As a software engineer or client, how much of your budget should you spend on software security mitigation for the applications and networks on which you depend? The authors introduce a novel way to optimize a combination of security countermeasures under fixed resources.
reports on the project) has a staff of approximately 1,000 employees and provides technical and managerial services to large organizations. Mitigating security concerns in their products and services is a central component of Acme’s business mission.

**Misuse cases**
The Square process is a comprehensive approach to eliciting, categorizing, and prioritizing the security requirements of a system under design. Here, we focus on a subset of this process, in which we identify threats to the client’s system with misuse cases and quantify their impact with standard risk-assessment methodologies. We then apply our IP optimization methodology in terms of choosing the optimal combination of actions that engineers can take in securing the system against these misuse cases under a fixed budget.

Misuse (or abuse) cases—as explained by Paco Hope, Gary McGraw, and Annie Antón—are scenarios that describe how an attacker or malicious user would attempt to abuse a system. In studying them, software developers can gain insight about how to design systems to counteract prospective attacks. Successfully preventing a misuse case requires implementing a combination of architectural recommendations (ARs) and policy recommendations (PRs). If we had a misuse case in which a user gains access to a system using spoofed identities, for example, an AR would be to set up firewalls between servers and workstations, and a PR would be to patch the firewall software weekly.

Obviously, not all misuse cases are this simple, and preventing a more complex one could require the implementation of multiple ARs and PRs. Alternatively, a single AR or PR could be part of the solution for multiple misuse cases: in the previous example, the suggested PR could also mitigate the effect of another misuse case (“main server infected with a virus or worm”). Hence, we can have a many-to-many relationship among misuse cases and recommendations, both policy and architectural.

**The costs of mitigation**
To prioritize recommendation implementation, the software engineer or client must assign a cost to each one—say, in dollars per year. To implement an AR, for instance, the costs might include initial hardware or software expenses and their corresponding implementation and maintenance requirements.

In the previous firewall example, the AR costs include the physical firewall unit’s price as a fixed hardware cost in addition to the number of hours an employee is expected to spend setting up and monitoring the firewall (multiplied by that employee’s hourly rate). The number of person-hours spent (again, multiplied by the hourly rate) learning how to patch the firewall and the expense of patching thereafter are included in the PR costs.

We can sum all these costs to reflect the total yearly costs per recommendation, but this sum is misleading because of the many-to-many relationship between recommendations and misuse cases: the more misuse cases a recommendation addresses, the lower its marginal cost of implementation becomes. Our optimization methodology reduces the complication involved in deciding which misuse cases to address when budgetary restrictions preclude addressing them all. In our sample application, Acme decided that misuse case resolution was a binary variable, meaning that each misuse case would either be completely resolved or completely unresolved. This implied that for Acme, even if it implemented four out of five of a misuse case’s recommendations, the vulnerability would still be considered unresolved. We can estimate an expected total yearly loss for these unresolved misuse cases by calculating them on a per-incident basis and multiplying by an estimated yearly frequency. This estimation also considers other opportunity costs, such as loss of reputation. Denis Verdon and Gary McGraw highlighted several common methods for calculating such losses.

In the Acme case study, the team discovered 12 misuse cases in their system, an average of roughly 10 recommendations per misuse case, and a single budget constraint (dollar cost), but our optimization approach readily extends to handle additional constraint types (such as time or the availability of staff with particular skills). The team then developed a yearly cost per recommendation, yearly cost per resolution of misuse cases, and yearly cost per unresolved misuse case. Acme wasn’t sure how much money it could budget for threat mitigation, so we solved the optimization model repeatedly for different budget levels between US$5,000 and $120,000. However, for any given budget, we could solve the model to optimality with Excel’s built-in “Solver” in a second or two.

**Results**
Our results were instructive in several dimensions. Figure 1 shows optimal spending on remediation as a function of budget, indicating that it’s never optimal for Acme to spend between $10,000 and $40,000 on security countermeasures. The number of misuse cases resolved doesn’t improve with more spending in this range because Acme wouldn’t consider a misuse case to be re-
solved unless it implemented every recommendation. With very limited resources, Acme could use the first $10,000 to address the low-hanging fruit, such as preventing brute-force password cracking attacks. Not until the budget reaches $40,000 does it make sense to address other more difficult misuse cases, such as buffer overflows, accidental deletion of configuration files, or SQL injection attacks.

Another interesting result was that AR or PR implementation wasn’t a monotonic function of budget size—that is, recommendations didn’t have a simple decision rule (“if the budget available exceeds X dollars, then implement additional recommendation Y”). The optimal strategy called for implementing recommendations to prevent “unauthorized access to the main server” if the budget were between $60,000 and $70,000 or greater than $100,000, but not if it were between $70,000 and $100,000. In retrospect, the reasoning is clear: when the budget is less than $60,000, it isn’t possible to implement all the required ARs and PRs necessary to address a vulnerability. If the budget grows beyond $70,000, then it’s possible to address an even more pressing misuse case, such as input validation attacks but not while simultane-

ously addressing “unauthorized access to the main server” unless the budget exceeds $100,000. Although this is sensible in retrospect, these strong interaction effects are hard to anticipate ex ante or to work out intuitively. Figure 2 shows exactly how the number of resolved misuse cases grows nonlinearly with budget size.

The method of resolving misuse cases by implementing discrete recommendations makes it easy to understand which specific actions pay dividends by addressing multiple misuse cases. Without this approach, such instances of “killing two birds with one stone” would tend to be overlooked, leading to misinformed judgments about the actual difficulty of addressing various combinations of misuse cases.

Naturally, this optimization approach has its limitations. First, as with any quantitative analysis, the results are only as good as parameter accuracy. Although we could calculate the cost of ARs and PRs reasonably precisely, misuse case losses tend to be expert judgments, making sensitivity analysis important. The software engineering discipline as a whole still needs formal tools to assess the accuracy of human estimations: even in so-called “hard” sciences, experts make consistent, serious errors in judgment and tend to be optimistic about their level of accuracy.11 This weakness could be mitigated by having multiple risk experts perform their analyses independently. Because human judgment is prone to fallibility, we can leverage to some extent the wisdom of crowds when estimating the costs of misuse cases.12 The second limitation of this approach is the fact that outlining the entire set of misuse cases isn’t trivial. Given the relatively small size of Acme’s application, this wasn’t a problem for our case study, but larger applications can have many ways in which a system could be infiltrated; there must be an effective method to select and prioritize a set of them.1,13

Note that none of these limitations is specific to our optimization-based approach—they’re equally pertinent for any quantitative approach to making decisions about investments in security countermeasures. Furthermore, the application of this methodology to a medium-sized company such as Acme proved to be quite fruitful, so we believe our methodology holds promise as a tool for further applications deployed in such environments. The Square method has undergone additional case studies, tool development, and development of educational materials. As we gain more experience with it, we expect additional refinement to take place.

References
2. D. Kahneman and A. Tversky, eds., Choices, Values and...


Jonathan P. Caulkins is Professor of Operations Research and Public Policy at Carnegie Mellon University’s Qatar campus in Doha and its Heinz School of Public Policy. His research interests include mathematical modeling and systems analysis with a particular focus on social policy problems pertaining to drugs, crime, violence, and prevention. Caulkins has a PhD in operations research from MIT. Contact him at caulkins@cmu.edu.

Eric Hough is a software engineer at the Space and Naval Warfare Systems Center San Diego, where he develops tools to improve network security for the US Navy. Hough has an MS in information security technology and management from Carnegie Mellon University. Contact him at hough@spawar.navy.mil.

Nancy R. Mead is a senior member of the technical staff in the Networked Systems Survivability Program at the Software Engineering Institute (SEI). Her research interests include information security, software requirements engineering, and software architectures. Mead has a PhD in mathematics from the Polytechnic Institute of New York. She is a fellow of the IEEE and the IEEE Computer Society and a member of the ACM. Contact her at nrm@sei.cmu.edu.

Hassan Osman is an information security advisor with Ernst & Young’s Security & Technology Solutions practice, where he helps clients assess, evaluate, and implement solutions to protect their IT infrastructure and business processes. He recently wrote a book entitled *Securing Your Information in an Insecure World* (BookSurge Publishing, 2007). He has an MS in information security policy and management from Carnegie Mellon University. Contact him at hassan.osman@ey.com.