, DAC file permission sets, access control lists, roles or user profiles. In case of *SPR*, High score for this metric indicates that the firewall rules and policies require fine tuning. When the score of this measure rises beyond the threshold level, *ROCONA* adds additional firewall rules to the firewall. The allowable IP addresses and ports need to be checked thoroughly so that no IP address or port is allowed unintentionally.

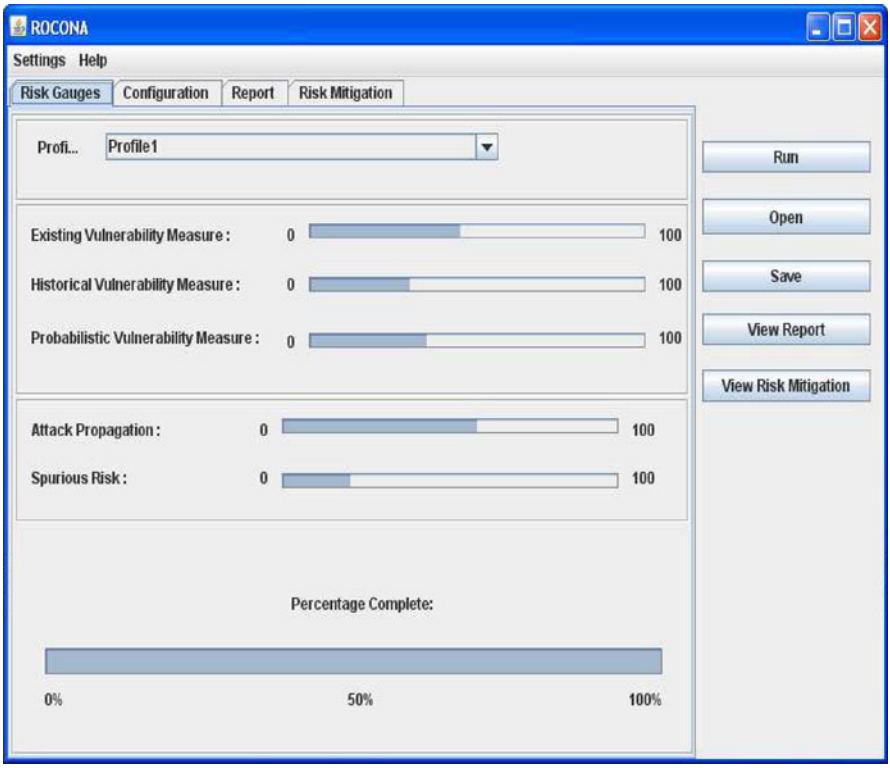

Fig. 2 ROCONA Risk gauges

3 Implementation of ROCONA

To simplify the network risk measurement and mitigation using our proposed framework, we have implemented the measures in a tool called *Risk based prOactive seCurity cOnfiguration maNAger (ROCONA)*. We have used *Java Programming Language* for this implementation. The tool can run as a daemon process and therefore provide system administrators with periodic risk updates. It provides risk scores for each of the 5 components. The measures are provided as risk gauges. The risk is shown as a value between 0 and 100 as can be seen from Fig. 2.

The users can setup different profiles for each run of the risk measurement. They can also configure different options like parameters for *EVM*, *AHVM* and *PVM* and the topology and vulnerability database files to consider during vulnerability measurement. Such parameters include:

1. The α_1 and α_2 weights required for measurement of *EVM*.
2. The decay factor β in the measurement of *AHVM*.
3. The *Prediction Interval* parameter *(T)* in *PVM* measurement.
4. The network topology file describing the interconnections between the nodes within the network.

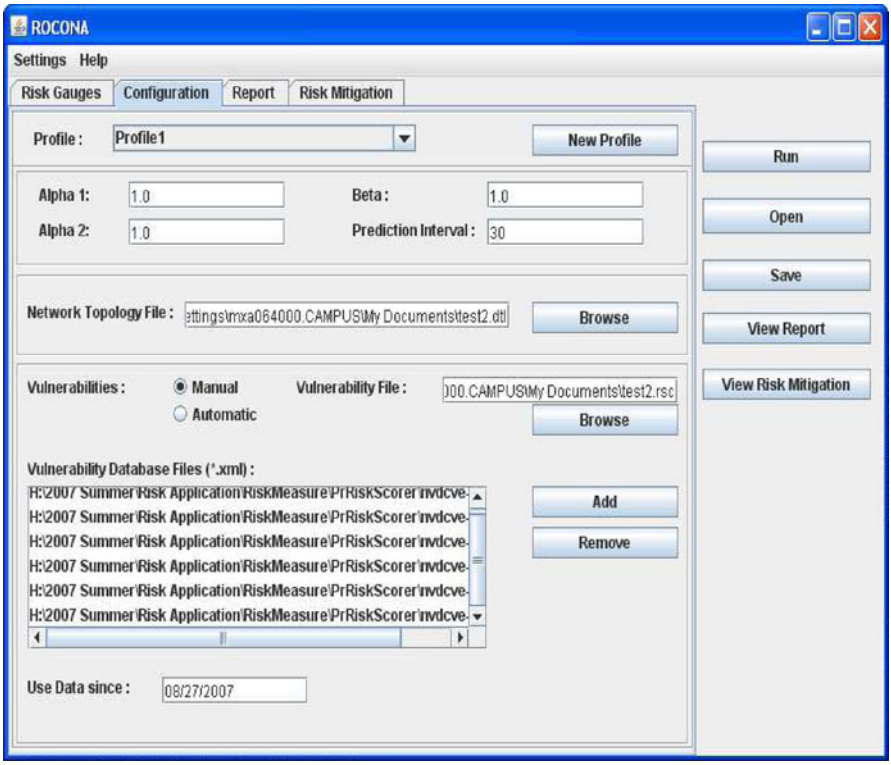

Fig. 3 ROCONA configuration

5. The .xml files containing the list of vulnerabilities from the *NVD*.
6. Options for manually providing the list of *CVE vulnerabilities* present in the system or to automatically extract them from third party vulnerability scanning software. The current version can only use *Nessus* for the automatic option.
7. The starting date from which vulnerabilities are considered (for example, all vulnerabilities from 01/01/2004)

All these configuration options are shown in Fig. 3. After the tool completes its run, it provides the system administrator with the measurement process in details. All the details can be seen in the *Details* tab. Using all the gauges, *ROCONA* also provides risk mitigation strategies, (i.e., which service needs to patched, how rigorous firewall policies should be) in the *Risk Mitigation* tab. All this information can be saved for future reference or for comparison purposes.

3.1 Deployment and Case Study

The *ROCONA* tool has both server and client sides. The client side is installed on individual computers in the system whereas the server part is installed for the administrator of the system. Each client communicates with the server to get the parameter values provided by the administrator. Third party softwares must also be installed so that the client program can extract the services currently running on a particular computer. The client side then sends the scores for individual components to the server where all the scores are combined to provide a unified score for the whole network system. We deployed our tool in two of our computer systems in *The University of Texas at Dallas Database and Data Mining Laboratory* to perform a comparative analysis of the risk for both the machines. Since both the machines are in the same network and under the same firewall rules, the *AP* and *SPR* values have been ignored (i.e., they are equal). The comparison, therefore, includes *AHVM*, *PVM* and *EVM*. The results generated by *ROCONA* are then compared with the results of two well known security tools *AOL Active Security Monitor* [8] and *Nessus*. *AOL Active Security Monitor (ASM)* provides a score based on seven factors, which include factors like firewall, virus protection, spyware protection and p2p software. *Nessus* on the other hand, scans for open ports and how those ports can be used to compromise the system. Based on this, *Nessus* provides a report warning the user of the potential threats. In Table 1, the comparison is provided.

As can be seen from the table, System *B* is more vulnerable than System *A*. For System *B*, all of *AHVM*, *PVM* and *EVM* scores show higher values indicating higher security risk. The same trend can be seen using the other two tools as well. For

Table 1 ROCONA Deployment and evaluation

System	ROCONA			AOL active security center	Nessus		
	AHVM	PVM	EVM		High	Medium	Low
A	16.422	10.76	4.114	64	3	1	0
B	18.57	11.026	5.149	51	7	1	0

Active Security Monitor, higher value indicates lower risk and for *Nessus*, the numbers indicate how many high, medium or low vulnerabilities are in the system. We also performed comparisons between systems where both had the same service and softwares, but one was updated and other was not. In that case only the *EVM* scores were found to be different (the updated system had lower *EVM*) and the *AHVM* and *PVM* scores remained the same.

It may not be feasible to provide side-by-side comparison between *ROCONA* and other tools because *ROCONA* offers unique features in analyzing risk such as measuring the vulnerability history trend and prediction. However, this experiment shows that *ROCONA* analysis is consistent with the results of the other vulnerability analysis tools such as *AOL Active Security Monitor* and *Nessus*. Another important feature of *ROCONA* is its dynamic nature. Since, new vulnerabilities are found frequently, for the same state (i.e., same services and network configuration) of a system, vulnerability scores will be different at two different times (if the two times are significantly apart) as new vulnerabilities may appear for the services in the system. But in such a case, both *Nessus* and *AOL Active Security Monitor* would provide the same scores, as their criteria of security measurement would remain unchanged. So far our knowledge, there is no existing tool that can perform such dynamic risk measurement and therefore, we suffice by providing comparison with these two tools. Here the scores themselves may not provide much information about the risk but can provide a comparison over time regarding the state of security of the system. This allows effective monitoring of the risk towards the system.

4 Quality of Protection Metric (QoPM)

Although the main goal of this research is to devise new factors in measuring network security, we show here that these factors can be used as a vector or scalar elements for evaluating or comparing system security or risk. For a system S, we can combine *EVM(S)*, *AHVM(S)*, *PVM(S)*, *AP(S)* and *SPR(S)* into one *Total Vulnerability Measure*, $\mathbf{TVM}(S)$, as a vector containing all these 5 measures as in (18).

$$\mathbf{TVM}(S) = [\,EVM(S) \quad AHVM(S) \quad PVM(S) \quad AP(S) \quad SPR(S)\,]^T \tag{18}$$

We define the *Quality of Protection Metric* of system S, $\mathbf{QoPM}(S)$, as in (19).

$$\mathbf{QoPM}(S) = 10\big[\, e^{-\gamma_1 EVM(S)} \quad e^{-\gamma_2 AHVM(S)} \quad e^{-\gamma_3 PVM(S)} \quad e^{-\gamma_4 (10-AP(S))} \quad e^{-\gamma_5 SPR(S)} \,\big]^T \tag{19}$$

This will assign the value of $[10\ 10\ 10\ 10\ 10]^T$ to a system with *TVM* of $[0\ 0\ 0\ 0\ 0]^T$. The component values assigned by this equation will be monotonically decreasing functions of the components of the *Total Vulnerability Measure* of the system. The parameters γ_1, γ_2, γ_3, γ_4 and γ_5 provide control over how fast the components of the *Quality of Protection Metric (QoPM)* decreases with the risk factors. The *QoPM* can be converted from a vector value to a scalar value by a suitable transformation, like taking the norm or using weighted averaging. In order to compare two systems,

we can perform the comparison of their corresponding *QoPM* vectors component-wise or we can convert them to scalar values and then compare those scalars. Intuitively, one way to combine these factors is to choose the maximum risk factor as the dominating one (e.g., vulnerability risk vs. spurious risk) and then measure the impact of this risk using the policy resistance measure. Although we advocate generating a combined metric, we also believe that this combination framework should be customizable to accommodate user preferences (e.g., benefit, sensitivity or mission criticality), which, however, is beyond the scope of this paper. Another important aspect of this score is that, for the vulnerability measures (i.e., *EVM*, *AHVM*, *PVM*), higher score indicates higher risk or low security, whereas for *QoPM*, it is the opposite. In case of *QoPM*, higher score indicates higher level of security or lower risk towards the system.

5 Experimentatal Evaluation

We conducted extensive experiments for evaluating our metric using real vulnerability data. In contrast to other similar research works that present studies on a few specific systems and products, we experimented using publicly available vulnerability databases. We evaluated and tested our metric, both component-wise and as a whole, on a large number of services and randomly generated policies. In our evaluation process, we divided the data into training sets and test sets. In the following sections, we describe our experiments and present their results.

5.1 Vulnerability Database

In our experiments, we used the *National Vulnerability Database (NVD)* published by National Institute of Science and Technology (NIST) which is available at http://nvd.nist.gov/download.cfm. The *NVD* provides a rich array of information that makes it the vulnerability database of choice. First of all, all the vulnerabilities are stored using the standard CVE (Common Vulnerabilities and Exposures) name http://cve.mitre.org/. For each vulnerability, the *NVD* provides the products and versions affected, descriptions, impacts, cross-references, solutions, loss types, vulnerability types, the severity class and score, etc. The *NVD* severity score has a range of (0, 10). If severity value is from 0 to 4, it is considered as low severity, scores up to 7 is

Table 2 Summary statistics of the NVD vulnerability database

Total number of entries	23542
Total number of valid entries	23309
Total number of rejected entries	233
Total number of distinct products	10375
Total number of versions of the products	147640
Total number of distinct vendors	6229
Earliest entry	10/01/1988
Latest entry	04/05/2007

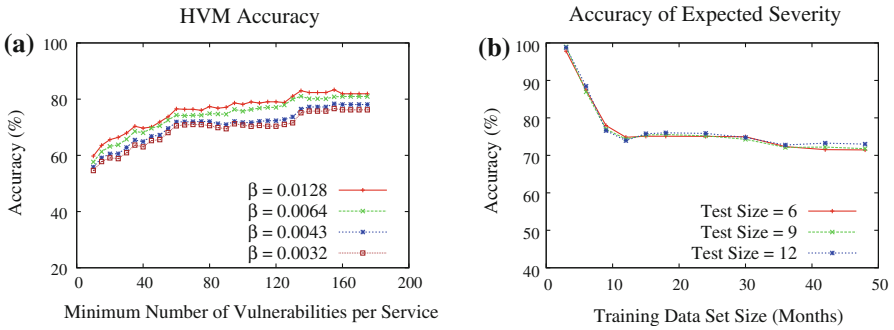

Fig. 4 (**a**) Accuracy of the HVM for different values of β and minimum number of vulnerabilities. (**b**) Training data set size versus accuracy graph for the expected severity calculation for different values of the test data set size

considered medium severity and a value higher than 7 indicates high severity. We have used the database snapshot updated at 04/05/2007. We present some summary statistics about the *NVD* database snapshot that we used in our experiments in Table 2.

For each vulnerability in the database, *NVD* provides *CVSS* scores [9] for vulnerabilities in the range 1 to 10. The severity score is calculated using the *Common Vulnerability Scoring System (CVSS)*, which provides a base score depending on several factors like impact, access complexity, required authentication level, etc.

5.2 HVM Validation

We conducted an experiment to evaluate the *HVM* score according to (1), with the hypothesis that if service *A* has a higher *HVM* than service *B*, then in the next period of time, service *A* will display a higher number of vulnerabilities than *B*.

In our experiment, we used vulnerability data upto 06/30/2006 to compute the *HVM* of the services, and used the rest of the data to validate the result. We varied β so that the decay function falls to 0.1 in 0.5, 1 , 1.5 and 2 years respectively, and observed the best accuracy in the first case. Here, we first chose services with at least 10 vulnerabilities in their lifetimes, and gradually increased this lower limit and observed that the accuracy increases with the lower limit. As expected of a historical measure, better results have been found when more history is available for the services and observed the maximum accuracy of 83.33%. The graph in Fig. 4a presents the results of this experiment.

5.3 Expected Risk (ER) Validation

The experiment for the validation of *Expected Risk (ER)*, is divided into a number of parts. First, we conducted experiments to evaluate the different ways of calculating the probability in (3). We conducted experiments to compare the accuracies obtained by the exponential distribution, empirical distribution and time series analysis method. Our general approach was to partition the data into training and

test data sets, compute the quantities of interest from the training data sets and validate the computed quantity using the test data sets. Here, we obtained the most accurate and stable results using *Exponential CDF*.

The data used in the experiment for *Exponential CDF* was the interarrival times for the vulnerability exposures for the services in the database. We varied the length of the training data and the test data set. We only considered those services that have at least 10 distinct vulnerability release dates in the 48 months training period. For *Expected Severity*, we used similar approach. For evaluating *Expected Risk (ER)*, we combined the data sets for the probability calculation methods and the data sets of the expected severity.

In the experiment for *Exponential CDF*, we constructed an exponential distribution for the interarrival time data and computed (3) using the formula in (7). For each training set, we varied the value of T , and ran validation for each value of T with the test data set. Fig. 5a presents the accuracy of the computed probabilities using *Exponential CDF* method for different training data set sizes and different values of the *Prediction Interval* parameter, T. In Fig. 5b, we have

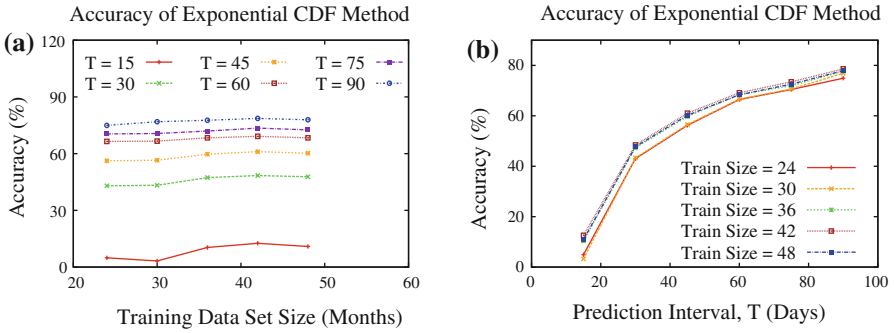

Fig. 5 (**a**) Training data set size versus accuracy graph for the exponential CDF method for probability calculation, test set size = 12 months. (**b**) Prediction interval parameter (T) versus accuracy graph for the exponential CDF method for probability calculation, test set size = 12 months

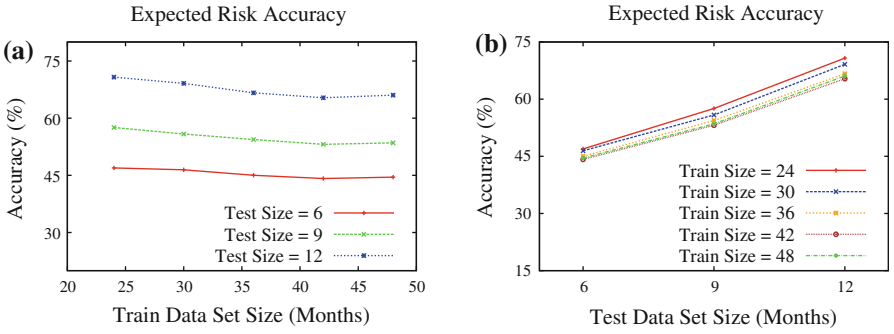

Fig. 6 (**a**) Results of ER validation with training data set size versus accuracy for different test data set sizes. (**b**) Results of ER validation with test data set size versus accuracy for different training data set sizes

presented the accuracies on the *Y* axis against the *Prediction Interval* parameter *T* on the *X* axis. For the test data set size of 12 months, we observed the highest accuracy of 78.62% with the 95% confidence interval being $[72.37, 84.87]$ for the training data set size of 42 months and *Prediction Interval* of 90 days. We present the results of the experiment for *Expected Severity* in Fig. 4b. The maximum accuracy obtained in this experiment was 98.94% for the training data set size of 3 months and the test data set size of 9 months. The results of the *Expected Risk* experiment are presented in the Fig. 6a and b. For the *Expected Risk*, we observed the best accuracy of 70.77% for training data set size of 24 months and test data set size of 12 months with the 95% confidence interval of $[64.40, 77.14]$.

It can be observed in Fig. 5b that the accuracy of the model increases with increasing values of the *Prediction Interval* parameter *T*. This implies that this method is not sensitive to the volume of training data available to it. From Fig. 4b, it is readily observable that the accuracy of the *Expected Severity* is not dependent on the test data set size. It increases quite sharply with decreasing values of training data set data size. This means that the expectation calculated from the most recent data is actually the best model for the expected severity in the test data.

5.4 QoPM Validation

To validate the *QoPM*, we need to evaluate policies using our proposed metric in (19). In absence of other comparable measures of system security, we used the following hypothesis – if system *A* has a better *QoPM* than system *B* based on training period data, then system *A* will have less number of vulnerabilities than system *B* in the test period. We assume that the *EVM* component of the measure will be 0 as any existing vulnerability can be removed.

In generating the data set, we chose the reference date separating the training and test periods to be 10/16/2005. We used the training data set for *ER* using *Exponential CDF* method. In the experiment, we generated a set of random policies and for each policy we evaluated (19). We chose $\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = \gamma_5 = 0.06931472$ so that the *QoPM* for a *TVM* of $[10\ 10\ 10\ 10\ 10]^T$ is $[5\ 5\ 5\ 5\ 5]^T$. Then, for each pair of

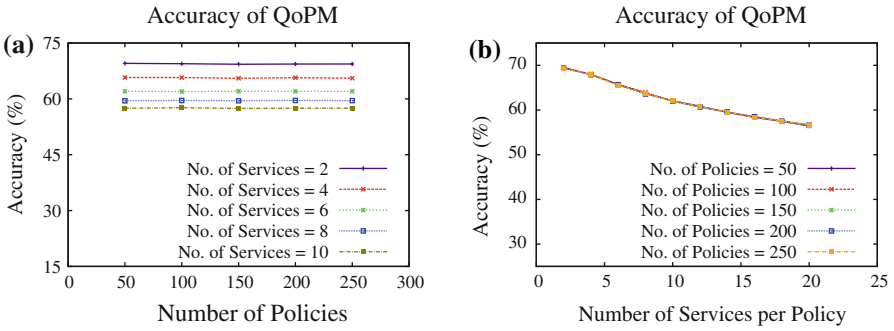

Fig. 7 (**a**) Number of policies versus accuracy graph for QoPM for different values of the number of services in each policy. (**b**) Number of services per policy versus accuracy graph for QoPM for different values of the number of policies

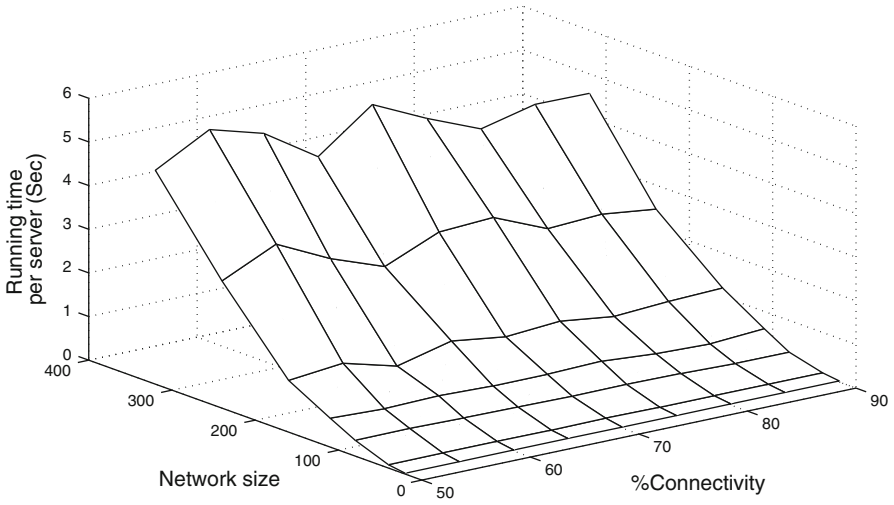

Fig. 8 The average running times for different network sizes and %connectivity

systems A and B, we evaluated the hypothesis that if $\mathbf{QoPM}(A) \geq \mathbf{QoPM}(B)$ then, the number of vulnerabilities for service A should be less than or equal to the number of vulnerabilities for service B. In our experiment, we varied the number of policies from 50 to 250 in steps of 25. In generating the policies, we varied the number of services per system from 2 to 20 in increments of 2.

We present the results obtained by the experiment in Fig. 7a and b. As mentioned previously, a policy can be regarded as a set of rules indicating which services are allowed access to the network traffic. We set up different service combinations and consider them as separate policies for our experiments. We can observe from the graph in Fig. 7a that the accuracy of the model is not at all sensitive to the variation in the total number of policies. However, the accuracy does vary with the number of services per policy – the accuracy decreases with increasing number of services per policy. This trend is more clearly illustrated in Fig. 7b where the negative correlation is clearly visible.

5.5 Running Time Evaluation of Attack Propagation Metric

To assess the feasibility of calculating the *AP* metric under different conditions, we ran a *MatLab* implementation of the algorithm for different network sizes, as well as different levels of network connectivity percentages (the average percentage of the network directly reachable from a host). The machine used to run the simulation had a 1.9 GHz P-IV processor with 768MB RAM. The results are shown in Fig. 8. For each network size, we generated several random networks with the same %connectivity value. The running time was then calculated for several hosts within each network. The average value of the running time per host was then calculated and used in Fig. 8. As can be seen from the figure, the quadratic growth of the running time against network connectivity was not noticeably dependent on the

network %connectivity in our implementation. The highest running time per host for a network of 320 nodes was very reasonable at less than 5 seconds. Thus, the algorithm scales gracefully in practice and is thus feasible to run for a wide range of network sizes.

6 Related Work

Measurement of network and system security has always been an important aspect of research. Keeping this in mind, many organizational standards have been evolved to evaluate the security of an organization. National Security Agency's (NSA) INFOSEC Evaluation Methodology (IEM) can be mentioned as an example in this respect. Details regarding the methodology can be found in [10]. In [11] NIST provides a guidance to measure and strengthen the security through the development and use of metrics, but their guideline is focused on the individual organizations and they do not provide any general scheme for quality evaluation of a policy. There are some professional organizations as well as vulnerability assessment tools including Nessus, NRAT, Retina, Bastille and others [12]. They actually try to find out vulnerabilities from the configuration information of the concerned network. However, all these approaches usually provide a report telling what should be done to keep the organization secure and they do not consider the vulnerability history of the deployed services or policy structure.

There has been a lot of research in the security policy evaluation and verification as well. Evaluation of VPN, Firewall and Firewall security policies include [6, 7, 13]. Attack graphs is another technique that is well developed to assess the risks associated with network exploits. The implementations normally require intimate knowledge of the steps of attacks to be analyzed for every host in the network [14, 15]. In [16] the authors, however, provide a way to do so even when the information is not complete. Still, this setup causes the modeling and analysis using this model to be highly complex and costly. Mehta et al. try to rank the states of an attack graph in [17]. But their work do not give any sort of prediction of future risks associated with the system and also, they do not consider the policy resistance of firewall and IDS. Our technique that is employed to assess the policy's immunity to attack propagation implements a simpler and less computationally expensive technique that is more suitable for our goal.

There has been some research focusing on the attack surface of a network. Mandhata et al. in [18] have tried to find the attack surface from the attackability of a system. Another work based on attack surface has been done by Howard et al. in [19] along channel, methods and data dimensions and the contributions of these dimensions in an attack. A. Atzeni et al. present a generic overall framework for network security evaluation in [20], and discuss the importance of security metrics in [21]. In [22] Pamula propose a security metric based on the weakest adversary (i.e. the least amount of effort required to make an attack successful). In Alhazmi et al. [1], present their results on the prediction of vulnerabilities and they present their work on vulnerability discovery process in [23]. They argue that vulnerabilities are essentially defects in released software and uses past data and track records of

the development team, code size, records of similar softwares etc. to predict the number of vulnerabilities. Our work is more general in this respect and utilizes publicly available data.

There has also been some research work that focus on hardening the network. Wang et al. use attack graphs for this purpose in [24]. They also attempt to predict future alerts in multistep attacks using attack graph [25]. A previous work of hardening the network was done by Noel et al. [26] where they made use of dependencies among exploits. They use the graphs to find some initial conditions that, when disabled, will achieve the purpose of hardening the network. Sahinoglu et al. propose a framework in [27, 28] for calculating existing risk depending on present vulnerabilities in terms of threat represented as probability of exploiting this vulnerability and the lack of counter-measures. But all these work do not represent the total picture as they predominantly try to find existing risk and do not address how risky the system will be in the near future or how policy structure would impact on security. Their analysis regarding security policies can not be regarded as complete and they lack the flexibility in evaluating them.

A preliminary investigation of measuring the existing vulnerability and some historical trends have been analyzed in a previous work [29]. That work was still limited in analysis and scope.

7 Conclusions

As network security is of utmost importance for an organization, a unified policy evaluation metric will be highly effective in assessing the protection of the current policy, and justifying consequent decisions to strengthen security. In this paper, we present a proactive approach to quantitatively evaluate security of network systems by identifying, formulating and validating several important factors that greatly affect its security. Our experiments validate our hypothesis that if a service has a highly vulnerability prone history, then there is higher probability that the service will become vulnerable again in the near future. These metrics also indicate how the internal firewall policies affect the security of the network as a whole. These metrics are useful not only for administrators to evaluate policy/network changes and, take timely and judicious decisions, but also for enabling *adaptive security systems* based on vulnerability and network changes.

Our experiments provide very promising results regarding our metric. Our vulnerability prediction model proved to be up to 78% accurate, while the accuracy level of our historical vulnerability measurement was 83.33% based on real-life data from *National Vulnerability Database (NVD)*. The accuracies obtained in these experiments vindicate our claims about the components of our metric and also the metric as a whole. Combining all the measures into a single metric and performing experiments based on this single metric also provides us with an idea of its effectiveness.

Acknowledgments The authors would like to thank Muhammad Abedin and Syeda Nessa of The University of Texas at Dallas for their help with the formalization and experiments making this work possible.

References

1. Alhazmi, O.H., Malaiya Y.K.: Prediction capabilities of vulnerability discovery models. In: Proceedings of reliability and maintainability symposium, Jan 2006, pp. 86–91
2. National institute of science and technology (nist), http://nvd.nist.gov
3. Lee, S.C., Davis, L.B.: Learning from experience: operating system vulnerability trends, IT Professional, **5**(1), Jan/Feb 2003
4. Abedin, M., Nessa, S., Al-Shaer, E., Khan, L.: Vulnerability analysis for evaluating quality of protection of security policies, In: 2nd ACM CCS workshop on quality of protection, Alexandria, Virginia, Oct 2006
5. Bock, F.: An algorithm to construct a minimum directed spanning tree in a directed network, In: Developments in Operations Research. Gordon and Breach, pp. 29–44 (1971)
6. Al-Shaer, E., Hamed, H.: Discovery of policy anomalies in distributed firewalls. In: Proceedings of IEEE INFOCOM'04, March 2004
7. Hamed, H., Al-Shaer, E., Marrero, W.: Modeling and verification of ipsec and vpn security policies. In Proceedings of IEEE ICNP'2005, Nov 2005
8. Aol software to improve pc security, http://www.timewarner.com/corp/newsroom/pr/0,20812,1201969,00.html
9. Schiffman, M.: A complete guide to the common vulnerability scoring system (cvss). http://www.first.org/cvss/cvss-guide.html, June 2005
10. Rogers, R., Fuller, E., Miles, G., Hoagberg, M., Schack, T. Dykstra, T., Cunningham, B.: Network Security Evaluation Using the NSA IEM, 1st ed. Syngress Publishing, Inc., Aug 2005
11. Swanson, M., Bartol, N., Sabato, J., Hash, J., Graffo, L.: Security Metrics Guide for Information Technology Systems. National Institute of Standards and Technology, Gaithersburg, MD 20899-8933, July 2003
12. "10 network security assessment tools you can't live without" http://www.windowsitpro.com/Article/ArticleID/47648/47648.html?Ad=1
13. Kamara, S., Fahmy, S., Schultz, E., Kerschbaum, F., Frantzen, M.: Analysis of vulnerabilities in internet firewalls. Comput. Secur. **22**(3), 214–232 (2003)
14. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: CCS '02: Proceedings of the 9th ACM conference on computer and communications security, pp. 217–224, ACM Press, New York, NY, USA (2002)
15. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: NSPW '98: Proceedings of the 1998 workshop on new security paradigms, pp. 71–79, ACM Press, New York, NY, USA (1998)
16. Feng, C., Jin-Shu, S.: A flexible approach to measuring network security using attack graphs. In: International symposium on electronic commerce and security, April 2008
17. Mehta, C.B.V., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Recent Advances in Intrusion Detection 2006, Hamburg, Germany, Sept 2006
18. Manadhata, P. Wing, J.: An attack surface metric. In: First Workshop on Security Metrics, Vancouver, BC, August 2006
19. Howard, M., Pincus, J., Wing, J.M.: Measuring relative attack surfaces. In: Workshop on Advanced Developments in Software and Systems Security, Taipei, Dec 2003
20. Atzeni, A., Lioy, A., Tamburino, L.: A generic overall framework for network security evaluation. In: Congresso Annuale AICA 2005, Oct 2005, pp. 605–615
21. Atzeni, A., Lioy, A.: Why to adopt a security metric? A little survey. In: QoP-2005: Quality of protection workshop. Sept 2005
22. Pamula, J., Ammann, P., Jajodia, S., Swarup, V.: A weakest-adversary security metric for network configuration security analysis. In: ACM 2nd workshop on quality of protection 2006, Alexandria, VA, Oct 2006
23. Alhazmi, O.H., Malaiya, Y.K.: Modeling the vulnerability discovery process. In: Proceedings of international symposium on software reliability engineering, Nov 2005
24. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. In: Computer Communications. Alexandria, VA, Nov 2006
25. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. In: Computer Communications, Sept 2006

26. Noel, S., Jajodia, S., O'Berry, B., Jacobs, M.: Efficient minimum-cost network hardening via exploit dependency graphs. In: 19th annual computer security applications conference, Las Vegas, Nevada, Dec 2003
27. Sahinoglu, M.: Security meter: a practical decision-tree model to quantify risk. In: IEEE Security and Privacy, June 2005
28. Sahinoglu, M.: Quantitative risk assessment for dependent vulnerabilities. In: Reliability and maintainability symposium, June 2005
29. Ahmed, M.S., Al-Shaer, E., Khan, L.: A novel quantitative approach for measuring network security. In: INFOCOM'08, April 2008

Author Biographies

Mohammad Salim Ahmed is currently working toward his Ph.D. degree in the Department of Computer Science at the University of Texas at Dallas (UTD). He graduated from Bangladesh University of Engineering and Technology with BS in Computer Science and Engineering degree in 2005. He was also a Lecturer of the same university. His research interests are in text mining, machine learning, and network security measurement using data mining. His recent research focuses on developing data mining techniques to classify high dimensional text data. He has published more than 5 research papers in peer reviewed conferences and workshops including INFOCOM, NOMS, DDDM and CIDU.

Ehab Al-Shaer is an Associate Professor and the Director of the Cyber Defense and Network Assurability (CyberDNA) Center in the School of Computing and Informatics at University of North Carolina Charlotte. His primary research areas are network security, security management, fault diagnosis, and network assurability. Al-Shaer edited/co-edited more than 10 books and book chapters, and published many refereed journals and conference papers in his area. Prof. Al-Shaer is the General Chair of ACM Computer and Communication 2009-2010 and NSF Workshop in Assurable and Usable Security Configuration, August 2008. Prof. Al-Shaer also served as a Workshop Chair and Program Co-chair for number of well-established conferences/workshops in his area including POLICY 2008, IM 2007, ANM-INFOCOM 2008, CCS-SafeConfig 09, MMNS 2001, and E2EMON 04-05. He also served as a member in the technical program and organization committees for many IEEE and ACM conferences. He was awarded many Best Paper Awards. Prof. Al-Shaer received his MSc and Ph.D. in Computer Science from the Northeastern University (Boston, MA) and Old Dominion University (Norfolk, VA) in 1998 and 1994 respectively.

Mohamed Taibah is a PhD student at DePaul University. He had his BSc and MS degrees in Electrical Engineering from King Fahd University of Petroleum and Minerals, Saudi Arabia and Northwestern University, Chicago, USA in 1996 and 1999 respectively. His area of research is worm control, botnet detection and risk measurement. He is currently working with Cisco

Latifur Khan is currently an Associate Professor in the Computer Science department at the University of Texas at Dallas (UTD), where he has taught and conducted research since September 2000. He received his Ph.D. and M.S. degrees in Computer Science from the University of Southern California, in August of 2000, and December of 1996 respectively. His research work is supported by grants from NASA, the Air Force Office of Scientific Research (AFOSR), National Science Foundation (NSF), IARPA, the Nokia Research Center, Raytheon, Alcatel, and the SUN Academic Equipment Grant program. In addition, Dr. Khan is the director of the state-of-the-art DBL@UTD, UTD Data Mining/Database Laboratory, which is the primary center of research related to data mining, and image/video annotation at University of Texas-Dallas. Dr. Khan's research areas cover data mining, multimedia information management, semantic web and database systems with the primary focus on first three research disciplines. He has served as a committee member in numerous prestigious conferences, symposiums and workshops including the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Dr. Khan has published over 130 papers in journals and conferences.