Editors: Vijay Varadharajan, vijay@ics.mq.edu.au
Fred Cohen, fc@all.net

Measuring Security

Sal Stolfo and Steven M. Bellovin
Columbia University

David Evans
University of Virginia

The field of computer and communications security begs for a foundational science to guide system design and to reveal the safety, security, and possible fragility of the complex systems we depend on today. To achieve this goal, we must devise suitable metrics for objectively comparing and evaluating the security of system designs and organizations.

For example, managers must have the tools they need to answer these fundamental questions: Is my organization secure? Are the personnel sufficiently educated and trained to minimize the risks to the organization? Is my organization complying with regulations on managing and safeguarding sensitive data? How do I measure the security risk of a new technology or service provided to our customers?

These and many other related questions are often answered qualitatively and retroactively, if at all. Rarely do we have hard measurements to objectively and scientifically answer them in ways that allow meaningful predictions. It is especially important to answer these questions longitudinally to understand whether a new technology or policy has improved an organization's security posture. With formal metrics, we could automate much of a system's lifecycle maintenance by measuring the system's successes and isolating its shortcomings. These desiderata all depend on developing security as a science.

Interpretations of Science

The meaning of *science* has at least three senses. The first is the weak sense: science as the systematization and generalization of knowledge. In recent years, much research has provided a strong understanding of a particular vulnerability or the security issues involved in designing a given system. However, we've seen little success in collecting the knowledge from this research into a general, systematic framework. That is, although there are a few broad classes of vulnerabilities—buffer overflows, injection attacks, and so on—details frequently differ enough that developers have little useful guidance other than the usual high-level guidance: check array bounds, sanitize inputs, and so on. Computer security would benefit from more research on this problem. One goal is to establish a common framework for classes of defenses, categorized according to the policies they can enforce and the classes of attacks those policies can thwart.

The second is the strong sense: science as a way to develop universal laws we can use to make strong, quantitative predictions. Although such laws would be wonderful, no clear consensus exists as to whether they even exist for system security. Nevertheless, efforts to find them would be useful, even if those efforts lead to a clearer understanding of why they don't exist.

The third is the methodological sense: science as a way to conduct research by forming hypotheses and carrying out experiments. For certain areas of computer security, experiments seem useful, and the community will benefit from better experimental infrastructure, datasets, and methods. For other areas, it seems difficult to do meaningful experiments without developing a way to model a sophisticated, creative adversary. From this perspective, there are several promising directions for research and engineering leading to a stronger scientific basis for computer security. Here, we focus on the one aspect that's both essential to becoming a legitimate science and the most lacking in our current understanding of computer security: metrics.

Cybersecurity Metrics

Despite Lord Kelvin's oft-repeated maxim that you can't really claim to understand something until you can measure it, few meaningful tools exist for measuring computing-system security. Without such tools, measuring progress scientifically and making practical decisions are difficult. Metrics can be either analytical or experimental; we need to develop and explore the value of both types for computer security.

The fundamental challenge for any computer security metric is that metrics inherently require assumptions and abstractions. A metric aims to take a complex system and produce a scalar value that describes some important property of that system. For computer security, any assumptions embedded in a metric are potential vulnerabilities. In some sense, what we really need is a *metametric*—a way to measure the risks associated with all the assumptions. One interesting direction is to devise new design approaches that focus on designing for measurable security properties. Another is to devise challenge problems to investigate metrics.

Relative Metrics

Although constructing an absolute security metric for a given system might be impossible, relative metrics might be feasible. For example, a metric might be able to measure how much more secure a system f' is than a system f, where f' is some transformation of the original system.

For security, we could note that a system with m independent defenses is $f(m)$ times as secure. Conversely, a system that accepts m times as many different inputs will be $g(m)$ less secure. In a different vein, we can scale a module's relative security value by a function that depends on the module's development and testing environment.

For each proposed type of metric, we can consider both absolute and differential metrics. For example, for experimental metrics, a differential metric might be able to determine how much less vulnerable a site is with an intrusion detection system (IDS) than without one.

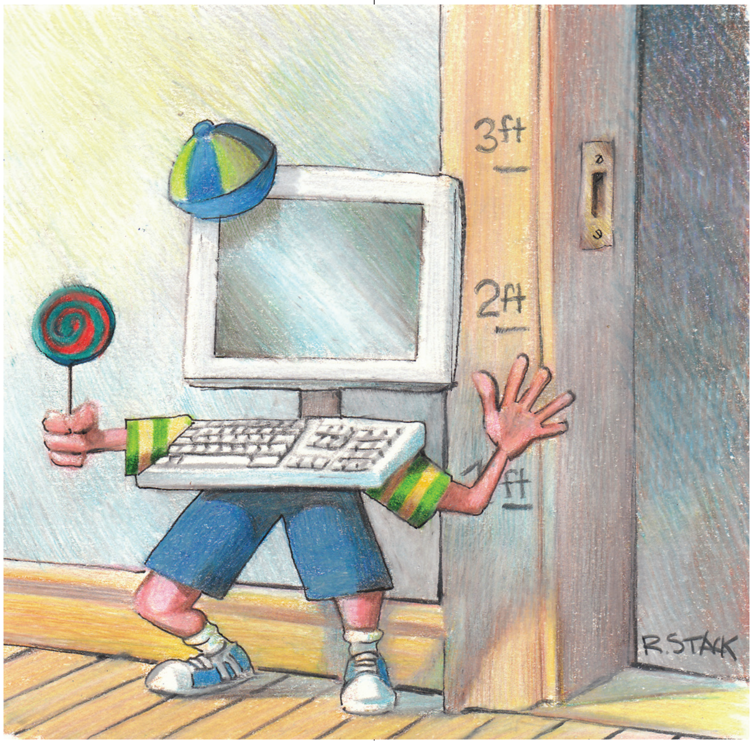
Computational-Complexity Metrics

Over the past two decades, cryptology has built a strong theoretical foundation by defining classes of adversaries on the basis of the amount of information (for example, a chosen-ciphertext attacker) and computational power (for example, a polynomial-time adversary) available to them. In cryptology, certain ciphers are argued to be strong because breaking them requires solving a problem believed to be hard (for example, RSA and factoring).

Challenges. In some cases, analogous computational-complexity metrics seem promising for system security but pose two main challenges. The first is the difference between mathematical abstractions and real implementations. The gap between theoretical cryptography results and practical cryptanalysis illustrates this: although no one has found a fast factoring algorithm, RSA implementations are regularly broken because of side channels (such as timing and power consumption), poor random-number generation, insecure key storage, message formats and padding, and programming bugs.[1] For system security, the gap between models simple enough to use for metrics and actual implementations is even larger. To make progress, we need metrics that work on more concrete models of actual systems, or ways to build systems that refine models without introducing security vulnerabilities.

The second problem is that it seems unlikely that we can reason well about adversary creativity. This argues for metrics that assume that adversaries can efficiently search the entire space of possible actions. Perhaps we can develop complexity metrics that analyze that space and the maximum effectiveness of different search strategies.

An example metric: automated diversity. Computational-complexity metrics are promising for evaluating security techniques based on automatically generated diversity. One example is *address space randomization*, which most modern OSs, including Windows 7 and Linux, incorporate in some form. This technique uses a cryptographic key to randomize the placement



of objects, so we might be able to measure an attack's difficulty on the basis of the amount of entropy in this key.

In practice, however, such a metric depends on many assumptions about how much of the entropy an attacker must break to launch a successful attack and whether attackers have other ways to learn the randomized state. Many implementations of randomization techniques have proven much weaker than would be estimated by this simple security metric.[2,3]

One research direction is to use randomization more systematically, eliminating some of the assumptions needed for previous security arguments or, in the case of *N-variant systems*, eliminating the need to keep the randomization key secret.[4] Promising approaches in this area include developing metrics that capture the implementation issues beyond just the randomization entropy, and understanding how composing and dynamically adapting randomization techniques impacts system security.

Economic and Biological Metrics

An alternative is metrics based on economic or biological approaches. A research community has emerged around economic approaches (embodied by the annual Economics of Information Security workshops). Metrics based on the expected cost required to compromise a system can help guide system designers. However, for most computing systems, these metrics' parameters are somewhat arbitrary. Regarding biological approaches, metrics for network vulnerability can be based on epidemiology, but they depend on abstract models of network nodes.

An example economic metric: cost-based IDS. IDSs have become critical in securing computer system infrastructures. IDS developers strive to maximize the detection accuracy and bandwidth of deployed systems in realistic environments. The goal is to develop efficient detectors with high true-positive rates while minimizing false-negative and false-positive rates.

However, even after many years of development, IDSs still have marginal detection rates and high false-positive and false-negative rates, especially when detecting novel intrusion attempts—*zero-day attacks*. The goal of detecting all true intrusions while minimizing false alerts is complicated by time and processing-speed constraints, especially for high-speed networks. So, it's desirable to focus detection on the costliest, most important attacks while ignoring as much benign traffic as possible.

Toward this goal, researchers have defined cost-based detection metrics for IDSs.[5] For each network event determined to be an attack, this approach estimates a response cost and a damage cost on the basis of factors such as the criticality of the targeted system component. If the damage cost is greater than the response cost, the system might initiate a response. These metrics in total define a cost-based metric for system defense that's closely related to "stop loss" metrics employed in fraud and risk management systems for financial-transaction systems.

An example biological metric: polymorphic-engine strength. Code injection attacks have traditionally received much attention from both security researchers and the black hat community, and researchers have proposed a variety of defenses. For example, many approaches capture a representation of the exploit to create a signature for use in detecting and filtering future versions of the attack.

Contemporary polymorphism techniques have rendered these defenses all but obsolete,[6] due largely to the sophisticated obfuscation techniques embodied in a number of publicly available (via Metasploit) polymorphic engines that stress various defenses' limits. These engines essentially mimic biological processes of mutation and diversity in evolutionary contexts at a macro level, and of diversity in the injection vectors at the micro level.

Judging different polymorphic engines' relative strength is difficult. To evaluate their relative strength, and hence the relative effort to defend against the wide range of malware variants they produce, researchers have established a set of metrics that capture the variability and distinctiveness of samples they generate. Two metrics are particularly promising.

The first is *variation strength*. An engine's variation strength measures its ability to generate sequences of length n that span a sufficiently large portion of n-space. This metric offers insight into the magnitude of the set of signatures that might be needed to accurately encapsulate all malware that a particular engine generates.

The second metric is *propagation strength*. For the sequence of malware samples that an engine can generate, the engine's propagation strength characterizes its efficacy in making any two samples look different from one another. This metric aims to quantify the information gain obtained by isolating a few samples from a particular engine.

Using a hybrid metric combining these two metrics, we can analyze polymorphic engines and measure their relative obfuscation utility. This yields a scaled score for each engine that we call the *relative polymorphism strength score* or *p-score*. We then rank-order each engine, identifying the most challenging to defend against.

These metrics aren't just useful for judging an exploitation's effectiveness. Polymorphic engines might aid defenses by generating a range of diversified code to thwart adversarial analysis and exploitation of a target system.

Empirical Metrics

There's also value in developing empirical metrics based on carefully structured experiments. This is analogous to how mechanical safes are rated on the basis of their ability to withstand different classes of attacks over a given time period. (For example, a TL-30 safe can withstand 30 minutes of attacks involving various mechanical and electrical tools.)

Red-team exercises are sometimes used to evaluate system security. However, current red-team exercises are too ad hoc to produce a meaningful measurement beyond knowing how easily a particular red team could compromise a given system. So, they're used primarily to identify apparent weaknesses in a particular deployment. Perhaps we can develop more systematic approaches to adversarial metrics.

Modeling adversaries. The main challenge in doing meaningful computer security experiments is modeling adversaries. Better experiments require improvements in adversary models, as well as in designing more reproducible experiments that don't depend so much on accurate models of adversary behavior. One way to improve adversary models is to systematize current knowledge about real adversaries and use it to develop experimental adversary models. Perhaps this approach could help us develop a canonical attacker model that we could use in a wide class of experiments, rather than rely on ad hoc models created by individual experimenters.

Generally, however, using experiments to evaluate new security mechanisms seems extraordinarily difficult. There's little cause for optimism that we could develop a useful model of a creative adversary attacking a new design.

An example metric: decoy properties. Decoy documents are honeyfiles that detect system compromise by adversaries, especially those with insider privileges. They aim to confuse and deceive adversaries by leveraging uncertainty to reduce the knowledge the adversaries ordinarily have. Accordingly, they can serve as effective bait in a trap-based defense system.

We can aid the design of decoy-based systems by evaluating certain key properties of decoys, such as enticement, believability, variability, and conspicuousness.[7] Each property is measurable and can serve as a guide for alternative decoy designs. For example, you can measure believability through a human subject study.

Borrowing the cryptography community's idea of perfect secrecy, we define a perfectly believable decoy as one that's chosen with a probability of 1/2 over all trials. That is, a perfect decoy is completely indistinguishable from one that isn't perfect. Although in practice, automatic construction of perfect decoys might be unachievable, this metric gives us a goal to strive for when designing and implementing deception systems.

An example metric: fuzzing complexity. Protocol fuzzing has proven extraordinarily powerful for finding implementation flaws. Fuzzing, though, isn't a matter of feeding purely random data to a target. It's guided randomness, with the guidance based on the input-language specification. A more complex input language, then, requires more complex guidance. Given that code complexity correlates well with the number of bugs (and hence the number of security holes), we could use the size of the fuzzing guidance input to measure a specification's inherent insecurity, independent of any implementations.

The Case for a Metrics Research Program

The fundamental problem with any security metric is that most security holes are due to buggy code. For example, our cryptography is very strong. However, more than a decade ago, a US National Academies study noted that 85 percent of reported vulnerabilities couldn't be fixed through cryptography; they were due to configuration or coding errors.[8] Nor are errors confined to implementers; as Carl Landwehr and his colleagues noted, many security problems are in the specifications.[9]

Any attempt to measure security, then, must confront another problem: we have no way to measure the residual bugs—and hence vulnerabilities—in a given body of code. Traditional methods, such as looking at the reported error rates over time and extrapolating, don't account for the adversarial process. Finding holes isn't a purely stochastic process but is driven by the effort expended, and you never really know an adversary's goals, capabilities, or resources. A useful metric, then, must account for varying degrees of damage of an attack against a code base of unknowable quality.

Systematization has its own related challenges. Without a general theory of bugs—and hence of security holes—systematization can reduce to a large catalog, with only a few generally useful categories. Again, though, bugs are too varied.

Finally, even gathering empirical data is challenging. It must be a large, uniformly selected collection. Open source software change logs are one useful input, but commercial code has very different properties. Unfortunately, concerns about proprietary information can inhibit the release of data. More fundamentally, the

Related Reading

For more background (with significant skeptical analysis), see Steven M. Bellovin's 2006 *IEEE Security & Privacy* article "On the Brittleness of Software and the Infeasibility of Security Metrics" (http://doi.ieeecomputersociety.org/10.1109/MSP.2006.101) and his 2010 PowerPoint presentation "The Cybersecurity Challenge" (www.cs.columbia.edu/~smb/talks/bellovin_onassis_challenge.pdf).

Also of interest is the JASON program's 2010 report, the *Science of Cybersecurity* (www.fas.org/irp/agency/dod/jason/cyber.pdf).

raw data isn't very useful without a great deal of context information and analysis. When was the code written? What's the design and revision history of the module and its callers? What development and testing practices were followed? How experienced were the programmers? These questions and more must be answered for each data point gathered.

Securing Legacy Software

Clearly, the development of a science of security, and particularly of security metrics, is a very hard problem. If an accurate security metric were possible, defense would be moot. A formal means of measuring the security of a large legacy software system would imply that every vulnerability was identifiable and quantifiable. So, we would simply repair the system to drive that metric to 0. Problem solved!

Several major research programs are dealing with securing legacy software, recognizing that perfect identification of all vulnerabilities is unrealistic. Over the next few years, these programs will likely produce demonstrably more secure software by combining static and dynamic analyses with runtime diversification and compartmentalization. The strategy is largely to prevent many more attacks than we can today and to contain whatever damage a successful attack might make. Time will tell how successful these approaches will be.

Layers of Defense for Resiliency

We posit an alternative strategy that

- extends the principle of defense in depth and
- provides a pathway to securing legacy systems with metrics that can adequately evaluate the safety, security, and resiliency of the mission the legacy software supports.

We conjecture that we can apply metrics to layers of defenses that manage and control the interaction of different kinds of adversaries with a target system.

Any system we design involves many layers, such as

- physical layers, such as the boundary between hardware and firmware, which control the movement of code and data;
- communication layers, which manage the movement of bits from network to host;
- access control layers, which manage users and their credentials and the access to applications and data; and
- management layers, which control people's access to critical systems and resources.

These layers will be composed and combined into one system that supports the mission we seek to defend.

The goal is to devise measurable systems that are resilient to failure under reasonable estimates of adversary effort. So, we believe we can measure resiliency by designing layers of defense that provide a lower bound on the resources adversaries need to pass through each layer to achieve their ultimate goals.

In a naive system, the strength is roughly proportional to the number of layers and can be measured as such.[10] Today's security, then, is linear and could thus be overcome by an adversary with linearly increasing resources. In the physical world, though, we can bond together materials, even dissimilar ones, to achieve far greater strength than a simple linear increase. Unfortunately, many of today's security mechanisms aren't readily composable. If we can build layers in the physical world that reinforce each other, perhaps by repairing flaws, we can do the same in the cyberworld. We can then develop a composition rule for each layer's security metrics.

Different Adversaries, Different Layers

At least three kinds of adversaries might interact with a system. The remote, nation-state actor might have numerous network paths to reach a target system. A corresponding defense would force the adversary to traverse a number of layers, each designed to adequately estimate how much effort the adversary might employ to successfully attack a system. We can design cost-based IDSs that limit the rate of information flow, using an estimate of how much data might be lost.

An expert operator adversary would presumably know all the target system's features and capabilities. In this case, operator behavior models might serve as a layered defense by specifying and limiting how users interact with mission-critical systems.

The insider expert developer who contributed to the design, specification, and implementation of the target system is probably the most dangerous adversary. Various randomization strategies that create diversified system code and layout might serve as another layer limiting the damage that even a developer might mount.

In each case, we assume the adversary has a certain level of knowledge of his or her quarry, and access to the system through different paths. Some of these paths might be monitored and defended; in other cases, no such defense might exist. Given a particular threat and adversary model, we envision alternative layered defenses that proactively manage and control access to a target system from any possible path. In many cases, we might need to design and implement new infrastructures and layers. For example, we might need host monitoring infrastructures to force internal users through a layer that manages their access to only a single compartment in a larger, critical system.

Measurable Defense in Depth

The key goal is *measurable defense in depth*. We must design each layer with measurement in mind. In some cases, prior research results or previous successful penetrations by particular kinds of adversaries might guide these measurements.

Generally, specifying a useful, verifiable metric will require considerable research. For example, for cost-based IDS metrics, the cost of traversing a layer protected by an IDS requires considerable knowledge of the target system, the data it hosts, and the relative value of the targeted system's different components. This knowledge is largely in the minds of the system designers, who might be adversarial. So, the metrics that we apply to each layer must be verifiable and defensible.

A research program that develops clear, concise, and useful security metrics in the context of layered defenses would considerably help to establish security as a science and guide security engineering as a formal professional discipline. The technical impediments are great, but the payoff would be transformative. Much work remains, but the metrics we've described might point the way to successful outcomes. Even if these metrics are unsuccessful, understanding deeply why our current models and metrics don't adequately capture our modern and legacy systems' security properties will help teach us how to build better, more measurable systems. □

Acknowledgments

This article is based partly on the workshop report for the 2008 US National Science Foundation / Intelligence Advanced Research Projects Activity / National Security Agency Workshop on the Science of Security. This meeting brought together 43 leading researchers in computer security and other relevant fields to consider the state of scientific research in computer security and to identify steps toward establishing a stronger scientific basis for computer system security. Slides from most of the presentations are available at the workshop website, http://sos.cs.virginia.edu/agenda.html.

References

1. J.P. Degabriele, K.G. Paterson, and G.J. Watson, "Provable Security in the Real World," *IEEE Security & Privacy*, vol. 9, no. 3, 2011, pp. 33–41.
2. H. Shacham et al., "On the Effectiveness of Address-Space Randomization," *Proc. 2004 ACM Conf. Computer and Communications Security* (CCS 04), ACM Press, 2004, pp. 298–307.
3. A.N. Sovarel, D. Evans, and N. Paul, "Where's the FEEB? The Effectiveness of Instruction Set Randomization," *Proc. 14th Usenix Security Symp.*, Usenix Assoc., 2005, pp. 145–160.
4. B. Cox et al., "N-Variant Systems: A Secretless Framework for Security through Diversity," *Proc. 15th Usenix Security Symp.*, Usenix Assoc., 2006, pp. 105–120.
5. W. Lee et al., "Toward Cost-Sensitive Modeling for Intrusion Detection and Response," *J. Computer Security*, vol. 10, nos. 1–2, 2002; pp. 5–22.
6. Y. Song et al., "On the Infeasibility of Modeling Polymorphic Shellcode," *Proc. 14th ACM Conf. Computer and Communications Security* (CCS 07), ACM Press, 2007, pp. 541–551.
7. B.M. Bowen et al., "Baiting Inside Attackers Using Decoy Documents," *Security and Privacy in Communication Networks*, Springer, 2009, pp. 51–70.
8. F.B. Schneider, ed., *Trust in Cyberspace*, National Academy Press, 1999.
9. C.E. Landwehr et al., "A Taxonomy of Computer Program Security Flaws," *Computing Surveys*, vol. 26, no. 3, 1994, pp. 211–254.
10. S.M. Bellovin, "On the Brittleness of Software and the Infeasibility of Security Metrics," *IEEE Security & Privacy*, vol. 4, no. 4, 2006, p. 96.

Sal Stolfo *is a professor in Columbia University's Department of Computer Science. Contact him at sal@cs.columbia.edu; www.cs.columbia.edu/~sal.*

Steven M. Bellovin *is a professor in Columbia University's Department of Computer Science. Contact him at smb@cs.columbia.edu; www.cs.columbia.edu/~smb.*

David Evans *is an associate professor in the University of Virginia's Department of Computer Science. Contact him at evans@virginia.edu; www.cs.virginia.edu/evans.*