It's All About the Bottom Line

Eli Bogart
Cal Pierog
Lori Thomas
Harvey Mudd College
Claremont, CA

Advisor: Hank Krieger

Summary

A brand-new university needs to balance the cost of information technology security measures with the potential cost of attacks on its systems. We model the associated risks and costs, arriving at an equation that measures the total cost of a security configuration and then developing two algorithms that minimize the cost. Both algorithms give a total cost just over half the cost of no security and just over 1.5 times the theoretical minimum cost.

Our model's lack of assumptions about the structure of the university allows the model to be used with any kind of organization, requiring only a set of opportunity costs and statistics about the size of the organization. Our model can even suggest upgrades to existing security systems by changing the costs associated with current security measures.

We consider two extreme cases that bound our solution area and also test the sensitivity of our results by varying the parameters to see the impact on the security configurations chosen by the algorithms. In addition, we analyze equal-cost configurations that lead to different levels of risk.

Introduction

We develop a model to evaluate and optimize choices of security systems for a new university, which could easily be extended to another organization.

- We make assumptions to simplify the problem.
- We present our method for calculating the cost of a combination of security measures.

The UMAP Journal 25 (2) (2004) 115–128.

- We develop a reduced-search-space brute-force algorithm and an iterative algorithm that use the cost formula to find an optimal security configuration.
- We report and analyze the results of these algorithms.
- We discuss the extensibility and flexibility of our approach, with particular attention to how it could be applied to an organization of considerably different priorities, such as a major commercial Internet search engine.
- We discuss improvements and further developments.

Assumptions

A university's computer systems must support activities ranging from word-processing to scientific simulation, from Web-hosting to accounting, for tens of thousands of users on a day-to-day basis. Our client may have as many as 35,000 networked computers [Levine et al. 2003], differing in their operating systems, configurations and primary purposes, and extensively organized into subnetworks and departments. This scope and complexity, combined with the ever-increasing number and diversity of threats to information security, and the wide variety of countermeasures available to combat those threats, make precise optimization of the school's information technology security a challenge. To simplify this process, we make several assumptions:

- **All security measures are applied universally.** We assume that a single, uniform package of defensive measures and policies is implemented for every computer on campus (although our model supports the ability to individually analyze subsystems). This assumption allows us to disregard any security-related interactions between differently-protected subsystems.
- **At most one security measure of each type.** Two security measures designed to protect against the same category of threat are highly likely to have overlapping capabilities. If we have two spam filtering programs, we would expect spam email detected by one program to be be flagged by the other, so that it is unlikely that operating both is profitable. On the other hand, this sort of redundancy could be desirable as a protection against system failure.
- **No redundancy or synergy among security measures of different types.** The presence or absence of a security measure or policy of one type cannot impact the effectiveness of a security measure or policy of any other type. In practice, system administrators could use the information provided by a network intrusion detection system to adjust the configuration of a firewall, improving its performance; but in the absence of any relevant data, ignoring these effects seems to be a relatively benign simplification of the problem.
- **Five-year time frame.** The procurement and installation of a security measure is a one-time cost, while the associated maintenance costs and security

benefits accumulate over time. Given the rapidly changing nature of security threats and computer technologies, it is reasonable to compare the net costs of security measures over five years.

- **Costs of security measures are independent.** The prices of different security products are independent, and we neglect any financial effects of installing multiple products at one time—simply put, there is no bulk discount. In this respect, our model is overly pessimistic; apart from any discounts, bulk installation would also be faster and cheaper. However, this assumption allows us our model to encompass systems with pre-existing components.

Cost Equation

Any analysis of risk requires a way to compare two security configurations and choose the better one. Our model accomplishes this by measuring the dollar amount that a security system will save over the next five years. This dollar amount has two components: attack costs and sunk costs. *Attack costs* accrue from information attack and the resulting litigation, data loss, loss of consumer confidence, and so on. *Sunk cost* is the price of implementing the security system plus the cost of maintaining it over five years. Additionally, the sunk cost includes the dollar estimate of the gain or loss in productivity caused by the security system. The sum of these two costs is total cost.

Attack Cost

To measure the cost of an attack, we could lump all costs together and assume that all security measures reduce that total cost. To do so would be overly simplistic, since three security measures that all reduce the same aspect of cost are not necessarily as effective as three that reduce different components.

We break the total risk into three components: information confidentiality, data integrity, and system availability [Levine et al. 2003]. The relative importance of the indices for confidentiality, integrity, and availability (C, I, and A) for a given company will drive its choices in security measures.

Table 1 of the problem statement allows us to break down the baseline cost (of no security whatever) into the three risk categories: $BaseCostC$ = \$ 4.3 million, $BaseCostI$ = \$ 3.585 million, and $BaseCostA$ = \$ 1.045 million, corresponding to confidentiality, integrity, and availability.

Each security device or policy affects C, I, and A in terms of percentage changes dC, dI, and dA from the initial values of 1. Positive changes reflect higher levels of confidentiality, integrity, and availability, so costs from attacks should decrease as the indices increase. We offer the following equation for the cost of an attack given n categories of security features and policies:

$$AttackCost = years \times \left(\frac{BaseCostC}{\prod_{i=1}^{i=n}(1+dC_i)} + \frac{BaseCostI}{\prod_{i=1}^{i=n}(1+dI_i)} + \frac{BaseCostA}{\prod_{i=1}^{i=n}(1+dA_i)} \right)$$

With no security features, the indicated products are all 1 and we get the baseline attack costs.

Sunk Cost

There are two aspects to sunk costs: money spent on security measures and change in productivity. The money spent includes the one-time cost of purchasing or implementing the measure or policy, maintaining it, and training individuals in its use. This amount can depend on the number of users, computers, and IT staff trained to work with the measure. We divide IT staff into two categories, help-desk workers and system administrators. The cost of training IT staff for each product is assigned to the appropriate category of staff. This provides a bit more realism for the model, as help-desk workers in general do not require as much training as system administrators.

The second aspect of the sunk cost is the change in productivity, P, which works much like the C, I, and A indexes used to determine attack costs. Increases in productivity should lead to decreases in the sunk costs, since increased productivity lessens the cost of the security feature, and the increase in productivity depends only on the existence of the security features, not on attacks. To calculate the change in productivity from the baseline, we subtract the base productivity value, getting

$$\begin{aligned} SunkCost = & \sum_{i=1}^{n}(procureCost + maintCost + trainCost) \\ & + years \times \left(\frac{BaseValueP}{\prod_{i=1}^{n}(1+dP_i)} - BaseValueP \right) \end{aligned}$$

The $BaseValueP$ is the product of the number of users and the productivity per user, obtained by estimating the number of hours per year that the average user spends using the university's IT services and the replacement cost of those services (as estimated by our team).We arrived at $BaseValueP = \$12$ million. (Later, we analyze the sensitivity of the model to this value.)

The costs to purchase or implement and maintain security depends on the numbers of computers, users, and IT staff. **Table 1** lists the values that we chose to simulate the university.

Table 1.

Fixed input parameters for the model.

Variable	Value	Variable	Value
Computers	17,900	$BaseCostC$	\$4.3 million
System administrators	16	$BaseCostI$	\$3.585 million
Help-desk staff	55	$BaseCostA$	\$1.045 million
Users	17,000	Productivity per user	365
Years	5		

Total Cost

Combining our two equations, we get

$$\begin{aligned} TotalCost = & \sum_{i=1}^{n} (procureCost + maintCost + trainCost) \\ & + years \times \left(\frac{BaseCostC}{\prod_{i=1}^{i=n}(1+dC_i)} + \frac{BaseCostI}{\prod_{i=1}^{i=n}(1+dI_i)} + \frac{BaseCostA}{\prod_{i=1}^{i=n}(1+dA_i)} \right. \\ & \left. + \frac{BaseValueP}{\prod_{i=1}^{n}(1+dP_i)} - BaseValueP \right) \end{aligned}$$

Input

The C, I, and A multipliers and prices for each security measure are obtained from Enclosures A and B of the problem statement. We reduce the values for P by a factor of 10 to reflect reasonable changes in productivity due to any single product. We also ensure that each security category has a null option, with a cost of zero and values for C, I, A and P of 1. We create two categories of IT staff and split training costs for system administrators between the two categories.

Models and Approaches

We explore several approaches to optimizing the university's security configuration using the cost formula above.

Brute-Force Computation

Calculating the net cost of every combination of security features allowed by our assumptions and picking the best would be guaranteed to find the best security configuration within the parameters of our model. However, evaluation of the cost formula would be computationally intensive. If j_i security features (including the null feature) are available in the ith category, one feature can be chosen from each category in $j_1 \times j_2 \times \cdots \times j_n$ distinct ways. Once a set of features has been chosen, calculating the resulting effects on confidentiality, integrity, availability, and productivity requires $4(n-1)$ multiplications. Comparison of all possible security configurations thus requires

$$4(n-1)\prod_{i=1}^{n} j_n$$

multiplications. For the security features available to the university, this would be 4.77×10^{12} multiplications.

While many of these multiplications are repeated many times, allowing a good algorithm to reduce the total number of calculations drastically, the brute-force approach is nonetheless too numerically intensive to produce results within a reasonable time frame.

Refined Brute Force

Reducing the number of security features under consideration could substantially reduce the number of calculations required for a brute-force approach. To do this, we calculate the net cost of each security feature in isolation. If a feature, installed alone, results in more costs due to procurement, maintenance, and lost productivity than savings due to increased security, we assume that the feature will not become profitable as part of the optimal security configuration.

This assumption is plausible but not guaranteed. A security policy such as allowing wireless networking, which allows a great increase in productivity at the expense of confidentiality, integrity and availability, might become profitable in combination with a (hypothetical) inexpensive combination of security measures that increase the C, I, and A indexes more than enough to compensate. But most of the security measures that we are considering have small effects on productivity; their net cost is determined primarily by their installation and maintenance costs and their security benefits, which are proportional to C, I, and A. Such measures, if unprofitable on their own, are almost certain to become even less profitable in combination with other measures that reduce C, I, and A. We decided that such security features are safe to neglect. This eliminates 34 of 83 technological measures and settles 4 of the policy choices studied—many of which would have cost well over $1 million over five years—reducing the number of necessary multiplications to 600,000.

Cherry-Picking

As an alternative to reducing the size of the search space, we created an iterative algorithm to construct a security configuration. Starting from an undefended network, we repeatedly add the most profitable security feature available until one technological measure (possibly null) from each category had been acquired and all policy choices had been made. In addition to producing an effective overall security system, this process also provides an outline for acquiring security features piecewise, as could be required on a limited budget.

Results

Model Results

We ran both the Refined Brute Force and Cherry-Picking algorithms on the data set provided in the problem statement. The resulting total costs can be compared to each other but are not useful without a frame of reference. To provide that sort of framework, we ran both algorithms again, this time minimizing only one component of the cost, either attack cost or sunk cost. In both cases, both algorithms produced exactly the same security configuration and total cost, thereby giving lower bounds on the attack cost and the sunk cost for with the data set. We illustrate tradeoffs between attack costs and sunk costs in **Figure 1**. Lines of slope -1 consist of points with the same total cost. **Table 2** shows the security configurations for points A, B, C, and D.

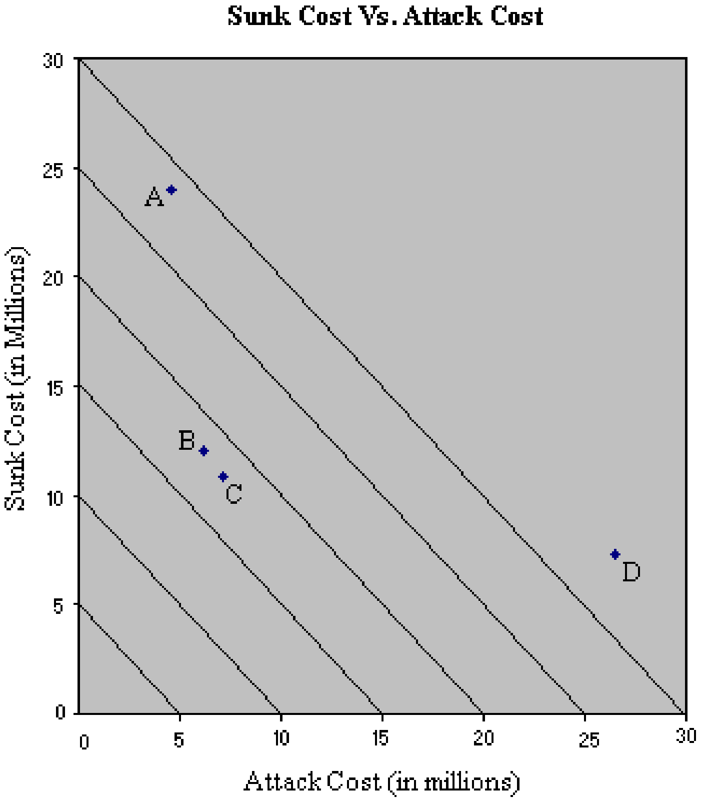

Figure 1. Sunk costs vs. attack costs. Diagonal lines are made up of points with equal cost. Point A which minimizes the cost of an attack regardless of the sunk cost. Point B is chosen by the Cherry-Picking algorithm. Point C is chosen by the Refined Brute Force algorithm. Point D minimizes the sunk cost, regardless of the cost of an attack.

No configuration can have an attack cost lower than that of point A nor sunk costs lower than that of point D. The vertical line through A and the horizontal line through D intersect at the theoretical lowest total cost—a combination of security features that costs almost nothing and reduces the cost of attack to almost nothing.

Table 2.

Products and policies for points A, B, C, and D in **Figure 1**.

category	A	B	C	D
Host Firewall	Intelli-Scan	Lava	Barrior	none
Net Firewall	EnterpriseLava	EnterpriseLava	none	none
Host Anti-Virus	BugKiller	Anti-V	Anti-V	none
Net Anti-Virus	Enterprise Stopper	System Splatter	System Splatter	none
Net IDS	Network Eye	Watcher	Watcher	none
Net Spam Filter	Spam Stopper	Spam Stopper	SpamStopper	none
Net Vulnerability Scan	none	none	none	none
Data Redundancy	none	none	none	none
Service Redundancy	none	none	none	none
Password Policy	Strong	Strong	Strong	Strong
Security Audit?	Formal	Formal	Formal	none
Wireless?	none	none	none	none
Removable Media?	none	none	none	none
Personal Use?	Restricted	Restricted	Restricted	Restricted
User Training?	Required	Required	Required	none
IT Staff Training?	Required	none	none	none

Deviation from Expected Attack Rates

In **Figure 1**, points B and C fall almost on the same line, so their total costs are almost equal. We would like to be able to distinguish between two points with similar total costs but different divisions of this total between sunk and attack costs. One way to do so is to consider what happens if the rate of successful attacks is different than expected.

- Suppose that the government cracks down on computer crime and only half as many attackers manage to break into the university networks; costs due to attacks will be half as much as before, benefitting those who spent more on attack costs than sunk costs.

- On the other hand, suppose that the number of attackers is more than anticipated. In this case, the amount spent on attacks is much larger than expected, so those who spent more on attack costs than sunk costs end up spending more than they had planned.

Figure 2 illustrates this point. A subscript 1 corresponds to cutting the rate of attack in half, a 2 to expected attack rates, and a 3 to doubling the attack rate.

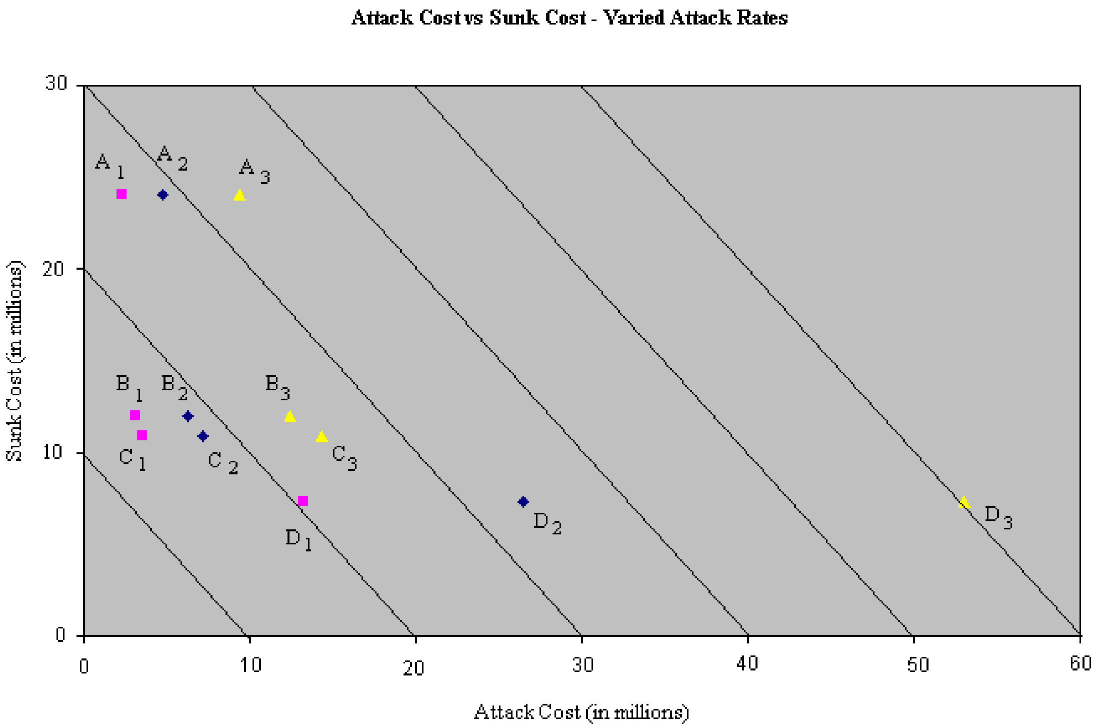

Figure 2. Points A, B, C and D correspond to minimizing attack costs, the total cost using the Cherry-Picking algorithm, the total cost using the Refined Brute Force algorithm and the sunk costs, respectively. Subscript 1 corresponds to half as much cost from attacks as expected, 2 to the expected cost from attacks, and 3 to twice the cost expected.

Among the points with halved attack rate, C_1 has the lowest total cost. However, when the attack rate is increased to twice the expected value, B_3 overtakes C_3, indicating that although the Brute Force algorithm (C) is best for low attack rates, the Cherry Picking algorithm (B) surpasses it when the attack rate increases.

Variation of Parameters

To test the robustness of our results, we individually varied $BaseCostC$, $BaseCostI$, $BaseCostA$ and $BaseValueP$ by a factor of 2 and by a factor of 1/2. These variations had a small effect on the security configurations chosen by the Refined Brute Force algorithm. The choice of host-based firewall varied the most in response to changes in values, with five of the nine configurations choosing Barrior, one choosing Watertight, and three choosing Lava. The next most varied was the choice of network-based firewall, with six choosing none and three choosing Enterprise Lava. Two of the configurations had single choices that differed from the other eight. These results suggest that our model is only somewhat sensitive to variations in these parameters with the restricted data set used by the Refined Brute Force algorithm.

We varied the same parameters by the same factors using the Cherry-Picking algorithm. Though the actual security configuration changed less than with the Refined Brute Force algorithm, the order in which each security feature was chosen varied from the norm in all but one case. Even when the order differed, it usually did not do so until the seventh or eighth purchase out of 16.

The two algorithms responded similarly to variation in the parameters except in the firewall categories, where there were consistent differences. The Refined Brute Force algorithm chose Barrior and no network-based firewall, where the Cherry-Picking algorithm chose Lava and Enterprise Lava in 10 instances out of 18. Thus, the two algorithms give generally consistent results and the inconsistencies are systematic to some degree.

Extensibility

The model makes no assumptions about the kind of organization under analysis and thus is highly adaptable and can be used to analyze any company or organization's computing resources. Our model simply requires three pieces of data to do its computations:

- A list of the currently available technologies and their expected impacts on confidentiality, integrity, availability, and productivity.
- The number of computers, users, and system administrators that the system is expected to support.
- How much the organization currently spends on confidentiality, integrity, and availability, as well as the estimated value of each user's productivity.

Since these parameters are not specific to any type of company or organization, any organization can be modeled, from universities to on-line banking to Internet search engines (see below).

Subsystems

What's more, our model can also analyze the security tradeoff of applying security features to only a subset of a larger system, such as buying a firewall for only one subnet on a university campus.

Using this method of breaking down a larger system into smaller subsystems, we can more effectively secure each subsystem, because we can choose a completely custom set of security measures for each subsystem individually. This means that each part of the organization's computing facilities can implement only the security that is most effective rather than following global security policies that only slightly benefit the particular subsystem, thereby not only increasing the overall security of the system but also substantially decreasing the cost of the security system.

Our model can also be utilized if the organization later decides to merge two subsystems or divide an existing system into several parts, even after the initial security system is in place.

New Technology

The constantly changing face of security makes periodic updating essential. Fortunately, our model allows analysis of security systems already in place, so it can re-evaluate an existing security system to see if it can be updated. Systems already in place receive an implementation cost of zero. Due to their presumed age, their effective confidentially, integrity, and availability must be recalibrated in light of emerging technologies.

Once we enter the old systems into the database, we use our analysis to determine if they are still financially viable. If maintenance costs exceed estimated utility, the systems' use should be discontinued. The analysis will additionally suggest upgrades or additional security systems that would be profitable to install.

Implementation Costs

However, projected income is not the only concern. Companies must also consider their limited cash on hand. For example, a security system that would save money over the next five years might be infeasible because the company does not currently possess enough financial reserves to cover the initial costs.

The Cherry-Picking Algorithm deftly addresses this concern, picking the most profitable systems first before others. In this manner, the company can most effectively allocate its limited financial reserves to the systems that will effect the largest profit increases. They need only start picking at the top of the list and working their way down until their security budget is expended.

Web Search Engines

The priorities of web search engines are very different from those of universities. Therefore a search engine's opportunity costs (as presented in the problem statement's **Table 1**) are quite differently distributed.

Consumer Confidence

Consumer confidence is of paramount importance to Web search engines. Nearly all search engines rely on advertising as their primary source of income. However, publicists are interested in advertising only with popular search engines, where the most users will see their ad. The financial situation of a search engine is intimately tied to its consumer confidence (usage), so it follows that a search engine's opportunity costs are primarily proportional to the consumer confidence category.

Service Reconstruction

Service reconstruction is closely related to consumer confidence. After all, a search engine can be viewed as a company that provides a solitary service: searching the Web. If a search engine is unable to provide this service to its users because of a security breach, it will lose consumer confidence: The longer the outage, the worse its effects. Thus, service reconstruction is another consumer confidence category in which search engines are interested.

Direct Revenue Loss

Although especially vulnerable to attack that damages consumer confidence, such as a denial-of-service attack, search engines are not susceptible to financial attacks such as a salami attack. The search engines' relative immunity to such attacks stems from the lack of financial transactions that involve the Website. Simply put, search engines do not have any sensitive data such as credit-card and bank-account numbers. This means that there is really no way for an attacker to steal money from the search engine, rendering the direct revenue loss category rather inconsequential.

Proprietary Data Loss

Not only do search engines not store sensitive financial data, they do not store any private data at all. The purpose of a search engine is to allow people to quickly find data that is available to everyone. Therefore, all the information cached by a search engine is freely available to anyone with an Internet connection, meaning that search engines have very little proprietary data to lose. Attackers are therefore unlikely to cause proprietary data loss.

Data Reconstruction

This complete lack of proprietary data also helps the search engines with regard to data reconstruction. Since all the information housed by a search engine is freely available, attackers are unlikely to attempt to corrupt or sabotage the data. Moreover, if any data is corrupted, it can be restored by downloading a fresh copy from the Internet. From a search engine's point of view, the entire Internet is a backup copy.

Litigation

The majority of a search engine's users pay no fee, and so have little grounds for litigation since the search engine has no legal obligations to them.

More problematic from a legal perspective are the advertisers. The search engine is contracted by the advertiser to display an ad in a certain manner. If the search engine is unable to do so because of the nefarious work of an attacker, then the advertisers have grounds for lawsuit. However, the situations that would earn the ire of advertisers are exactly the same ones that would lower consumer confidence, that is, the site going offline. Therefore, while some attention must be paid to legal ramifications, the defensive measures involved would be the same as the consumer confidence category.

Directions of Further Work

The first priority of future work is to remove our assumption that one set of security features will be applied to every computer system within the organization. At present, entire categories of features (data and system redundancy) are excluded from configuration because they are far too expensive to implement campus-wide. In fact, those features are not intended to be applied on such a large scale but only to critical systems and services. Our model is entirely capable of handling the implementation of different security features on subsets of an organization's computer systems; doing so would require only a breakdown of the university's computing needs into subnetworks, each with its own productivity and costs associated with confidentiality, integrity, and availability.

Another valuable refinement of this model would be in the method for assessing $BaseCostC$, $BaseCostI$, $BaseCostA$ and $BaseValueP$ for different components of the university or other organization. Hsiao [1979] assigns to each component of the IT network not only a value but also a probability of attack; the product of the two gives the expected cost of attack for that component. This cost could be broken down into costs due to C, I, and A, allowing us to consider each feature individually and its ideal security configuration. We could then go a step further and figure out how to group different components to share security features or policies in a cost-effective manner.

The value of our results would also be enhanced by relaxing our assumptions regarding redundancy and synergy among security features. Two security features could interact, with effects difficult to judge from information on the individual effect of each. Quantitative estimates of such interactions could be obtained in the same way as the data on individual security features, by polling of industry experts or experienced system administrators.

To highlight a particularly important synergy effect, a more-detailed model would acknowledge that as an organization's systems become more resistant to attack, not only will fewer attacks succeed but the organization will present a less-appealing target and fewer attacks will be launched. The provided estimates of security effects may take this into account for individual security features, but a successful combination of many security measures will have an even greater effect.

We have ignored the variation in the expert estimates of the effects of different security measures and policies. A more-detailed analysis could easily produce estimates of the uncertainty in our predictions of the savings resulting from each security configuration we propose. An improved version of our model would give priority to a security feature whose effects are known with reasonable certainty over a feature which is expected on average to be more beneficial but in which we can have no confidence.

Finally, to make this model more effective, it is essential to expand the number of security measures and policies considered. The nine types of technological defensive measures and seven policy defenses considered here hardly represent the entire spectrum of approaches. For example, no consideration has been given to physically protecting the university's hardware, a legitimate information technology concern with definite potential effects on the C, I, and A factors considered in the model. Perhaps worse, currently we consider only one nonspecific "user training" policy. Some form of user training is the best defense against "social engineering" attacks, which are already a major unrealized vulnerability and likely to become only more common in the future. Research into available measures to address physical and other security factors, and a closer examination of user training possibilities, would make our model potentially much more powerful.

References

Greenberg, Eric. 2003. *Mission-Critical Security Planner: When Hackers Won't Take No for an Answer*. New York: Wiley.

Honeynet Project. 2003. Know your enemy: Honeynets—What a Honeynet is, its value, how it works, and risk/issues involved. `http://project.honeynet.org/papers/honeynet/index.html` . Last modified 12 November 2003.

Hsiao, David K. 1979. *Computer Security*. New York: Academic Press.

Levine, John, Richard LaBella, Henry Owen, Didier Contis, and Brian Culver. 2003. The use of honeynets to detect exploited systems across large enterprise networks. *Proceedings of the 2003 IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, June 2003. `http://www.tracking-hackers.com/papers/gatech-honeynet.pdf` .

Spitzner, Lance. 2003. Honeypots: Simple, cost-effective detection. `http://www.securityfocus.com/infocus/1690` . Last updated 30 April 2003.