Measuring Security Risk of Networks Using Attack Graphs

STEVEN NOEL, SUSHIL JAJODIA
Center for Secure Information Systems, George Mason University, Fairfax, Virginia, USA
LINGYU WANG
Concordia University, Montreal, Quebec, Canada
ANOOP SINGHAL
National Institute of Standards and Technology, Gaithersburg, Maryland, USA

Today's computer systems face sophisticated attackers who combine multiple vulnerabilities to penetrate networks with devastating impact. The overall security of a network cannot be determined by simply counting the number of vulnerabilities. To accurately assess the security of networked systems, one must understand how vulnerabilities can be combined to stage an attack. We model such composition of vulnerabilities through attack graphs. By simulating incremental network penetration, and propagating attack likelihoods, we measure the overall security of a networked system. From this, we score risk mitigation options in terms of maximizing security and minimizing cost. We populate our attack graph models from live network scans and databases that have knowledge about properties such as vulnerability likelihood, impact, severity, and ease of exploitation. Our flexible model can be used to quantify overall security of networked systems, and to study cost/benefit tradeoffs for analyzing return on security investment.

Keywords: Network security risk assessment, attack graphs, optimal risk mitigation options

1. INTRODUCTION

The traditional binary view of information security is that the system is either secure or not secure. But such a simplistic view makes it difficult to answer important questions such as "are we more secure than yesterday" or "how should we invest our limited security resources." Decision makers in other areas of business and engineering often use metrics for determining whether a projected return on investment justifies its costs. Spending for new cyber-security measures is such an investment.

By increasing security spending, an organization can decrease the risk associated with security breaches. But such tradeoff analysis requires quantitative rather than qualitative models. Previous approaches to security metrics have focused on individual vulnerabilities. Metrics such as percentage of patched systems ignore interactions among network vulnerabilities.

While useful, such metrics are limited, because vulnerabilities in isolation lack context. Attack-

This material is based upon work supported by the National Institute of Standards and Technology Computer Security Division under grant 60NANB9D9192; by the Army Research Office MURI award number W911NF-09-1-0525 administered by The Pennsylvania State University; by the National Science Foundation under the grants CT-20013A, CT-0716567, CT-0716323, and CT-0627493; by the Air Force Office of Scientific Research under grants FA9550-07-1-0527, FA9550-09-1-0421, and FA9550-08-1-0157; by the Army Research Office DURIP award W911NF-09-01-0352; by the Natural Sciences and Engineering Research Council of Canada under Discovery Grant N01035, and by Fonds de recherche sur la nature et les technologies. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations. Authors' addresses: Steven Noel and Sushil Jajodia, Center for Secure Information Systems, George Mason University, Fairfax, Virginia, USA 22030-4422, USA, email: {snoel,jajodia}@gmu.edu. Lingyu Wang, Concordia Institute for Information Systems Engineering, 1515 St. Catherine West EV007.629, Montreal, Quebec, Canada H3G 2W1, email: wang@ciise.concordia.ca. Anoop Singhal, Computer Security Division, National Institute of Standards and Technology, 100 Bureau Drive, Stop 1070, Gaithersburg, Maryland, USA 20899, email: anoop.singhal@nist.gov.

ers can combine related vulnerabilities to incrementally penetrate networks, potentially leading to devastating consequences.

We meet this challenge by capturing vulnerability interdependencies, measuring security in the exact way that real attackers could penetrate the network. We analyze all attack paths through a network, providing a metric of overall system risk. Through this metric, we analyze tradeoffs between security costs and security benefits. Decision makers can therefore avoid over investing in security measures that do not pay off, or under investing and risk devastating consequences. Our metric is consistent, unambiguous, makes underlying assumptions explicit, and provides context for understanding security risk alternatives.

Good metrics can be measured consistently, are inexpensive to collect, are expressed numerically, have units of measure, and have specific context [Jaquith et al. 2007]. Security metrics have been suggested based on criteria compliance, intrusion detection, security policy, security incidents, and actuarial modeling.

Statistical methods (Markov modeling, Bayesian networks, etc.) have been used in measuring network security. Complementary to our approach, measurements of attack resistance [Wang et al. 2007] and weakest successful adversary [Pamula et al. 2006] have been proposed. Early standardization efforts in the defense community evolved into the system security engineering capability maturity model [SSE-CMM], although it does not assign quantitative measures. The National Institute of Standards and Technology (NIST) describes a security metrics implementation process [Swanson et al. 2003], and principles for establishing a security baseline [Stoneburner et al. 2004]. There are also standardization efforts for vulnerability assessment, such as the Common Vulnerability Scoring System [CVSS], although these treat vulnerabilities in isolation, without considering attack interdependencies on target networks. In early work on attack graphs, model checking was used to decide whether a given goal state is reachable from the initial state, and to enumerate attack sequences linking the two states. Because the number of sequences can increase exponentially, we use a quadratic representation that encodes them implicitly through exploit interdependencies, with efficient high-level abstractions for groups of fully-connected hosts [Noel et al. 2004]. Such graphs can be generated for tens of thousands of hosts within a few seconds.

In our approach, causal relationships among exploits are based on a well-accepted attack graph model. Attack graphs can be generated using readily available tools, such as the Topological Vulnerability Analysis (TVA) system [Jajodia et al. 2009]. The existence of such powerful and efficient tools justifies the practicality of our attack graph metrics. We adopt a straightforward approach towards the logical relationships among exploits and their likelihoods. The information our model needs already exists in various standards and commercial products, further supporting the practicality of our approach.

The remainder of this paper is organized as follows. Section 2 reviews attack graphs, and Section 3 describes how they can be used to measure likelihood of attack. Section 4 demonstrates security risk analysis using attack graphs, and Section 5 extends this to analysis of return on security investment. Section 6 shows how to quantify confidence in our analyses, and Section 7 summarizes the results.

2. ATTACK GRAPHS

The attack graphs model how multiple vulnerabilities may be combined for an attack. They represent system states using a collection of security-related conditions, such as the existence of vulnerability on a particular host or the connectivity between different hosts. Vulnerability exploitation is modeled as a transition between system states.

As an example, consider Figure 1. The left side shows a network configuration, and the right side shows the attack graph for compromise of the database server by a malicious workstation user. In the network configuration, the firewall is intended to help protect the internal network. The internal file server offers file transfer (ftp), secure shell (ssh), and remote shell (rsh) services.

The internal database server offers ftp and rsh services. The firewall allows ftp, ssh, and rsh traffic from a user workstation to both servers, and blocks all other traffic.

In the attack graph, attacker exploits are blue ovals, with edges for their preconditions and postconditions. The numbers inside parentheses denote source and destination hosts. Yellow boxes are initial network conditions, and the green triangle is the attacker's initial capability. Conditions induced by attacker exploits are plain text. The overall attack goal is a red octagon. The figure also shows the direct impact of blocking ssh or rsh traffic (to the fileserver) through the firewall, i.e., preventing certain exploits in the attack graph.

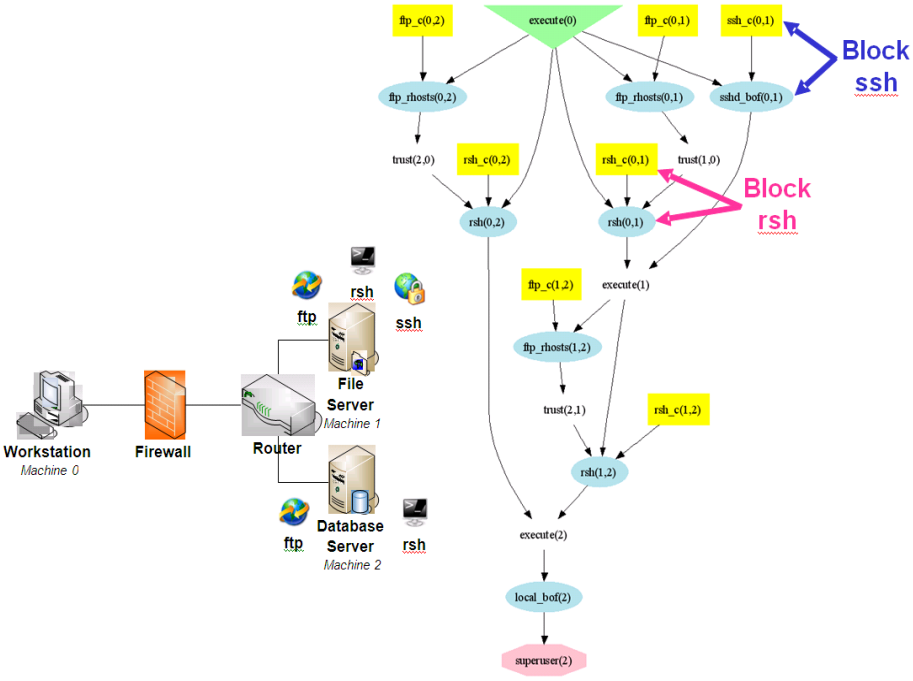


Figure. 1: Example network, attack graph, and network hardening choices

The attack graph includes these attack paths:

(1) sshd_bof(0,1) → ftp_rhosts(1,2) → rsh(1,2) → local_bof(2)
(2) ftp_rhosts(0,1) → rsh(0,1) → ftp_rhosts(1,2) rsh(1,2) → local_bof(2)
(3) ftp_rhosts(0,2) → rsh(0,2) → local_bof(2)

The first attack path starts with sshd_bof(0,1). This indicates a buffer overflow exploit executed from Machine 0 (the workstation) against Machine 1 (the file server), i.e., against its secure shell service. In a buffer overflow attack, a program is made to erroneously store data beyond a fixed-length buffer, overwriting adjacent memory that holds program flow data. The result of the sshd_bof(0,1) exploit is that the attacker can execute arbitrary code on the file server. The ftp_rhosts(1,2) exploit is now possible, meaning that the attacker exploits a particular ftp vulnerability to anonymously upload a list of trusted hosts from Machine 1 (the file server) to Machine 2 (the database server). The attacker can leverage this new trust to remotely execute shell commands on the database server, without providing a password, i.e., the rsh(1,2) exploit. This exploit establishes the attacker's control over the database server as a user with regular privileges. A local buffer overflow exploit is then possible on the database server, which runs in the context of a privileged process. The result is that the attacker can execute code on the database server with full privileges.

3. MEASURING ATTACK LIKELIHOOD

In practice, vulnerabilities often remain in a network, even after they are discovered. Vendors may be slow to release software patches, or deployment may be delayed because of excessive cost or effort. Attackers often leverage even correctly functioning network services to gain new capabilities. An organization will often trade security risk for availability of services.

Our attack graph metric quantifies this risk through measuring the likelihood that such residual paths may eventually be realized by attackers. When a network is more secure, attack likelihood is reduced. Preventing exploits (as in Figure 1) removes certain paths, in turn reducing attack likelihood. When the attacker cannot reach the goal, our metric is zero. When the attacker is completely assured of reaching the goal, the metric is unity.

An attack graph naturally encodes both conjunctive and disjunctive attack relationships. For example, in Figure 1, the attacker cannot upload the list of trusted hosts if the ftp service does not exist; neither can this happen if the attacker cannot use Machine 1 as a normal user. Such a relationship is conjunctive. On the other hand, if a condition can be satisfied in more than one way, it does not matter which path the attacker follows to satisfy it, making the relationship disjunctive.

To compute our metric, we propagate exploit measures through the attack graph, from initial conditions to goal, according to conjunctive and disjunctive dependencies. When one exploit must follow another in a path, this means both are needed to eventually reach the goal (conjunction), so their measures are multiplied, i.e., p(A and B) = p(A)p(B). When a choice of paths is possible, either is sufficient for reaching the goal (disjunction), i.e., p(A or B) = p(A) + p(B) - p(A)p(B). Paths coming into an exploit may form arbitrary logical expressions. In such cases, we propagate exploit measures through corresponding conjunctive/disjunctive combinations. Also, we resolve any attack graph cycles through their distance from the initial conditions, and remove any parts of the graph that do not ultimately lead to the goal.

Our exploit measures could be Boolean variables, real numbers, or even probability distributions. Defining the actual measures (value or distribution) of each individual exploit is part of the ongoing effort of keeping our database of modeled exploits current. A number of vendors and open sources provide ongoing descriptions of reported vulnerabilities, which we can leverage for our model population. A particularly rich resource of vulnerability data is Symantec's DeepSight Threat Management System [SYMANTEC], covering 18,000 distinct versions of 4,200 products from 2,200 vendors, based on 150 authoritative sources, with numerical ratings in the areas of urgency, severity, impact, and ease of exploitation.

4. RISK ANALYSIS USING ATTACK GRAPHS

In practice, vulnerabilities often remain in a network, even after they are discovered. Vendors may be slow to release software patches, or deployment may be delayed because of excessive cost or effort. Attackers often leverage even correctly functioning network services to gain new capabilities. An organization will often trade security risk for availability of services.

Removing attack paths reduces options for an attacker, but at what cost to the organization? For example, in Figure 1, blocking ssh or rsh traffic disables certain initial network conditions, preventing exploits that depend on these vulnerable connections, thereby reducing the number of attack paths. But what is lacking is a clear measure of exactly how security has been improved in each case. For example, which is better, blocking ssh or blocking rsh? Are these significant or only marginal improvements over the current configuration? These are the kinds of questions that are answered though our attack graph metric.

For risk analysis, we simulate attack graphs through Monte Carlo methods. This allows us to handle uncertain input values, by modeling their distributions through statistics such as numerical ranges, mean values, etc. We then simulate attack graphs with inputs generated according to the probability distribution model. This allows us to make optimal choices against complex cyber-attack models that cannot otherwise be easily identified. Monte Carlo methods are ideal for such

problems with highly complex logical (non-linear) relationships with many input variables. In our case, the attack graph is computed once, and then evaluated for each Monte Carlo sample.

In Figure 1, the network hardening options are to block either ssh or rsh traffic (to the file server) through the firewall. Here, we assume that no software patches or other forms of mitigation are available for the ftp and buffer overflow vulnerabilities on the inner network. The remote shell service is working correctly (even though it is being leveraged in the attack), and needs no patching. Assume that the workstation needs access to the ftp services on the inner network, i.e., we avoid blocking ftp traffic through the firewall. Further, we assume that the workstation needs at least one type of shell access (either remote shell or secure shell) to each machine on the inner network.

Figure 2 shows the attack graphs resulting from blocking ssh (left side) and rsh traffic (right side) from the workstation to the file server. In other words, for each network hardening choice, a particular set of attack paths still remains, from which we compute our metric. In the figure, the cumulative likelihood of each exploit is shown, as propagated through the attack graph. For example, the likelihood of rsh(0,2) occurring is the conjunctive combination (Boolean AND) of ftp_rhosts(0,2) and rsh(0,2), while the likelihood of local_bof(2) is the disjunctive combination (Boolean OR) of rsh(0,2) and rsh(1,2). Since local_bof(2) yields the overall goal for this attack scenario (compromise of the database server), the likelihood of local_bof(2) occurring is the overall attack graph metric.

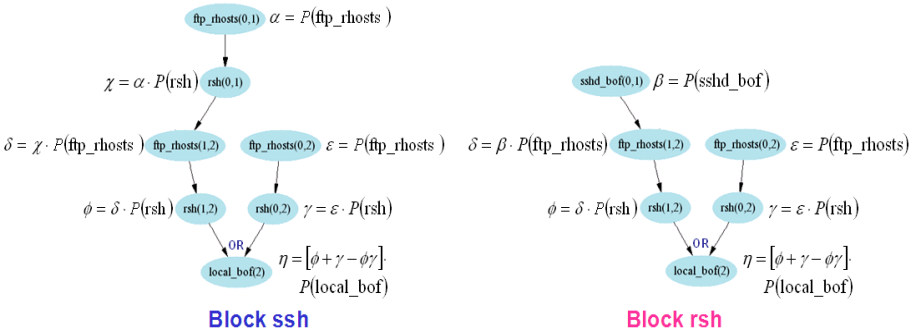


Figure. 2: Residual attack graphs for network configuration choices

In Figure 2, the model inputs P(o) represent the inherent chance of each exploit occurring, independent of other exploits. For example, these could be relative frequencies of events observed on a network over a period of time. Or, they could be taken from a product such as Symantec DeepSight. In this particular case, let us assume we have a minimum and maximum likelihood value for each exploit. We model these likelihoods as uniformly distributed between their minimum and maximum values, i.e., ftp_rhosts and rsh between [0.5, 0.6], and sshd_bof and local_bof between [0.1, 0.2]. We can interpret such input distributions as likelihoods of successful exploit execution over a given period of time (month, year, etc.).

Figure 3 shows the result of Monte Carlo simulations of the attack graph model in Figure 2. Each choice of network configuration (block ssh, block rsh, and no change) has a corresponding residual attack graph. In the simulations, we generate inputs randomly according to the given distributions, and combine inputs according to the logical relationships of the particular residual graph.

The simulation output (bottom of Figure 3) is the overall attack graph metric, i.e., the likelihood of database compromise for each network configuration. This shows that blocking rsh traffic provides the best protection, i.e., it is skewed toward lower likelihood of compromise. Simply comparing representative (e.g., mean) values gives an immediate determination of the best choice. Or we could gain further insight by comparing such features as uncertainty spread.

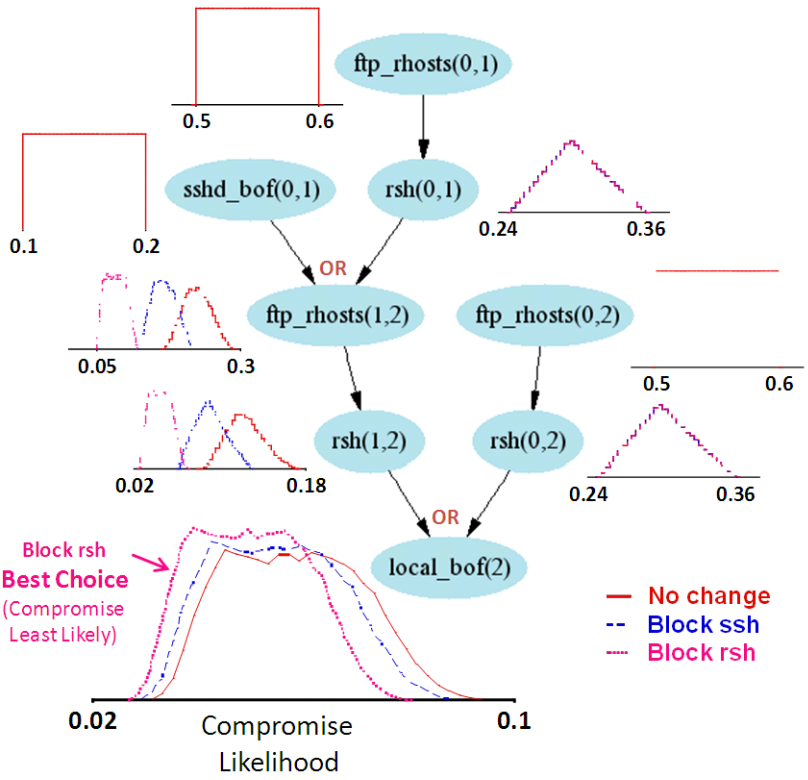


Figure. 3: Attack-graph metrics for each network configuration choice

5. ANALYZING RETURN ON INVESTMENT

Through attack graph simulations, we analyze risks in terms of attack likelihoods. We can extend this risk analysis to include associated network operational costs, attack impact costs, etc. This grounds our analysis in monetary reality, and supports informed decision making about return on security investment. In other words, we can quantify whether expenditures on additional security measures are justified by reduction in expected losses from security breaches.

Figure 4 shows a model for such return-on-investment analysis. We first compute the metric for likelihood of attack compromise (Figure 3). This feeds a cost analysis, which combines compromise likelihood with the projected costs of an actual compromise, along with the costs of implementing network changes to mitigate the risk. In other words, each network configuration has a particular implementation cost, and reduces risk (expected loss from breach) by a certain amount. We seek the configuration that minimizes overall costs.

In this model, the estimated cost of recovering from a database breach (Incident) is $200,000 (standard deviation $20,000). Each network configuration has an associated likelihood of a breach occurring, with a corresponding multiplicative reduction (from the full $200,000) in expected loss. The estimated cost for implementing firewall changes (Firewall) is $1,000 (standard deviation $100). So the question is whether the firewall implementation costs are justified in terms of reduced risk (expected loss), versus simply making no change to the network.

Figure 5 shows the resulting probability distributions (density and cumulative) of overall costs for each network configuration. In terms of overall cost, blocking ssh traffic is now seen to actually cost more than simply making no change to the network. This is because the slight decrease in breach likelihood (expected loss) is outweighed by the cost of implementing the firewall change. On the other hand, blocking rsh traffic reduces breach likelihood more, making its overall costs lower versus leaving the network unchanged. In our risk model (Figure 4), overall cost for each

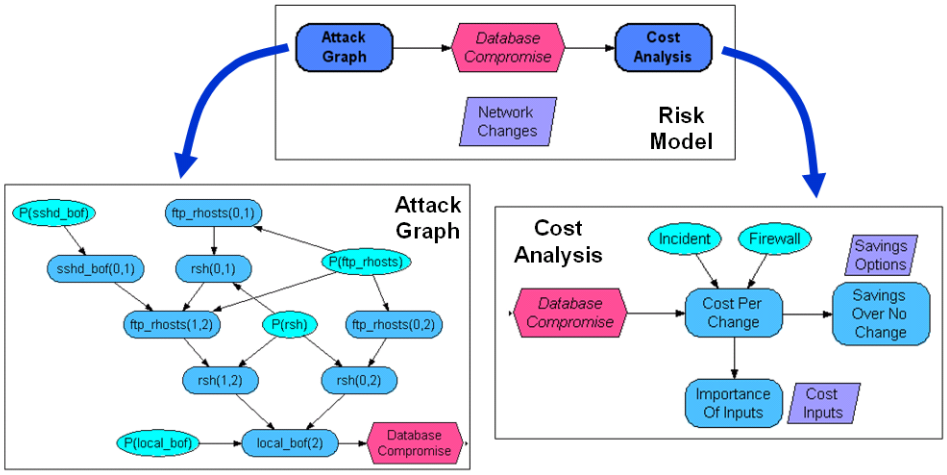


Figure. 4: Security return-on-investment model

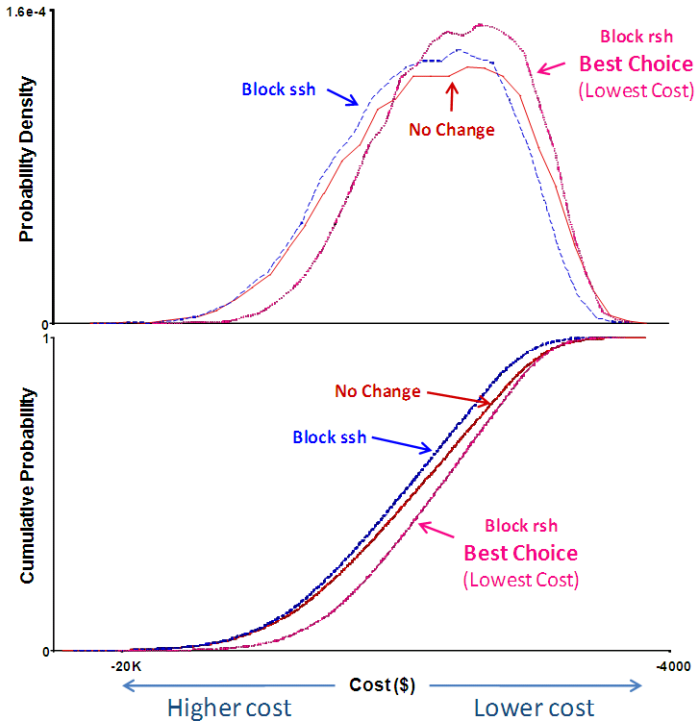


Figure. 5: Cost of each network change based on attack-graph metrics

network configuration (Cost Per Change) is input to Savings Over No Change. This directly compares the costs of blocking rsh and ssh (respectively) to the cost of no change, for each Monte Carlo sample. These model comparisons are shown in Figure 6. We see that blocking rsh traffic can provide almost $2,500 in reduced cost (maximum difference from no change), while blocking ssh traffic can incur almost $1,000 additional cost (minimum difference from no change).

6. QUANTIFYING CONFIDENCE

To quantify the confidence in our analysis, we begin by measuring how strongly the model output is correlated with each model input. This shows the relative importance of each input variable,

i.e., how strongly its uncertainty translates to uncertainty in the output result. In our risk model (Figure 4), this is Importance Of Inputs, which correlates Cost Per Change (Figure 5) with each of the model inputs (indexed by Cost Inputs).

Figure 7 shows the resulting input/output correlations, for each network configuration (block rsh, block ssh, or no change). For all configurations, the output uncertainty is dominated by the likelihood of local_bof, a critical exploit that is included in all attack paths. The next most important input is the cost of a database compromise incident, followed by likelihoods of the rsh and ftp_rhosts exploits. Uncertainties in firewall costs and likelihood of sshd_bof have little impact on uncertainty of results.

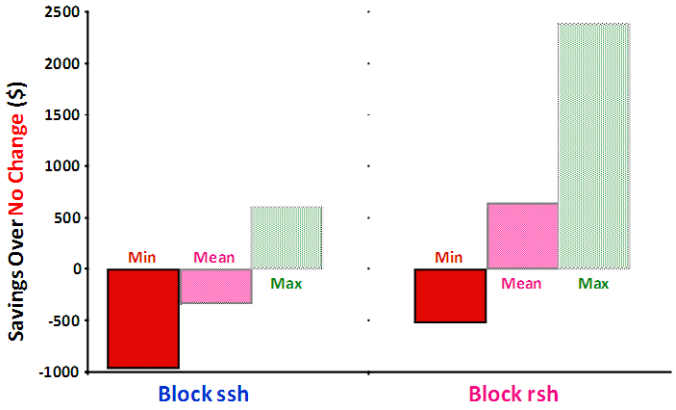


Figure. 6: Comparative savings (versus no change)

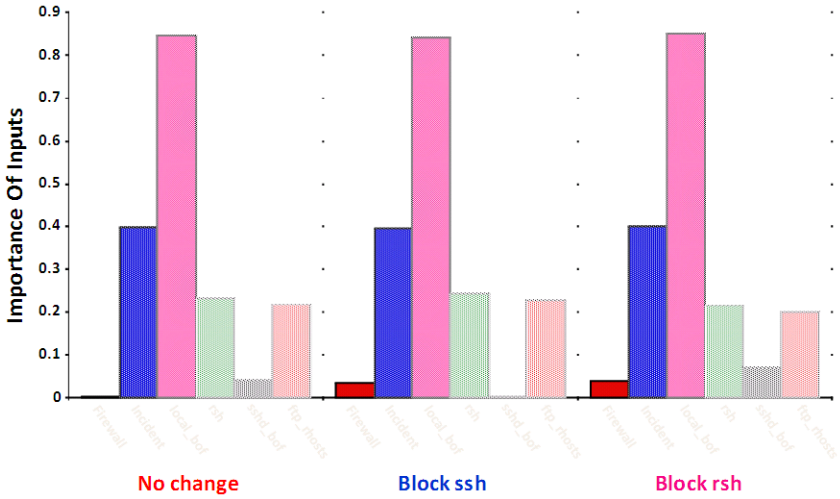


Figure. 7: Relative importance of model inputs

We gain further confidence by testing how model input changes might potentially alter our choice of the best network configuration. In particular, we calculate how overall costs (Cost Per Change) vary as each model input changes. Figure 8 is the variation in Cost Per Change as a function of each input individually, with all other inputs held at their nominal (mean) values. To test robustness, we vary input values well outside their actual ranges in the model.

The upper left of Figure 8 shows how model output depends on the likelihood of the local_bof exploit. If the likelihood of local_bof were less than the breakpoint value of 0.09, then making no change to the network would be the best solution. But in our model, the range of local_bof likelihood is actually between 0.1 and 0.2, for which blocking rsh traffic remains the best solution. The upper right of Figure 8 shows how the model output depends on the cost of a database compromise incident. In the model, the cost per incident is between about $130,000 to $275,000. Except for a small portion of this range (up to about $135,000), blocking rsh traffic remains the best solution.

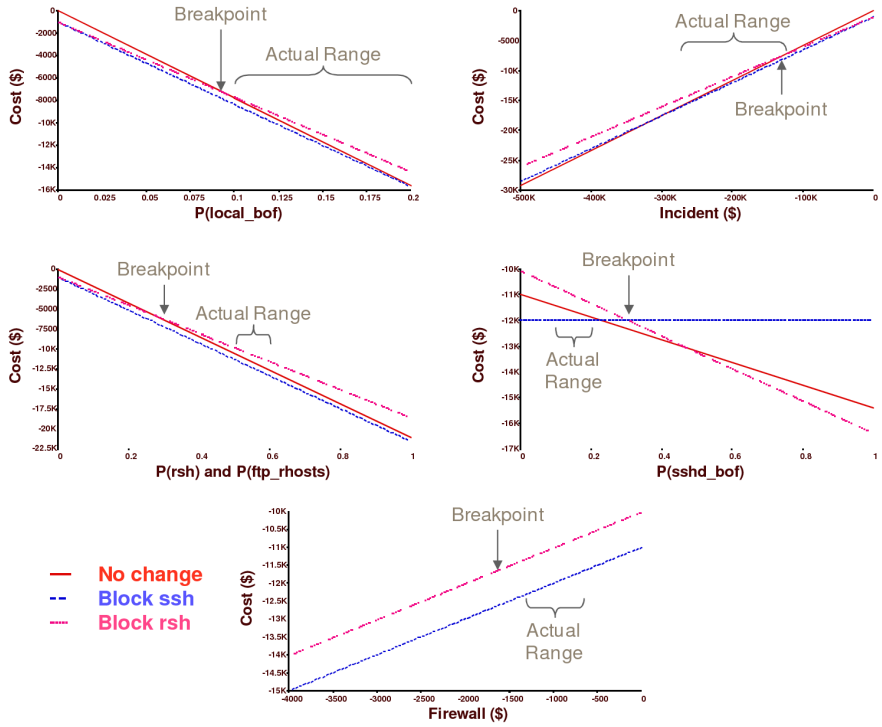


Figure. 8: Cost dependency on individual inputs

Figure 8 also shows the model dependencies on the remaining input variables. For all these variables, blocking rsh traffic is the best solution within the full range of actual model values. Overall, we have a strong measure of confidence that blocking rsh traffic is the best choice for providing return on security investment.

7. SUMMARY

This article introduces a model and a methodology for quantitative analysis of network security risk using attack graphs. This model populates the attack graph with known vulnerabilities and their likelihoods of exploitation. By propagating exploit likelihoods through the attack graph, a metric is computed that quantifies the overall security and risk of a networked system. This metric is applied to study the tradeoffs between cost and benefit for analyzing return on investment. In the future, we plan to build a tool that supports decision making for network planning, security monitoring, and attack prediction/response.

REFERENCES

CVSS "Common Vulnerability Scoring System (CVSS)," Forum of Incident Response and Security Teams (FIRST), *http : //www.first.org/cvss/*.

Jajodia, S., and Noel, S. "Topological vulnerability analysis," In *Cyber Situational Awareness: Issues and Research, Sushil Jajodia, Peng Liu, Vipin Swarup, Cliff Wang, eds.,* Springer, 2009, pages 139-154.

Jaquith, A. 2007 Security Metrics: Replacing Fear, Uncertainty, and Doubt, Addison Wesley, 2007.

Noel, S., and Jajodia, S., "Managing Attack Graph Complexity through Visual Hierarchical Aggregation," In *Proceedings of the ACM CCS Workshop on Visualization and Data Mining for Computer Security,* 2004.

Pamula,J. , Jajodia, S., Ammann, P., and Swarup, V. 2006 "A Weakest-Adversary Security Metric for Network Configuration Security Analysis," In *Proceedings of the 2nd ACM Workshop on Quality of Protection,* ACM Press, 2006.

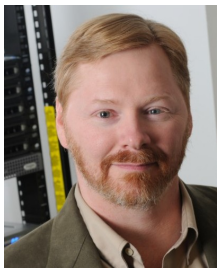
SSE-CMM "The Systems Security Engineering Capability Maturity Model," available at *http : //www.sse − cmm.org/index.html*.

Stoneburner, G., Hayden, C., and Feringa, A. Engineering Principles for Information Technology Security, Technical Report 800-27 (Rev A), National Institute of Standards and Technology, June 2004.

Swanson, M., Bartol, N., Sabato, J., Hash, J., and Graffo, J. Security Metrics Guide for Information Technology Systems, Technical Report 800-55, National Institute of Standards and Technology, July 2003.

SYMANTEC "DeepSight Threat Management System," Symantec Corporation, *https : //tms.symantec.com/*.

Wang, L., Singhal, A., Jajodia, S. "Measuring the Overall Security of Network Configurations using Attack Graphs," In *Proceedings of the 21st IFIP WG 11.3 Working Conference on Data and Applications Security,* Springer-Verlag, 2007.



Steven Noel is Associate Director of the Center for Secure Information Systems at George Mason University. Dr. Noel has led a team of Mason scientists and engineers developing breakthrough patented technology for cyber attack modeling, analysis, and visualization (Cauldron). His research interests include network attack graph modeling, intrusion detection, and visualization for information security. He received his Ph.D. in Computer Science from the University of Louisiana at Lafayette in 2000, with dissertation work in data mining and visualization for information retrieval. He also earned an M.S. in Computer Science from the University of Louisiana at Lafayette (1998) and a B.S. in Electro-Optics from the University of Houston - Clear Lake (1989). From 1990 to 1998, Dr. Noel was a research scientist at the Naval Surface Warfare Center in Dahlgren, Virginia, where he worked in image/video compression, wavelets, neural networks, genetic algorithms, radar signal processing, missile guidance, and astronomy.

Lingyu Wang is an Associate professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Quebec, Canada. He received his Ph.D. degree in Information Technology from George Mason University. His research interests include database security, data privacy, vulnerability analysis, intrusion detection, and security metrics. He holds a M.E. from Shanghai Jiao Tong University and a B.E. from Shen Yang Institute of Aeronautic Engineering in China.

Anoop Singhal is currently a Senior Computer Scientist in the Computer Security Division at NIST. He has several years of research experience at AT&T Labs, Bell Labs and as a faculty member at George Mason University. He is a senior member of the IEEE and he has published more than 30 papers in leading conferences and journals. He received his Ph.D. in Computer Science from Ohio State University, Columbus, Ohio.

Sushil Jajodia is University Professor, BDM International Professor of Information Technology, and the director of Center for Secure Information Systems at the George Mason University, Fairfax, Virginia. He has authored six books, edited thirty four books and conference proceedings, and published more than 375 technical papers in the refereed journals and conference proceedings. He is also a holder of five patents and has several patent applications pending. His h-index is 66. The URL for his web page is http://csis.gmu.edu/jajodia.