Computer Security Strength & Risk: A Quantitative Approach

A thesis presented
by

Stuart Edward Schechter

to
The Division of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
May 2004

Thesis advisor

Michael D. Smith

Author

Stuart Edward Schechter

Computer Security Strength & Risk:A Quantitative Approach

Abstract

When attacking a software system is only as difficult as it is to obtain a vulnerability to exploit, the security strength of that system is equivalent to the market price of such a vulnerability. In this dissertation I show how *security strength* can be measured using market means, how these strength measures can be applied to create models that forecast the *security risk* facing a system, and how the power of markets can also be unleashed to increase security strength throughout the software development process. In short, I provide the building blocks required for a comprehensive, quantitative approach to increasing security strength and reducing security risk.

The importance of quantifying security strength and risk continues to grow as individuals, businesses, and governments become increasingly reliant on software systems. The security of software deployed to date has suffered because these systems are developed and released without any meaningful measures of security, causing consumers to be unable to differentiate stronger software products from weaker ones. Even if we knew that we could make systems measurably stronger, the lack of accurate security risk models has blurred our ability to forecast the value to be gained by strengthening these systems. Without the tools introduced in this dissertation, those of us tasked with making security decisions have been forced to rely on expert opinion, anecdotal evidence, and other unproven heuristics.

Contents

Acknowledgments

The first paragraph of this section is a mad-lib. Each number in the list below describes a term of your choice which should be filled into the paragraph below it.

Please fill in a term for:

(1) an inanimate object or a synonym for idiot.
(2) the name of a third-world country.
(3) the name of an academic field not closely related to Computer Science.
(4) the name of a highly renowned graduate school.
(5) the name of a profession that pays minimum wage.
(6) the name of the place you would least like to live in.
(7) the name of a software company.

Michael D. Smith could successfully advise a __________________ (1) with a pre-school level education from __________________ (2) to complete a doctoral degree in the study of __________________ (3) at __________________ (4). If it had not been for his support, flexibility, sense of humor, and relaxed attitude I suspect that I would now be employed as a __________________ (5) in __________________ (6), or worse might be working in the standards compliance division of __________________ (7).

Glenn Holloway has been like a second advisor to those of us in Mike's research group. He has the patience to read every paper we write, the ability to quickly understand what the paper is about, a knack for figuring out how to best improve the paper in the time available, and the attention to detail to find the typos. His endless knowledge of the tools of the trade, from LaTeX to Visual Studio, has proved invaluable. I suspect that having Glenn in our group reduces the time-to-graduate for Mike's students by at least a semester.

Michael Rabin and H. T. Kung provided invaluable advice from early in my graduate career through the final draft of this document. I especially appreciate their encouragement to attack problems I found interesting even when these problems were not connected to existing research projects within DEAS.

Marty Loeb was kind enough to read some of the earlier, less polished, thesis drafts and fly to Boston for the defense. Without the seminal papers he wrote with Larry Gordon, and those of Ross Anderson, Jean Camp (my unofficial fifth committee member), Hal Varian, and the other founding organizers of the Workshop on Economics and Information Security, I would likely still be looking for a dissertation topic. I might not have discovered this research area if Ron Rivest had not taken the time to point me in their direction.

I cannot count the number of times Susan Wieczorek, the fairy godmother of DEAS graduate students, has waved her wand to make bureaucratic tasks and paperwork disappear.

Much of the background research in risk management was performed over a summer visitation at the University of California at Berkeley that was kindly arranged by Doug Tygar and Hal Varian.

For my grandparents and parents,
whose examples I can only aspire to follow,
and for the students of Leverett House,
who I implore not to follow mine.

Chapter 1

Introduction

How secure is a software system? How secure does a system need to be? By how much can security be improved by putting safeguards into place?

Those of us who work to secure systems ask these questions in order to evaluate the efficacy of our security efforts. We seek answers that provide measures of how effective our security efforts have been in reducing risk, or that forecast the reduction in risk that we expect from further security efforts. Often this means estimating past values, and forecasting future values, of such security risk metrics as the frequency of security incidents or the annual cost of these incidents. To make security decisions we must use these metrics to gauge how the choices we make will influence the effectiveness of our security strategies in reducing our security risk.

A general methodology for modelling security risks has proved elusive because the security of a system is affected not only by our actions, but by the strategic choices of our adversaries. What's more, security questions are approached differently when the answers are to be presented in terms meaningful to these adversaries.

When an adversary asks how secure a system is, his primary concern is most likely to be either the personal risk to his safety or freedom from attacking the system, or the difficulty he will face in attempting to subvert or bypass the system's safeguards. An adversary will perceive a system with an additional safeguard to be more resilient to attack only if that safeguard interferes with the plan of attack that would be used by the adversary in the safeguard's absence. A system's security becomes *stronger* as more time, effort, or other resources are required to subvert it. From an adversary's perspectives, this *security strength*, in combination with the *personal risk* of the attack to the adversary's reputation, safety, or freedom, are the metrics of interest when evaluating the security of a prospective target. For example, when using a cost-benefit analysis to evaluate the attractiveness of a potential target system, that system's security strength and the personal risk incurred in attacking that system represent deterrent costs that must be weighed against the benefits of targeting the system.

Because security strength measures the resource cost of breaching a system's security, it is fundamentally an economic measure. Formal methods of Computer Science, such as information theory, have been used to address questions of security strength when these questions can be translated into assertions of what can and cannot be calculated. For example, the one-time pad cipher [100] and Shamir's secret sharing algorithm [99] have been proven to be secure against certain classes of attack, regardless of the resources available to the attacker. However, the applicability of computational approaches to security strength is severely limited. Computer systems and networks, which make up the environment in which security mechanisms are deployed and in which adversaries attack, have become too complex to be addressed using formal proofs of computability.

In this dissertation I overcome these limitations by providing an economic approach for measuring the security strength of software in units of dollars. Because this methodology does not rely on computational assumptions, it can be applied in situations where purely computational methods are not applicable.

1.1 Economic approaches to security

Economic approaches have only recently been introduced into the study of computer security. In his seminal paper, Ross Anderson [9] asserts that perverse incentives to create insecure systems have caused as much harm as technological shortcomings. Both Anderson [7] and Hal Varian [113] describe how security failings result when those parties who are charged with protecting systems are not the parties who must pay the consequences when security is breached. Examples include ATMs deployed in Europe, which banks were tasked with securing but for which customers paid the costs of ATM fraud, and distributed denial of service attacks, in which computers secured by home users were compromised by hackers and then used to attack other's networks.

Other researchers have gone beyond using economics to describe problems of information security and have used economic approaches in formulating solutions. Jean Camp and Catherine Wolfram [23] proposed addressing the problem of denial of service attacks through governmental issue of vulnerability credits (similar to pollution credits). In their proposal, owners of systems left vulnerable to network attack are made to pay for the negative externality that results when these systems are compromised and used to attack other systems.

Researchers have also applied economic models to security decision making and measurement. Lawrence Gordon and Martin Loeb [52] have used economic models to bound the amount that a firm should spend on security as a function of the size of the potential loss. Kevin Soo Hoo [59] proposed measuring changes in security risk by extending metrics used to gauge the absolute level of security risk. By doing so, these metrics could be used to choose the most effective bundle of safeguards possible within one's budget constraint.

At the first Workshop on Economics and Information Security, many researchers argued that the software industry should borrow from the quantitative models of safety and reliability used in the insurance industry [16, 41, 96]. They reasoned that if insurance companies can model the safety of a car and driver, or gauge the risk of the failure of a safety system at a factory, then similar models could be developed to assess software security risks. They proposed that insurers could then drive security investment by discounting policies to firms that implemented safeguards that reduce risk. While insurers could play an important role in managing security risks if accurate risk models were available to them, in the past these firms have relied on actuarial tables to gauge risk. This approach is acceptable for safety risks for which historical data can be used to forecast future event likelihoods. However, the actions that cause security breaches in computing systems are strategic, and the strategies available to both the adversaries and those who secure systems are in a constant state of flux. As such, no actuarial table or risk model available today can accurately forecast the effect of choosing one security strategy over another.

1.2 A new approach

Past research has left open the problem of *quantitatively* measuring the security strength of software and quantitatively forecasting security risk metrics. If we are to make good security decisions today we need to be able to estimate the effect each of these choices will have in reducing our security risk in the future. Because the security strength of a system deters attack, the influence of security strength should be accounted for when forecasting security risks.

Most existing security risk models already take into account the deterrent effect of the personal risk incurred by an adversary who chooses to attack a system. However, for networked software, security strength often has a greater effect in deterring attack than the presence of risks that an adversary must incur to stage an attack. After all, there is little risk to searching software for vulnerabilities when that software can be copied to one's own computer and tested in private. When vulnerabilities are found, attackers can minimize their personal risk by staging attacks remotely and rerouting their communications in order to hide their identities.

Measures of security strength have additional uses beyond forecasting security risk. They can be used to improve processes for making systems stronger, and they can help consumers differentiate more secure systems from less secure ones.

This dissertation is structured to attack these problems in a bottom-up manner. Before we can address problems of measuring and forecasting security metrics we must first formalize how these questions should be posed. The study of security lacks a common language, and so I begin in Chapter 2 by introducing a lexicon and conceptual framework for use in addressing questions of security. The terms and approach are not new, but are rather the distillation of language and techniques from a diverse body of security and risk management literature.

I then detail barriers that have limited the measurement and forecasting of security risk and strength in Chapter 3. This background material includes relevant history of security strength from century-old ciphers to public key cryptography. It also covers risk management, a field developed to model *reliability and safety risks* posed by natural failures that has recently struggled to adapt to the *security risks* posed by adversaries.

In Chapter 4, I introduce a methodology for measuring the security strength of software systems. I argue that the strength of systems with known vulnerabilities is negligible, and that the security strength of systems with no known vulnerabilities is bounded by the market price to obtain a new vulnerability. The metric I propose is the market price to discover the next new vulnerability, rather than the actual costs expended to discover a vulnerability, because the value must be measured under circumstances in which the next new vulnerability has yet to be found.

One immediate application of security strength metrics, their use in differentiating products, is introduced in Chapter 5. This new approach, which applies the framework of the previous chapter, is significant because the lack of security metrics has left consumers unable to distinguish more secure products from less secure ones. When consumers cannot make this distinction, software developers cannot justify investments in making their products more secure.

In Chapter 6, I integrate market methods for security measurement into the software development process so that programs can be built to be more secure from the start. The continuous measurement of security strength during the development process enables software development firms to create incentives for developers and testers to increase the security strength of the firm's products. These new techniques have the potential to transform the software development process so that trade-offs between security, cost, and time to market can be better understood and managed.

With tools in hand to measure and improve security strength, I show in Chapter 7 how to better model and forecast security risks by employing measures of security strength. As evidence for the utility of security risk forecasting models, I explain why these models have been successful outside the realm of computer security, against threat scenarios where the effect of security strength in deterring adversaries has been negligible.

What remains is to forecast how the introduction of new threat scenarios, for which historical data is not available, will affect risk. Anticipating new threats is essential because no analysis of past security incidents can prepare one for the potential risks posed by future threat scenarios that can result from radical changes in your adversary's strategy or technology, and thus fall outside of existing risk models. New and evolving threats have the potential to affect a variety of systems and a potentially catastrophic number of installations of these systems. The best we can do to prepare is to anticipate new threats, evaluate their viability, and do our best to understand their impact on our security risks without using historical data, as none is available. Chapter 8 shows one approach to modelling a specific class of threat scenario: attacks on large numbers of systems with the intent of financial gain.

When combined, the techniques and methods presented in this dissertation can be used to answer the questions with which it began. We can measure how secure a software system is by determining the market price of a vulnerability. We can forecast how secure a system needs to be by applying models of security risk that employ security strength. Similarly, I show that these models can be used to gauge how much security can be improved by putting safeguards into place.

Thus, the tools I introduce for the measurement improvement of security strength and the models that I refine to better forecast security risk can be used to create a comprehensive quantitative approach to security. These quantitative techniques can be immediately applied to measure and improve the strength of existing software. As we acquire security strength statistics, we can begin to employ security risk models to forecast risk even in newly released software.

Chapter 2

What is security?

Security

1. The process of identifying events that have the potential to cause harm (or *threat scenarios*) and implementing safeguards to reduce or eliminate this potential.
2. The safeguards, or countermeasures, created and maintained by the security process.

At its simplest, security is the process of protecting against injury or harm. Security also describes the safeguards, or countermeasures, put in place by that process. In computer security, harm implies a loss of desired system properties such as confidentiality, integrity, or availability. The goals of security may be distinguished from those of reliability in that they focus on preventing injury or harm resulting not only from random acts of nature, but also from the intentional strategic actions of those with goals counter to your own.

Used to describe a diverse and ever growing set of processes, the word 'security' appears to be condemned to imprecision by the weight of its own generality. If security is, in the words of Bruce Schneier, 'meaningless out of context' [95, page 13], can a useful definition remain relevant in the face of ever changing technological contexts? To address these matters, I have presented at the top of this chapter a general definition of security as the combination of two sub-processes that are themselves free of references to any specific technology or application.

In this chapter I will further refine the concepts of threat scenarios and safeguards on which this definition of security is built. I will describe existing tools for modelling threats and discuss their limitations. Finally, I will argue that this definition is sufficiently general for modelling questions of security as they have changed over time.

2.1 Threat scenarios

Threat Scenario

1. A series of events through which a natural or intelligent adversary (or set of adversaries) could cause harm.
2. *(In Computer Security)*
   A series of events through which a natural or intelligent adversary (or set of adversaries) could use the system in an unauthorized way to cause harm, such as by compromising the confidentiality, integrity, or availability of the system's information.

Like security, safeguards must be understood within a context. When presented with a new safeguard the first question one is likely to ask is what is it intended to guard against? This is why the security process, as defined above, begins by describing chains of events with undesirable consequences that we wish to avert. We call these chains of events threat scenarios.

In the security and risk management literature, the word threat does not refer to the adversary who may cause harm, but to the chain of events that begin with the adversary (who may also be known as a 'threat source') and end in damage or harm [42, page 37] [79, page 21]. The word scenario is also used to describe such chains of events [3, 6, 42, 67]. *Threat scenarios*, as I will call them[1], provide the context for thinking about security. By establishing a common understanding of the events that lead to harm in a threat scenario, parties can better agree on what is at stake and what safeguards may reduce the risk. Exploring threat scenarios by adding detail not only helps us understand what can go wrong to cause security failures, but also helps to ask how and why these events may occur and who may be likely to cause them to occur.

The information security literature divides threat scenarios into three data-centric categories, based on what desired property of the data is lost: confidentiality, integrity, or availability. These basic threat scenarios might be written as follows.

- Information is exposed to someone who should not have access to it.
- Information is modified in a manner contrary to policy.
- Authorized users are prevented from accessing information or resources in a timely manner.

As we dig deeper into the who, how, and why harm may occur, the descriptions of these threat scenarios can become much more detailed. Taking the first basic threat

[1] I will occasionally use 'threat' alone to describe the most basic threat scenarios in which no means of attack is specified.

scenario, described above, and detailing how a violation of confidentiality could come about might result in the description of the following chain of events.

> An employee uses a laptop running a packet sniffer in order to eavesdrop on data sent over the local area network connected to his office. He or she reads confidential management documents attached to emails as they are in transit between managers, and then sells these documents to a competitor.

If the events described above occur, then we say that security has been *breached*, or that the threat scenario has been *realized*. The more detailed the description of the threat scenario, the better we can can understand the likelihood of a breach, the potential harm a breach may cause, and what we can do to prevent this scenario from being realized.

For example, a firm with little need for confidentiality might examine the above scenario and determine that the consequences do not justify additional security measures. Another organization may surmise that this specific scenario is already well guarded against. This may be true if network jacks lead directly to switches, rather than hubs, such that network cables only carry information that originates or terminates at the machine connected to the data jack.[2] Another organization might conclude that the risk posed by this threat scenario can be reduced if managers are encouraged to share files using a secure file system, rather than an insecure email network. All of these conclusions can be made because the threat scenario is described in adequate detail.

On the other hand, dividing general threat scenarios into many unnecessarily detailed sub-scenarios may make any form of analysis intractable. We might ask if the

[2]Note that while the location of the network switch may counter this specific threat scenario, it will not safeguard against all other eavesdropping scenarios.

threat scenario above need only apply to situations in which an employee eavesdrops using a laptop and his own network connection, or whether a single threat scenario should be used to describe all network eavesdropping attacks. Similarly, we may ask whether we need separate threat scenarios for each possible motive for the theft of information, or whether we should simply assume the most compelling motivation we can conceive of.

Containing the potentially exponential explosion of increasingly detailed threat scenarios is of particular concern as past approaches to security and risk management have failed due to the multitude of scenarios generated [59, page 7]. As we start with more general threat descriptions and divide them into more detailed ones, we must be sure only to make these divisions when the value of the insights gained outweighs the cost in added complexity. Fortunately, procedures for generating and detailing threats are well known, and are described in Section 2.3.

2.2 Safeguards

Safeguard

> A policy, process, algorithm, or other measure used to prevent or limit the damage from one or more threat scenarios.
>
> *Synonyms:* countermeasure, control, security measure

Safeguards act to prevent or reduce the damage caused by the realization of one or more threat scenarios. Safeguards are also often called countermeasures, controls, or security measures, though the last of these terms will be used sparingly to avoid confusion with the process of measuring (gauging the level of) security. Safeguards may take any number of forms from physical barriers, to sensors, to software algorithms.[3] Safeguards needn't even be objects unto themselves, but may encompass investments in improving existing policies, procedures, or processes such as design, development, or quality assurance. Safeguards can be compliments, working more effectively in combination than when apart, or substitutes.

When adding a safeguard to a system, one must consider that an adversary might break through the protections that the safeguard offers, or find a way to bypass the safeguard entirely. Thus each safeguard introduced may lead to the introduction of additional, more detailed, threat scenarios that describe events in which safeguards are circumvented or penetrated. These new detailed scenarios may in turn lead to the introduction of new safeguards. Security is often referred to as an 'arms race' because of this cycle in which new safeguards and new plans of attack are introduced to counter each other. The process of adding safeguards, and responding to new threat scenarios targeting those safeguards, terminates when all remaining unimplemented safeguards are impractical or uneconomical.

Those charged with securing systems may disregard threat scenarios when the responsibility for implementing safeguards lies outside the system's boundaries. For example, printers are not expected to control access to documents released into their output trays. Rather than requiring authentication to take place at a printer before documents are released, the onus to safeguard against printed documents reaching the wrong hands has been placed upon organizational procedures for locating print-

[3]Using a single term to encompass organizational procedures and algorithms is not revolutionary. The use of the word 'software' to refer to organizational procedures appears in risk management texts as early as 1980 [105, page 20].

ers and distributing the documents that they print. Defining clear boundaries for the components of a system and their inputs and outputs, is known as a *systems approach* [105, page 11] and its necessity is widely accepted. Well-defined boundaries are essential for determining which threats a system component (itself a system) should safeguard against, and which threats can be disregarded and countered elsewhere within the larger system in which the component will reside. The documentation of these delineations of responsibility is essential.

When using a systems approach one sees that security is not a single feature one can buy in a product. Rather, the prodcut development process and products themselves contain safeguards which, if properly combined with the safeguards of the systems and organizations that the product is integrated into, may help to counter threats. Careful specification of system boundaries and responsibilities is necessary to prevent common security failures.

2.3 Expanding and organizing threat scenarios

To choose the right safeguards to protect a system, it is necessary to understand the threat scenarios faced by that system. The first steps in this process are to find all plausible threat scenarios, add detail as necessary, and organize them so that we can best determine the effect of safeguards.

We begin the threat scenario discovery process from the simplest point possible: enumerating the most basic threat scenarios faced by the system. These simple threats scenarios are sometimes just called threats, as they are devoid of most scenario specifics such as the means of attack, the motive of the adversary, or the asset targeted.

For example, armed robbery and check fraud would be two basic, yet distinct, threats faced by a bank. These acts are worth distinguishing as separate basic threats as breaking into a bank requires a very different type of adversary, resources, and skills than passing a false check.

The choice of basic threats is subjective and is specific to the type of system being modelled. The occupants of a castle might differentiate the basic threat of castle wall penetration from external sieges intended to starve the occupants and avoid a fight. For an operating system, privilege escalation and denial of service are examples of basic threats. Basic threats need not be entirely independent. For example, in operating system security, the scenario of privilege escalation may lead to the distinctly different scenario of denial of service.

2.3.1 Trees and graphs

Once you have identified basic threat scenarios, it is time to add specifics. This is done by taking individual threat scenarios and expanding them into multiple threat scenarios, distinguished by either the means of attack, motive, type of adversary, or asset targeted. For example, a castle penetration scenario may be divided into additional threat scenarios based on means of attack. Castle walls may be penetrated by digging tunnels underneath them, using ladders or platforms to scale them, through an open gate, or by destroying the walls and going straight through them. Each time a threat scenario is augmented with more specifics a new threat scenario is formed. A tree may be formed by placing the basic threat scenario at the root, placing the augmented threat scenarios into its child nodes, and then repeating the process by

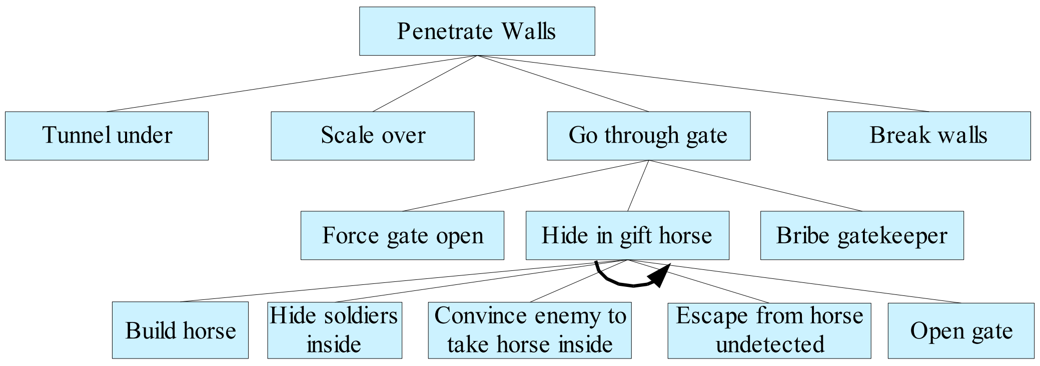

Figure 2.1: A tree detailing how a castle's walls may be penetrated by an invading army.

expanding each of the child nodes. When nodes no longer benefit from the addition of further detail, the leaves that remain represent detailed, unique, threat scenarios.

Figure 2.1 is an example of such a tree applied to a castle and the threat of penetration by an enemy army. The basic threat scenario appears as the root node at the top level of our tree. The four ways we can envision the walls being penetrated are represented by the second level of the tree, which contains the children of the root node. If *any* of these child threat scenarios should be realized then the parent threat scenario is also said to have been realized.

While we could expand each of these child nodes, I have only expanded the scenario in which the enemy passes through the castle gate. There are a variety of ways that the adversary could carry out the gate-penetration scenario. To illustrate this we expand this node as we expanded the root node. Its children may represent options available to the enemy army such as (but are not limited to) bribing the gatekeeper, forcing the gate open, or even emulating the ancient Greeks by hiding soldiers in a statue of a horse.

The relationship between the “hide in gift horse” node and its child scenarios is different than the other parent/child relationships in the tree. In order for the gift horse scenario to be realized, the enemy must have the means to perform not just one, but all of the tasks described by the child nodes. If the horse is not brought inside the walls, or if the soldiers do not escape and open the gate, then all of the attacker’s other steps will be for naught. Many different representations have been used to indicate this all or nothing requirement, from AND gates (borrowed from the field of computer architecture) to semicircle connectors that link the edges between the parent and those child nodes that must be realized together. The latter is used in Figure 2.1, with the addition of an arrow at the end of the semicircle. The arrow represents a further requirement, that the events in each of the child nodes occur in the sequential order given by the direction of the arrow. After all, the attack will fail if the attackers convince the occupants to take the horse and then realize they have yet to hide soldiers inside.[4]

A more contemporary example of partially completed threat tree is shown in Figure 2.2, which shows how an outsider (or nonuser) can obtain root access to an operating system. For each feature described in the castle example, an analogous feature is present in this example.

Originally called fault trees [18], these tree-based threat enumeration techniques were first used to model natural threats for safety analyses. The first application of fault trees, in 1961, was to ensure the safety of missile launch control systems [117]. Fault trees were quickly adopted by the Nuclear Regulatory Commission [115] and

[4]Fans of the movie “Monty Python and the Holy Grail” [24] may recognize this scenario from King Arthur’s failed castle siege.

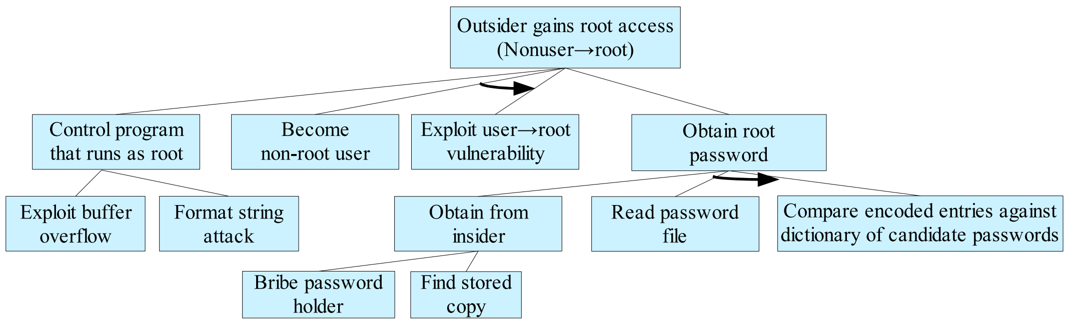
Outsider gains root access (Nonuser→root)
Control program that runs as root
Become non-root user
Exploit user→root vulnerability
Obtain root password
Exploit buffer overflow
Format string attack
Obtain from insider
Read password file
Compare encoded entries against dictionary of candidate passwords
Bribe password holder
Find stored copy

Figure 2.2: A tree detailing how an outsider (nonuser) might gain access to a networked operating system.

NASA [64]. They have been applied not only in preventing failures, but also to analyze the cause of accidents after they have occurred. They were used after the Apollo 1 disaster, Three Mile Island, and the losses of the spaces shuttles Challenger and Columbia [64, 12]. As fault tree analysis has been adapted to include adversarial threat sources for use in security analysis, they have also been called threat trees [4, 62] or attack trees [93, 94]. In addition to the graphical representation of trees, complex trees are often represented textually as one would represent the structure of an outline or a table of contents. Those interested in how these approaches evolved and where they differ are encouraged to read Appendix A.

Because fault trees evolved from the study of safety where the adversary (nature) has no sense of motive, the expansion of nodes focused exclusively on the events of the scenario and not the actors. Events were modelled as resulting from stochastic processes. Because these systems all faced the same, consistent, natural adversary it was possible to move and reuse tree branches from system to system. Safeguards could be added at the very end of the process because their efficacy, in the context of safety, could be reliably estimated using historical data.

One point of contention when using trees to model those threat scenarios that are posed by intelligent adversaries is where to place safeguards. Some assume the system being analyzed is static and so all countermeasures are assumed to be existing parts of the system [93]. Microsoft's Howard and LeBlanc propose mitigation circles [62] be placed below the leaves of the tree, with each circle addressing the threat scenario described by the leaf above it. They even admonish the reader from focusing on safeguards during the threat modeling process.

> Note that you should not add the mitigation circles during the threat-modeling process. If you do you're wasting time coming up with mitigations; Rather, you should add this extra detail later. [62, page 91]

The problem with placing safeguards below the trees is that many threat scenarios involve attacks on the safeguards themselves. When Howard and LeBlanc place encryption protocols in mitigation circles at the bottom of their trees, they leave no room in which to expand the attacks on the encryption algorithm or protocols they have chosen. As Anderson and Kwok have pointed out, safeguards do not eliminate threats, but rather transform them into less plausible threats [5, page 246].

Leaving safeguards out of the trees entirely has consequences as well. In Schneier's example attack tree [94, Figure 21.1], in which the basic threat scenario is the opening of a safe, both "pick lock" and "learn combo" appear as potential attacks. The relevance of each of these nodes, and the threat scenarios they represent, depends on the choice of a countermeasure (lock) with which to safeguard the door – a key lock may be picked and a combination may be guessed. If all possible countermeasures are assumed to be present, the tree will grow unwieldy. If any analysis is to be performed it would first be necessary to remove those scenarios that target countermeasures that are not present in the system.

Instead of trees, Caelli, Longley, and Tickle [22] use directed graphs that integrate safeguards by representing them as nodes, placed as needed, throughout the diagram. Safeguard nodes are placed below threat nodes to prevent the threat scenarios represented by those nodes from being realized. Additional threat nodes can then be placed below a safeguard node to represent attacks on that safeguard, and so the process iterates. The graph terminates at those threat scenario nodes that do not pose a great enough risk to justify further countermeasures. Safeguards may be similarly integrated into a tree based approach, though Directed Acyclic Graph (DAG) representations are more compact. Whereas a single safeguard may counter a number of threats in a DAG, the safeguard node and all of its children must be replicated in a tree.[5]

Regardless of whether scenario diagrams are trees or DAGs, integrating safeguards throughout the representation can be beneficial for understanding their effect. Placing safeguards into the diagram makes explicit the modeler's assumptions about where safeguards are deployed and what scenarios they are intended to counter. Adding safeguards into the threat modelling process also increases the chance that threat scenarios in which the safeguards are attacked or bypassed will be included in the model.

Figure 2.3 is threat scenario diagram representing the opening of a safe, adapted from one of Schneier's attack trees [94, Figure 21.1], in which safeguards have been added in the form of transparent boxes with rounded corners. The safe being modelled

[5]DAGs may be converted to a tree by making copies of each child node, and all of its children, for each of its parents.

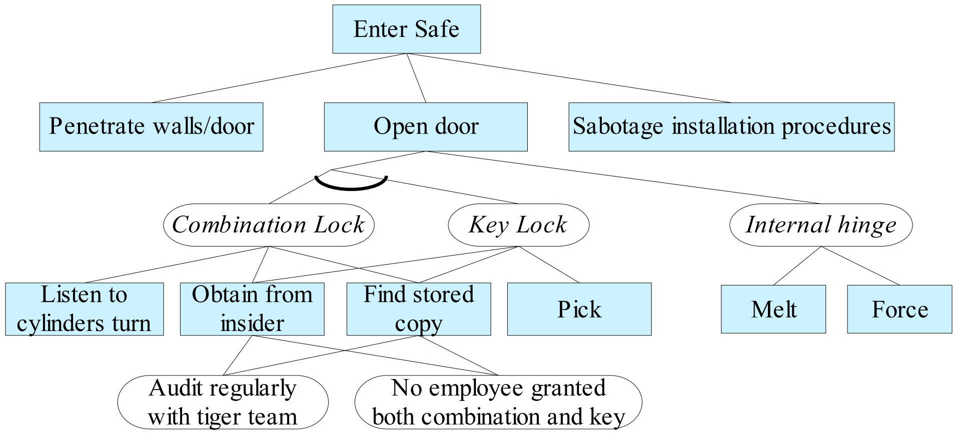

Figure 2.3: A Directed Acyclic Graph (DAG) representing the threat scenarios that result in the opening of a safe and the countermeasures used to safeguard against them.

contains both a combination lock and a key lock, configured so that both must be unlocked if the door is to be opened. In addition, the safe door's hinge is placed on the inside of the safe to make it harder to remove or destroy. The diagram is a Directed Acyclic Graph (DAG) and not a tree, as tree nodes cannot have multiple parents. For example, the two different lock safeguards share a single child node that represents that a key, be it metallic or a combination code, could be obtained from an insider.

Safeguard nodes may also have multiple parents. For example, audits that safeguard a system by simulating attempts to obtain keys both discourage employees from giving out their keys and from leaving their keys unguarded. Similarly, a prohibition against granting both the combination and the key to a single employee will help to counter two threat nodes, as it reduces the likelihood that the carelessness or corruption of a single employee can be exploited to open both locks.

2.3.2 Limitations of threat modelling

No amount of historical research or brainstorming can ensure that all basic threats will be discovered. New technologies or attack strategies can result in new threat scenarios, or may uncover threats that had long been present but not discovered. For example, before the link between disease and germs was established, the threat posed by poor sanitation could neither be understood nor countered. Before the invention of the skateboard, noise pollution was not considered a threat when architects designed concrete steps and ramps. When settling on a set of basic threat scenarios to analyze, it is important to also look at the rate at which new basic threats have been discovered.

Even when all the basic threats are known, there is no way to ensure that all of a node's child scenarios have been discovered. If the scenario of interest has been modelled with existing threat diagrams you may find assurance in knowing these diagrams have stood the test of time. However, there is always the possibility that new attacks will appear.

Finally, countermeasures are not always implemented correctly and even those that are act only to reduce threats, not eliminate them. Keys may be guessed, well screened employees may be bribed, and components that functioned during a million consecutive tests may fail the next time they are used.

Despite the admitted imperfections of the security process, the better threats and safeguards can be understood the better the effectiveness of the process can be measured.

2.4 Do threats and safeguards encompass all security models?

There are many reasons why one might be tempted to reject a definition of a security process consisting of discovery of threat scenarios and placement of safeguards. One might wonder if such a model is general enough to reflect existing and future security theory and practice.

One motivation for rejecting the definition of security in this thesis is the implication that one cannot reach a state of perfect and complete security. Even if perfect countermeasures existed for each threat scenario, there is no way to know if threats haven't been envisioned and thus left unaddressed. The security process, as I have described it, also bears an unfortunate resemblance to the oft-criticized 'discover and patch' approach to security, in which the security process encroaches into the period after a product's release. One might also ask where decades of progress creating formal security models, such as those used to prove statements about cryptographic primitives or network protocols, fit into this framework.

In fact, such formal models have always been based on scenarios. The *threat models* used by cryptographers act to separate those scenarios that their algorithms and protocols protect against from those circumstances that they cannot control. The guarantees of formal models have always been limited to a set of known threat scenarios used to construct the models. If new methods of attack are found outside of those threat scenarios, the system may not be secure despite the assurances of formal methods. In the words of Dorothy Denning [33], "Security models and formal methods do not establish security. Systems are hacked outside the models' assumptions."

Given that security, especially computer security, is a constantly changing field, one might also ask if the definition and approach will stand the test of time. While one cannot anticipate all future events, we can apply this security process to historical examples to see whether it remains timely.

One might consider the security offered by the safeguards of a castle in protecting the kings and nobles of medieval times from their enemies.[6] One can imagine the threat posed by having to fight an advancing line of enemy knights, and how gates and strong lower walls would initially limit the number of forces that the defense would need to face at a time. High upper walls might be added to counter the threat posed by the archers' flying arrows. Still more countermeasures, such as moats, were required as an adversary might tunnel under the castle walls, scale them on a ladder or belfry (a mobile siege tower), or attempt to destroy the walls by slamming them with a battering ram or a large projectile. A scenario-based approach would also remind the king that his castle could not protect him from all possible threats, such as poisoning by his kitchen staff. He'd need a food taster for that.

Castles even provide an example of how new technologies can render insecure those systems that had seemed impenetrable when they were designed. Towards the end of the Middle Ages, the range and size of projectiles fired from trébuchets and cannons increased. Walls could no longer substitute for the combined safeguards of a strong army and powerful weapons. Those that believe that we could completely eliminate the problems of updating (patching) systems if only we improved the design and testing process may also learn from this example.

[6]For a historical description of medieval siege tactics and countermeasures, see *The Medieval Fortress* by Kaufmann and Jurga [66].

2.5 Chapter summary

In this Chapter, I defined *security* in terms of *threat scenarios* and the *safeguards* that are put in place to counter them. I showed how graphs, most commonly in the form of trees, can be used to model the interaction between these threat scenarios and safeguards. I also addressed the limitations of these approaches to threat modelling.

Finally, I argued that the definition of security provided in this chapter is generally applicable, and will be as useful for discussing the security questions of tomorrow as it is for understanding the problems faced by our ancestors.

Chapter 3

Why measuring security is hard

3.1 Security risk

From home users, to firms, to governmental agencies, those of us who rely on our systems to be secure would like to be able to forecast the risk to these systems and the effectiveness of different security strategies in reducing these risks.

From the perspective of a business, security is an investment to be measured in dollars saved as a result of reduced losses from security breaches, or in profits from new ventures that would be too risky to undertake without investments in security. As a result, security modelling often falls under the control of a firm's risk management function.

Government also considers information security, and the effectiveness models that guide decisions, to be a matter best addressed through risk management. The Computer Security Act of 1987 [112] mandates that government agencies should have security plans "commensurate with the risk and magnitude of . . . harm." The guidelines for such a plan take the form of the Risk Management Guide for Information Technology Systems [109] from the National Institute of Standards and Technology

(NIST) which defines risk management as “the process of identifying risk, assessing risk, and taking steps to reduce risk to an acceptable level.” However, these guidelines lack a quantitative method for modelling risk in order to asses it.

The risk management literature includes safeguards among the *responses* to the risk posed by threat scenarios. These responses are avoidance, assumption, limitation, and transference [69, 80, 109]. One *avoids* an optional risk by choosing not to participate in the risky activity, sacrificing the opportunity to benefit from that activity but safeguarding against the risk that the activity could result in a breach. When risk is unavoidable or one chooses to accept the potential losses rather than implement additional safeguards, one is said to *assume* a risk. By buying insurance one may *transfer* the risk of loss to another party. If entering a contract presents a risk, that risk may also be transferred through clauses in the terms of the contract that assign liability to each party. Finally, one may *limit* risk by introducing safeguards that reduce the likelihood of harmful events or that limit the damage caused when such events occur.

While the study of computer security has centered around those safeguards that limit risk, avoidance is now a more commonly accepted security practice. For example, firewalls can be seen as techniques for risk avoidance, as they require an organization to forgo the benefits of open networks, such as ease of access, in order to avoid risk. The choice not to adopt electronic ballots for public elections is also the result of risk avoidance, as is the current push to ship software with lesser used features turned off by default.

From the risk management literature a number of metrics have evolved to measure security risks. The remainder of this section will cover a progression of these metrics, which are summarized in Figure 3.1, and why procedures for quantitatively measuring them have been evasive.

3.1.1 Annual Loss Expected (ALE)

The most common measure for the risk of a harmful event is Annual Loss Expected, or *ALE*, which is the product of the expected yearly rate of occurrence of the event times the expected loss resulting from each occurrence.

$$\text{ALE} = \text{expected rate of loss} \times \text{value of loss}$$

The annual loss expected from all of an organization's operations would be the sum of the expected yearly losses that could result from each threat scenario. Unfortunately, determining accurate inputs to the ALE equation is significantly harder for security threats than natural ones [6, 59].

If a risk manager is calculating the ALE of losses due to a natural threat source, such as an earthquake or degradation (wear and tear) of components, she can forecast the expected rate of future occurrences by using the simplest possible model,

Annual Loss Expected	ALE	(rate of loss) $\times$ (value of loss)
Savings (reduction in ALE)	S	$\text{ALE}_{\text{baseline}} - \text{ALE}_{\text{with new safeguards}}$
Benefit	B	$S +$ (profit from new ventures)
Return On Investment	ROI	$\frac{B}{\text{cost of safeguards}}$
Internal Rate of Return	IRR	$C_0 = \sum_{t=1}^{n} \frac{B_t - C_t}{(1+\text{IRR})^t}$

Figure 3.1: Common metrics used by security risk managers

substituting in the historical rate. This forecast must then be adjusted to account for recent trends. Models can accurately forecast future events using historical data only when the statistical relationships on which they are built remain stationary over time. If the influence of the independent variables on the dependent variable changes, the model's forecast and error estimates will be unreliable. This *stationarity* requirement is difficult to achieve when modelling events caused by strategic adversaries, as acting counter to known models is often the dominant strategy of these adversaries. Unlike nature, strategic adversaries learn to attack a system at its weakest point, improve their skills over time, and thwart attempts to measure their behavior. Even if adversaries did cooperate, historical data for human threats is lacking [76, page 63].

Between 1988 and 1991, the National Institute of Standards and Technology (NIST) held four workshops in hopes of improving risk models for information technology. The dearth of progress is reflected in today's state of the art. In October of 2001, NIST published its Risk Management Guide for Information Technology Systems [109], in which it recommended that event likelihoods be estimated into only three qualitative categories: low, medium, and high. The same three categories are recommended by the OCTAVE$^{\text{SM}}$ approach [2, 3] from Carnegie Mellon's Software Engineering Institute (SEI) and CERT® Coordination Center. Unlike quantitative likelihood estimates, these qualitative categories cannot be used to estimate Annual Loss Expected (ALE) or other quantitative cost metrics. Upon reading the NIST guide, the reader is left to wonder how this process can meet its requirement of enabling senior managers to "use the least-cost approach and implement the most appropriate controls to decrease mission risk." [109, page 27]

Gordon and Loeb [52] use a version of ALE that is modified for situations in which at most one loss will occur. Thus the dollar cost of a loss is multiplied by the likelihood of a loss, rather than the expected frequency of loss used to calculate ALE. They model the probability that a breach will occur as a function of the dollars invested in security. In this theoretical work, they then assume that this security breach probability function is continuously twice differentiable (no discrete investments), that the first derivative is negative (breach becomes less likely as investment increases), and that the second derivative is positive (diseconomies of scale[1]). Their results show upper bounds for optimal levels of security investment against a loss, informing the risk manager of the maximum that should be spent on safeguards if all the assumptions of the model hold. Beyond this result, the technique is not intended to be applied for quantitative risk management decisions, such as which safeguards to chose.

3.1.2 Security savings (S) and benefit (B)

In his doctoral dissertation [59], Kevin Soo Hoo puts aside the problem of estimating how secure any system is and focuses on measuring the benefits of investments in additional safeguards. He contends that the benefits of an investment in safeguards goes beyond the reductions in expected cost of security breaches (decreased losses). In addition, he adds the expected profits from new activities that could not have been profitably undertaken without the added security measures. We call the savings S, and we call the sum of the savings and this new revenue the total benefit, B, of a

[1]Diseconomies of scale means that each dollar invested in security provides a smaller fractional reduction in security breaches than the previous dollar did.

security investment.

Soo Hoo uses an ALE-based methodology to calculate security savings. The amount that can be saved by reducing the rate of successful attacks and damage per successful attack is calculated as the decrease in annual losses that results.

$$S = \text{ALE}_{\text{baseline}} - \text{ALE}_{\text{with safeguards}}$$

Soo Hoo models the effect of a safeguard as causing a fractional reduction, s_i, in ALE.

$$\text{ALE}_{\text{with safeguard } i} = s_i \cdot \text{ALE}_{\text{baseline}} \qquad \text{where } 0 \leq s_i \leq 1$$

To keep things simple, Soo Hoo assumes that a safeguard has the same fractional reduction on ALE regardless of the other safeguards implemented. That is:

$$\text{ALE}_{\text{with safeguards } i,j} = s_i \cdot s_j \cdot \text{ALE}_{\text{baseline}} \qquad \text{for all } i, j$$

There are obvious limitations to the applicability of this model. For one, it overstates the reduction in risk resulting from the use of safeguards that act as substitutes for each other. For example, turning off unused local network services when these services are already inaccessible due to a machine-level ("personal") firewall is likely to have a significantly lower impact on risk than if no firewall were present. However, in Soo Hoo's model, removing services will result in equal fractional risk reductions regardless of whether a firewall is already in place. The model also fails to capture the effects of complimentary safeguards. For example, an investment in an intrusion detection system, which alerts administrators to suspicious network activity, reduces risk only when paired with a procedure for acting on these alerts.

As with previous ALE based approaches, Soo Hoo's leaves open the question of how to forecast the rate at which loss events will occur and how to forecast the reductions in these rates that will result from adding safeguards. Instead, his methodology requires as its input the fractional reduction in security breaches that can be expected from implementing each of the safeguards under consideration. While Soo Hoo cites data from the fourth and fifth Computer Security Institute/FBI Computer Security and Crime Surveys [27, 28], and believes that safeguard effectiveness could be derived from past incident data, he provides no procedure for producing these forecasts. In lieu of a methodology and data source, his analyses rely on safeguard efficacy estimates based on "expert judgements" [59, page 52].

3.1.3 Investment return: ROI and IRR

A metric that is quickly gaining in popularity is return on security investment, also known as security ROI [15, 21, 65, 48, 60]. Assuming that the annual benefit of a security investment will be received not only in the first year, but in all subsequent years, security ROI is defined by Blakley [15] as the amount of this annual benefit over its cost. The benefit is calculated as it was earlier by Soo Hoo [59], by adding the expected cost savings (reduced expected loss) to the new profit expected from ventures that could not have been profitably undertaken without the additional safeguards. The cost is assumed to be incurred immediately.

$$\begin{aligned}
ROI &= \frac{\text{benefit of safeguards}}{\text{cost of safeguards}} \\
&= \frac{(\text{savings from safeguards}) + (\text{profit from new ventures})}{\text{cost of safeguards}} \\
&= \frac{\text{ALE}_{\text{baseline}} - \text{ALE}_{\text{with safeguards}} + (\text{profit from new ventures})}{\text{cost of safeguards}}
\end{aligned}$$

Gordon and Loeb [53] advocate that firms should discard the above ROI formula and instead use the Internal Rate of Return (IRR, also known as the economic rate of return) because IRR incorporates discounted cash flows for investments that have different costs and benefits in different years. If C_0 is the initial cost of an investment, and C_t and B_t are the respective costs and benefits in year t, one can solve for the IRR using the following equation:

$$C_0 = \sum_{t=1}^{n} \frac{B_t - C_t}{(1 + \text{IRR})^t}$$

Though they contend that IRR is superior to ROI, Gordon and Loeb also warn that even the correct rate of return can be used inappropriately. They caution that rates of return not be used when comparing two investments, as an investment can have a greater net benefit but lesser rate of return. If enough cash is available to invest in either of the options, but a firm can invest in only one (the options are substitutes or mutually exclusive for other reasons), it would be less profitable to choose the investment with the higher rate of return over that with the greater net benefit.

Rate of return is useful for determining whether a security investment is justified given the investor's cost of capital. Rates of return have been growing in popularity because the metric is familiar to CFOs and others who control corporate budgets and approve expenditures.

Unfortunately, calculating either of these rates of return requires that one first calculate security benefit (B or B_t) which in turn requires one to calculate ALE. Like our other metrics, security ROI and IRR forecasts are only as accurate as the forecasts of loss event frequencies on which they rely and today these forecasts use best guesses rather than quantitative models.

3.1.4 The elusiveness of quantitative models

The lack of quantitative studies in computer security is attributed, in part, to a lack of data on variables that can be shown to influence security risk.

Information collected by CERT provides data about security incidents, but not about aspects of the larger environment, such as properties of systems that were not victims of security incidents. Thus, these data are similar to what criminologists would call *victim studies*. Broader information is obtained by the CSI/FBI survey, but the survey's questions were not crafted to fit the needs of regression studies [82], especially not the kind that seek to model the effects of different system and safeguard choices. Both the CERT and CSI/FBI data sources suffer from reporting omissions, as many firms decide not to report security breaches in fear that they might lose customers if breach reports were to be leaked to the public.

Surprisingly, studies using regression models have been successful in forecasting security risk outside of software security in domains such as home security, as we will see in Chapter 7. These studies are successful because they measure the risk to systems with a homogenous architecture (homes), with fairly homogenous safeguards (deadbolts, alarms), from a threat that remains stationary over time (burglary). Most importantly, we will see that the adversary at the source of the home burglary threat is one that is deterred by the risks to his person of capture or harm during the attack, and that the factors that indicate this personal risk are measurable. Many of the adversaries faced by computer systems attack from a great distance, and are thus deterred more by the strength of the system than the personal risk resulting from the attack. Unlike indicators of personal risk, factors that affect the difficulty of attacking a system (or the strength of a system to make attack difficult) have been difficult to measure.

Thus, regression studies in network security that excluded measures of security strength would suffer from omitted variable bias. Because network attackers are often deterred by the difficulty of attacking systems, and not risk, one or more indicators of security strength must be included as independent variables if a model's forecasts are to be accurate enough to be useful.

In lieu of a measure of security strength, the best we can do is to substitute independent variables that are believed to be correlated with it. For example, in software, code complexity is often considered to be negatively correlated with security [95], though complexity is itself difficult to measure.[2] If one found enough independent variables correlated with security strength, such as complexity, security budget, testing period, the recent rate of vulnerabilities reported, and the version

[2]Complexity is often measured in terms of code size. This may not be the right measure as an increase in the size of source code due to comments may reduce, and not increase, the complexity. An increase in the size of compiled code may be the result of an inclusion of bounds checks, which would also be expected to make the software more secure.

number, then one might be able to estimate security strength well enough to then forecast security risks. Even if these results were at first accurate, the moment such a security strength forecasting methodology was published its accuracy would begin to quickly degrade. Software vendors would use the least cost approach available to increase their software's security strength scores regardless of whether these actions actually increased the actual strength of the software. Reports of testing budgets and testing periods would be inflated to make testing appear more rigorous and code would be condensed to make it appear less complex, all without any change to the strength of the actual product.

One might attempt a regression analysis on a single system configuration in order to isolate security strength as a constant that need not be measured. If the systems studied do not change, one might hope that their security strength would also remain constant. However, security strength changes the moment a new vulnerability in one of these systems is discovered. If that system were updated to repair the vulnerability then the strength would change yet again. The approach of using a single set of systems and versions is doomed because the systems themselves are not stationary. Thus, historical data becomes unable to predict future security risks should a new vulnerability be found or should the system be updated. On the other hand, if we know the strength of a system and the incentive that would lead adversaries to look for vulnerabilities, we could look to historical data to tell us how often new vulnerabilities have been found and attacks staged in similar situations in the past. In the end, there's no substitute for a direct means to measure security strength.

An alternative approach to calculating the likelihood of a security breach might appear to be available in the tools of fault tree analysis, using the structures introduced in Chapter 2. The goal of this analysis is to determine the likelihood of the basic threat event at the root of the tree. The technique is based on the observation that for any parent node, the probability that the event will occur is a function of the probabilities that the child nodes will occur. Assume that the event represented by a parent has two children, c_1 and c_2, which occur with probabilities $P(c_1)$ and $P(c_2)$ respectively, either of which trigger the parent event. The laws of probability tell us that the probability that the parent event will occur is $P(c_1 \vee c_2) = P(c_1) + P(\bar{c}_1)P(c_2|\bar{c}_1)$, which will simplify to $P(c_1) + (1 - P(c_1))\, P(c_2)$ if the two child events are independent. If both child events must occur to trigger the parent event then the likelihood of the parent event is $P(c_1 \wedge c_2) = P(c_1) \times P(c_2|c_1)$, which simplifies to $P(c_1) \times P(c_2)$ if the child events are independent.

If the parent event has more than two children, the analysis can be performed iteratively over all the children of the parent so long as the laws of precedence are respected. In other words:

$$\begin{aligned} P\left(c_1 \vee (c_2 \wedge c_3)\right) &= P(c_1) + P(\bar{c}_1)P(c_2 \wedge c_3|\bar{c}_1) \\ &= P(c_1) + P(\bar{c}_1)P(c_2|\bar{c}_1)P(c_3|c_2 \wedge \bar{c}_1) \end{aligned}$$

One need also account for the possibility that one has not anticipated a threat that would result in an additional child node. The longer a given portion of a tree has been in use, the less likely it is that new child nodes will be discovered. One can use bayesian stochastic models, based on the previous rate of discovery of child nodes for the node in question, to estimate the likelihood that a new child node will be discovered over a given time period. In calculating the likelihood of a security breach at a given node, one can then combine the likelihood of a breach at known child nodes with the likelihood of a breach at a previously undiscovered child node.

Unfortunately, this tree-based analysis does not solve the problem of estimating probabilities, but rather pushes the problem down to the leaf nodes. The likelihood of a breach at any given node cannot be calculated until all the leaf nodes below it are calculated. Thus, the analysis does not eliminate the need for regression analysis or other means of obtaining security risk statistics, but instead enables us to move the task of calculating event probabilities down to the level of those events represented by the leaf nodes.

3.2 Security strength

Safeguards have long been measured by how difficult they are to circumvent. Metrics of security strength attempt to quantify the time, effort, and other resources that must be consumed in order to bypass a system's safeguards. Whereas metrics of security risk are useful from the perspective of those the safeguards are intended to defend, strength metrics are intended to be viewed when taking the perspective of the adversary. For example, strength metrics might tell us how a castle's safeguards may increase the number of soldiers, equipment, time, or motivation required to lay a siege with a reasonable expectation of success.

The history of security strength metrics is perhaps best understood through development of ciphers and cryptography. A cipher is a set of rules for encoding and decoding messages so as to ensure their confidentiality and integrity. A good cipher should require little effort to encode or decode a message with a key. The security of a cipher rests on how difficult it is to correctly decode a message if one does not possess the decryption key. Threat scenarios against ciphers are often differentiated by the amount and type of information available to the adversary. For example, a known text attack is a threat scenario, on a symmetric cipher, in which the adversary analyzes both the plain-text and enciphered copies of one or more messages in order to derive the decryption key.

Many have made the mistake of assuming that a complex cipher, that appears impenetrable to those who use it, will confound all adversaries who try to decode it. One early example of such a mistake took place in 1586 when Mary Queen of Scots, imprisoned by her cousin Queen Elizabeth of England, received an offer of rescue enciphered in a letter.[3] Rather than accept indefinite imprisonment, Mary chose to trust the strength of her cipher and risked a reply in the affirmative. This was an act of treason that could lead to almost certain death – if the letter fell into the wrong hands and could be deciphered. The risk undertaken by Mary was very much a function of the strength of the cipher in resisting cryptanalysis by Queen Elizabeth's agents. Those agents, empowered with the resources of a ruling monarch, could dedicate more time and effort to breaking the cipher than Mary's agents had been able to. As Mary was a threat to Elizabeth's thrown, Elizabeth also had the motivation to expend these resources. In the end, Elizabeth's codebreakers were able to decipher the message using a laborious technique known as frequency analysis. Mary was sent to the gallows.

[3]The cryptographic history is from "The Code Book" [101] by Simon Singh, who in turn cites [45, 102, 107].

Since the mechanisms of a cipher must be known if the cipher is to be fully examined, a widely tested cipher cannot be a very well kept secret. Because of the clear danger in relying on poorly tested ciphers, Auguste Kerckhoffs proposed in his 1883 book, *La cryptographie militaire*, that the strength of a cipher should rest only on keeping the key secret, and not on the secrecy of the design of the cipher or cipher machine. This became known as Kerckhoffs' law. Even with this advance, ciphers still came in two strengths – those that had been broken and those that had not. The only way to test the strength of a cipher was to publish encoded messages and challenge others to decode them.

In the first half of the twentieth century, it was the Germans who failed to learn from Mary's example and failed to heed Kerckhoffs' law. In 1926 they introduced the first version of their Enigma cipher and the Enigma machine used to encipher and decipher messages. The machine and its cipher were complex, and their designs were a German secret. However, by the 1930s, devices known as a *bombes*, invented by Marian Rejewski of Poland, were being used to decipher messages and determine the key used for transmission [101]. Like Queen Elizabeth, the Polish had ample motivation to break the cipher and were willing to dedicate more resources than the Germans who relied on the code to protect their messages. Why, after all, would the Germans waste resources testing such a seemingly impenetrable machine?

While the Germans increased the complexity of the Enigma machine and its cipher for World War II, the tremendous resources of the allied forces at England's Bletchley Park were able to decipher the German messages for much of the war [58, 101]. The intelligence obtained from German messages allowed the allies to avoid submarine attack, land at Normandy with a minimum of casualties, and frustrate Rommel's forces in Africa [11]. While Rommel suspected that his messages were being read by the allies, his superiors refused to believe that such a complex code could be broken [10]. The lesson was finally learned that security is never certain, especially if your adversary is willing to expend more resources to test the security of your safeguards than you are.

There remain no unbreakable ciphers outside of the one-time pad, a symmetric cipher proven by Shannon to be information theoretically secure [100], but that requires impractically large keys.[4] Rather, the developers of fixed-length key ciphers test strength by challenging themselves and others to try to find a means of cryptanalysis that would break the cipher. The problem faced by those who employ ciphers is how to determine when enough effort has been expended in attempting to break the cipher before it can be deemed suitable for use. No matter how many resources are expended in attempts to break a cipher, there is always the possibility that the enemy can exert greater effort or have better fortune in picking the right strategy to attack the cipher.

In the 1970s, a loophole was discovered through which the problem of dedicating testing resources to each new cipher could be bypassed. The loophole emerged with the advent of public key cryptography, in which the key used to encrypt a message is public knowledge and a matching private key is used to decrypt messages. As

[4]One time pads require one bit of randomly generated key material for each bit of data that might be sent between the communicating parties before they next meet and can exchange new key material. Thus, while one time pads are information theoretically secure against the scenario in which an adversary attempts to derive the encryption key, they are vulnerable to attacks that target key generation, key distribution, or the secure storage of keys.

first publicly described by Whitfield Diffie and Martin Hellman [34][5], such schemes required the use of a trap door function, $f(x)$, for which it was prohibitively time consuming or expensive to compute its inverse $f^{-1}(x)$. However, with knowledge of the secret key (the trap door), a fast inverse function $f^{-1}(x)$ can be constructed.

The first such trapdoor function and resulting public key encryption scheme was the RSA scheme invented by Ron Rivest, Adi Shamir, and Leonard Adleman [83] in 1978.[6] The public key contained a composite number that was the product of two large prime numbers (factors). The private key could be constructed if one knew the prime factors. If the adversary could factor the composite number in the key, he could recreate the private key and the cipher would be broken. The inventors of the RSA scheme could not rule out the possibility that it would be possible to decrypt the message without knowing the private key and without the ability to factor the composite in the public key. Still, RSA remains the most commonly used public key cipher.

In 1979, Michael Rabin introduced his public key encryption scheme which also used a composite of two large primes as the public key, and for which the factorization of the composite served as the private key [81]. Rabin went further and proved that if one could build a machine to decrypt randomly chosen messages, one could use the same machine to factor composites. Thus, if used within these constraints, breaking Rabin's cryptosystem was proven to be as hard as factoring the composite that composed the public key. The problem of efficiently factoring composites is one

[5]Public key cryptography was first conceived in 1970 as 'non-secret encryption' by James Ellis [38, 39] as part of his classified research within the British Communications-Electronics Security Group (CESG).

[6]Once again, this discovery was preceded by classified work at the British CESG, this time by Clifford Cocks [26, 39].

that has intrigued mathematicians, and later computer scientists, for ages. Thus, the security strength of Rabin's encryption scheme had already been subject to decades of testing by some of the world's most talented, highly motivated, individuals. Shafi Goldwasser and Sylvio Micali [51] would later devise an encryption algorithm that reduced the security of all encrypted messages (not just randomly generated ones) to the strength of long studied computational problems. In addition to the factoring problem, they added the assumption that it is difficult to distinguish quadratic residuosity modulo a composite with unknown factors [50].

Security measurement benefited from the advent of cryptosystems that were established to be as difficult to break as a known computational problem. One could estimate the amount of time required to break the system with current technology and knowledge by using computational complexity analysis, or even by simulating a portion of the algorithm. Thus, one could answer the question of how much money your adversary would need to spend on equipment and how long he would have to work in order to break your cryptosystem using existing technology and algorithms.

One could also crudely estimate the effect of advancing technology in reducing the cost of future attacks on the cryptosystem. Moore's law, which predicts exponential decline in computing costs over time, could be used to model the decline in the cost of computation. Crude models were also created to predict advances in factoring algorithms (critical to many cryptosystems) based on past progress in the search for faster algorithms. The necessity of these models demonstrates that even strength results in cryptography, a field known for formal and precise results, depend on uncertain factors such as human innovation in overcoming unsolved problems. Unless scientists prove that there exist problems that are significantly more expensive to solve than to verify (such as by proving $P \neq NP$)[7], the strength of cryptosystems will continue to depend, in part, on how well the computational problems on which they are based have been tested. Regardless, cryptographic models will remain subject to new forms of attack outside the models on which they were built, as was discovered with the advent of timing and other side channel attacks.

While cryptography has advanced greatly through the reduction of cryptographic problems to known computational problems, few other security safeguards can benefit from this approach. As a result, the progress in the theory of cryptography has far outpaced that of most other security research. Public key cryptography has also far outpaced the development of the public key infrastructure required to support it. Almost a quarter of a century after the invention of RSA, almost all the systems that use it rely on infrastructure for certifying the identity of public key holders that experts consider unacceptably weak [40, 55].

Outside of cryptography, estimating the strength of a safeguard still requires extensive examination. It is a common lament that testing cannot prove the absence of any security flaws, and that at best it can probabilistically show the expected time or effort required to find the next flaw. However, given that any measure of security strength in real-world systems will contain uncertainty, testing can still play an important role in reducing and measuring this uncertainty.

If the strength of a system depends on how well the system has been tested, how does one measure it? Some security experts track the security strength of software

[7]That is to say, proving that there exist problems which take exponentially more time to solve than it takes to verify that their solution is correct.

by watching the rate at which flaws are discovered and reported, look at which features are included, or examine the reputation and size of the manufacturer. However, little evidence has been published to support any of these metrics. The problem is even more complex for systems with human components (organizations) as people introduce a level of nondeterminism into the system that makes security strength impossible to measure with the traditional computer science tools of logic and computational complexity alone.

3.3 Chapter summary

In this chapter we reviewed a number of security risk metrics: Annual Loss Expected (ALE), security savings (S), security benefit (B), and Return On Investment (ROI and IRR). While the values of these metrics may be calculated after the fact (ex-post) from historical data, forecasting future values has proved problematic. The crux of the problem is that the frequency and severity of incidents has been changing over time, and is not likely to remain stationary given the flux inherent to an environment driven by advancing technology.

We also reviewed the problems inherent in measuring security strength. In essence, problems of security strength come down to measuring how hard it is for the adversary to violate your security requirements. Measuring time, effort, and other resources is the domain of economics. In the following chapter, I'll explain how, using economic principles, markets can be used to to the measure security strength of software systems.

Chapter 4

Measuring the security strength of software

4.1 Security strength

For any given threat scenario, a system is only as strong as it is difficult for an adversary to cause the scenario to be realized. Difficulty not only implies the adversary's effort but also his or her need to obtain other resources, such as time and equipment. The sum of the resource costs incurred in breaking through safeguards to realize a threat scenario is called the *cost-to-break* [90].

We saw in Chapter 3 that, when the security of a system rests on a well studied computational problem, cost-to-break is often measured in units of computation. The amount of computation required to accomplish that adversary's goals is rarely known ahead of time, but is instead probabilistic. For example, when guessing a password or using a randomized factoring algorithm against a public key cryptosystem, there is an extremely small, but non-zero, probability that the adversary will succeed immediately based on lucky guesswork. Because the amount of computation required to breach security is probabilistic, it is the expected cost-to-break that must be measured.

Solving computational problems is only one of the many ways to breach the security of real, more complicated, systems, especially those systems composed of people with motives and flaws of their own. While the cost of equipment, effort, and time may not be measurable with the same level of formal precision as the cost of computation, we need not abandon measurement. Measuring the current cost of equipment may be as simple as shopping around to find its market price. Estimating the future cost of equipment may be nearly as straightforward. The cost of computation has declined at a rate very close to that predicted by Moore's law. Like equipment, time and effort can also be measured in units of dollars or other currency. If the amount of time and skill required to breach security is known, we can use labor market statistics to estimate its dollar cost.

One of the difficulties in measuring cost-to-break is that it is a function of your adversaries' costs, not your own. It's impossible to research the skill of every potential adversary, the value they place on their time, and the other resource costs they require to realize a threat scenario. What's more, the adversaries themselves are not likely to know how much time and resources they will need to expend to breach security. This is especially true if breaching the security of a system requires that the adversary find a vulnerability to exploit.

To understand why even the adversary may not be able to measure his own costs, assume that finding a vulnerability is an essential step in breaching the security of a system. There are a series of tasks the adversary can perform to look for vulnerabilities, from inspecting code to writing and executing tests. To maximize his

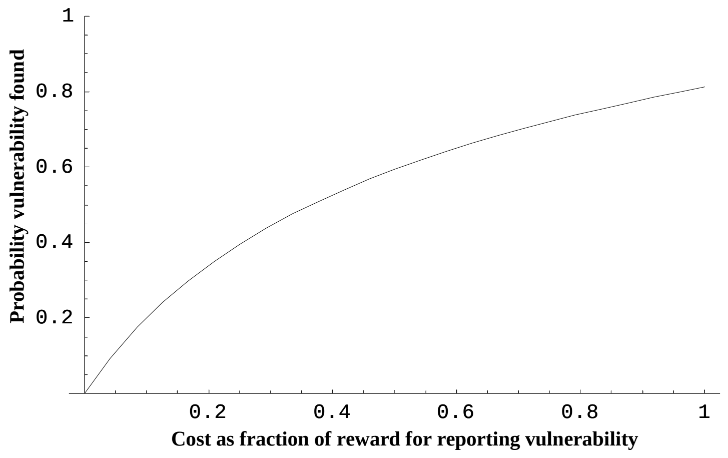

Figure 4.1: A cumulative probability distribution that represents an individual's beliefs of the likelihood that he or she can find a vulnerability in a system (the y axis) for the cost represented by the x axis.

productivity, the adversary will start with the tasks that have the greatest chance of success in finding a vulnerability per unit cost. Diminishing expected returns result because the tasks with the highest expected profitability are executed first. We can see in Figure 4.1 that individuals may perceive cost-to-break not as a single value, but a cumulative probability distribution. The chance of success in finding a vulnerability increases with total investment (the first derivative is positive), but the chance of success for each additional dollar invested is smaller than for the previous dollar (the second derivative is negative).

Economically rational individuals will only perform tasks so long as their expected return is greater than their expected cost. If an individual believes a vulnerability is worth r dollars, the cost of task i will be c_i, and the probability that task i will result in the discovery of a vulnerability is p_i, then he will perform the task only when $c_i \leq p_i \cdot r$, or equivalently when $\frac{c_i}{p_i} \leq r$. In fact, while a risk-neutral individual will continue to perform tasks when $\frac{c_i}{p_i} < r$ and be indifferent to performing tasks when $\frac{c_i}{p_i} = r$, a risk-averse individual will not search for vulnerabilities when $\frac{c_i}{p_i} > r - \epsilon$ for some positive measure of risk aversion ϵ.

There is no reason to believe an individual will expend the same amount of resources to find a vulnerability as he would if he knew the true cost-to-break. Perceptions may be updated after one task is completed and before choosing to start the next task, though updating estimates of c_i and p_i cannot guarantee that they will be accurate. It is the perceived expectation of the cost-to-break, not the true cost, that determines how many resources adversaries will be willing to spend to find vulnerabilities in a system. Thus, a metric of security strength that incorporates perceptions of cost-to-break may not only be as valid as a measure that includes only true costs, it may be more valid.

Security is not the only field in which perception can often trump reality. The theory of investment tells us the true value of a firm is the net present value of the income stream that the firm will return to its investors. However, a firm with a higher perceived value will be more likely to obtain debt financing, make deals, and retain employees. Such a company will thus be more likely to succeed, and return money to its investors, than a firm that is identical except for perceptions. This is why it is impossible to separate perceptions from true value. The true value of a firm can never be known so long as it is a going concern with an uncertain future. The perceived value of a publicly traded firm is usually readily available and falls within a tight range between the market bid and ask price.

It should not be surprising that a market-based approach could also be used for security. The first such proposal was presented by Jean Camp and Catherine Wolfram [23] in the context of securing individual computers from being used in denial of service attacks. In their work, insecurity in networked computers is treated as an externality much like pollution in power plants.

A market approach is also effective for measuring the strength of software [90]. This will be the topic of the remainder of the chapter.

4.2 Why measure software systems?

Quantifying the security of physical components and systems is more difficult than doing so for software. No two physical components can ever be shown to be truly identical. There is always a risk that defects will occur in a replica that did not occur in the original.

I focus on software systems because they are more suitable for security strength measurement. The ease with which software systems may be copied increases the ease of measurement, as well as the value of measurement. Software, like all information goods, has a negligible copy cost and each copy maintains the exact same qualities as the original. Verifying that two information goods are identical replicas is also trivial. After a program has undergone testing or security strength measurement, you can be assured that your copy is indeed a replica of the software that was measured. The ability to amortize the costs of measuring security strength over a large number of copies is essential to making secure software affordable.

It is also possible to verify that a software program hasn't changed since you last used it. Unlike mechanical systems or systems with human components, software does not degrade or otherwise change over time. Changes, such as upgrades and patches, may be controlled by the system's administrator. While the security strength of a system changes when new flaws and vulnerabilities arise, the software itself can be verified as remaining unchanged.

Measuring software alone implies system boundaries that delineate human components as external to the system, or part of the larger system in which the software is deployed. To make this delineation we must define how the system interacts with the outside world, including its users. We must specify what each type of user is and isn't allowed to do. The value of defining these boundaries is that we can then measure security strength of the software system exclusive of the people and external systems that will differ at each location in which it is installed. Because software systems may be more formally specified than organizational systems, the threat scenarios they face can also be more clearly defined.

The quantity and wide availability of software tools that exploit vulnerabilities to attack systems, which often appear very soon after vulnerabilities are publicized, implies that the cost-to-break of systems with known vulnerabilities is quite low. It is more interesting to estimate the security strength of systems in which there are no known vulnerabilities than to measure the negligible security strength for systems that do have known vulnerabilities. In order to realize a threat scenario against a system with no known vulnerabilities, a new vulnerability must be found and a technique to exploit the vulnerability to breach security (known simply as an *exploit*) must be developed. The cost-to-break is thus dominated by the cost to find a vulnerability and create an exploit. Since primitive exploits are often required in order to prove the existence of a vulnerability, these costs may be considered together as the cost of finding a vulnerability.[1]

An adversary intent on breaching the security of a computer system may search for a vulnerability himself or he may exploit the labor market to pay someone else to do it. If we assume the worst, the individual with the lowest cost of finding and exploiting a vulnerability is the adversary with the most to gain from this knowledge, or someone that this adversary trusts enough to do business with.

4.3 Pricing vulnerabilities

The market price to find and report a vulnerability, or *MPV*, estimates the cost of finding a vulnerability in a software system as measured by a market in which anyone, including those we might consider to be our adversaries, can participate. The MPV bid price is the highest price offered to buy a previously unreported vulnerability. The MPV ask price is the lowest price at which a seller is asking to report a vulnerability. Transactions occur when the bid price reaches the ask price. The MPV at all other times is a range bounded at the bottom by the bid price and at the top by the ask price.

For any given threat scenario, a firm can establish an MPV bid price for vulnerabilities that could be exploited by an adversary to realize that threat. One way to establish this bid price is to make a public offer to pay a reward to anyone who

[1]It is possible for the cost of exploiting a vulnerability to exceed the cost of discovering it. An extreme example of such a vulnerability is a back-door guarded by public key cryptography. That is, a vulnerability intentionally placed in the code in order to grant access to anyone who knows a secret key. While such a vulnerability may be easy to find, writing an exploit without knowledge of the secret key requires one to accomplish the monumental task of breaking the public key cryptosystem.

reports a new vulnerability of this type. If the reward offer is common knowledge to those who have the skills to find vulnerabilities then the test will lead to one of the following three states:

1. Someone has accepted the offer and reported a vulnerability.
2. One or more vulnerabilities have been found but not reported.
3. No vulnerabilities have been found.

For sake of example and without loss of generality, consider what happens if the reporting reward (MPV bid price) is set at \$10,000.

In state (1) we have failed to establish a lower bound on the security strength of the system. A vulnerability was found and reported for less than \$10,000, and there's no reason to believe that another vulnerability wouldn't be found for the same, or even a lesser, price. The MPV bid price is no longer bounded as the reward has been claimed. To restore the bound, the vulnerability must be repaired and a new reward offered.

While it is impossible to know that no vulnerabilities have been found, and establish that one is in state (3) and not state (2), the lower bound on market price of a vulnerability is the same in both states.

The only individuals who can distinguish state (2) from state (3) are those who discover one or more vulnerabilities (or believed they could find one for the reward price) but decide not to report them. State (2) may occur when an individual has a more lucrative use for the vulnerability information. However, by not reporting the vulnerability and declining the reward one pays an opportunity cost of \$10,000.

In other words, $10,000 is sacrificed by anyone who could report a vulnerability but decided not to and thus $10,000 remains a valid lower bound on market price of a vulnerability.

In state (3) the system remains secure despite an offer of $10,000 for a means to break it. So long as all potential adversaries (and those who would find vulnerabilities on behalf of those adversaries) are aware of the offer, $10,000 is now a lower bound on the price at which one would find and report a vulnerability if offered the opportunity.

Making the existence and amount of the reporting reward common knowledge among all those who might find vulnerabilities may appear to be a tall order. Solutions include exchanges similar to stock exchanges. After all, while the current asking price of a share of IBM may not be common knowledge, the means for anyone interested in transacting in IBM shares to get the price and make a trade is. Just as press releases and analyst reports give investors the information they need to trade in equities, similar intelligence sources can guide security testers to find the best systems for them to investigate. In lieu of a full market, reward postings from a small set of security firms that administer reward offers may be sufficient to establish a market given the size of the security research community.

Given the low price of personal computers, and their availability to those with the skills to test software for security vulnerabilities, equipment resource costs for testing software systems should be negligible in most cases. There will always be exceptions, such as software that requires a large number of CPUs, extremely high network bandwidth, expensive peripherals, or data which the software developer cannot distribute to testers without violating another party's copyright.

One set of costs of attack that are not accounted for by the market price of a vulnerability are the personal risks the adversary undertakes by staging the attack. These personal risks, which may include reputation damage, personal harm, or incarceration, are separate from security strength because they can't be measured in the same way.

4.4 Precedent for vulnerability discovery rewards

Recent history already contains a number of examples where firms have offered rewards to those who find means of breaching the security one of the firm's products. One of the longest running and most reputable set of challenges is run by RSA Security.

While providing no formal metrics, RSA Security [87] contends that one of the goals of its \$10,000 Secret-Key Challenge is to "quantify the security offered by. . . ciphers with keys of various sizes." The threat scenario they address is made abundantly clear.

> For each contest, the unknown plaintext message is preceded by three known blocks of text that contain the 24-character phrase "The unknown message is:". While the mystery text that follows will clearly be known to a few employees of RSA Security, the secret key itself used for the encryption was generated at random and never revealed to the challenge administrators. The goal of each contest is for participants to recover the secret randomly-generated key that was used in the encryption.

RSA Security also offers a factoring challenge [84, 86]. The problem of factoring is chosen because data encrypted with the RSA algorithm may be decrypted by anyone who can factor the composite contained in the public key. They contend this challenge is helpful "in choosing suitable key lengths for an appropriate level of security." Rewards range from \$10,000 for factoring a 576-bit composite to \$200,000 for factoring a 2048-bit composite [85].

The RSA factoring challenges only address one threat scenario faced by users of the RSA cryptosystems. There is no proof that the RSA cryptosystem is as difficult to break as factoring composite keys. Threat scenarios that bypass factoring to break the cryptosystem are not addressed by the challenge. Threat scenarios in which keys are not generated randomly, or in which the RSA cipher fails to provide security when used under certain conditions, are not addressed. For example, Don Coppersmith [30] has shown messages encoded multiple times for multiple receivers may be decoded if the receivers use RSA public keys in which the publicly known exponent is a small number. This vulnerability cannot be reported for a reward under the rules of the RSA factoring contest.

To RSA's credit, the rules for both of their contests are clear and well defined, and evidence of past pay-outs establish the credibility that the reward offer will be honored. The publicity these contests have received ensure that a large fraction of the individuals with the potential to succeed in the challenge know about it.

Unfortunately, there is no centralized clearinghouse for these challenges and so the costs of discovering the contests, as well as reading a non-standardized set of contest rules, still pose significant barriers to participation.

Finally, the rewards offered by RSA Security, while significant, are not likely to spur a great deal of research. A criminal armed with the ability to factor RSA public keys could make millions, if not billions, of dollars with that knowledge. The fame, and security consulting fortunes, destined for anyone who finds a way to factor 2048-

bit composites far outweigh the value of the $200,000 reward.

Other security challenges have made the flaws of RSA's contests seem small. Perhaps the most notorious was run by the Secure Digital Music Initiative (SDMI) to test the integrity of their music watermarking schemes. The challenge was launched by the release of an *Open Letter to the Digital Community* [25] from SDMI.

> The Secure Digital Music Initiative is a multi-industry initiative working to develop a secure framework for the digital distribution of music. SDMI protected content will be embedded with an inaudible, robust watermark or use other technology that is designed to prevent the unauthorized copying, sharing, and use of digital music.
>
> We are now in the process of testing the technologies that will allow these protections. The proposed technologies must pass several stringent tests: they must be inaudible, robust, and run efficiently on various platforms, including PCs. They should also be tested by you.
>
> So here's the invitation: Attack the proposed technologies. Crack them.
>
> By successfully breaking the SDMI protected content, you will play a role in determining what technology SDMI will adopt. And there is something more in it for you, too. If you can remove the watermark or defeat the other technology on our proposed copyright protection system, you may earn up to $10,000.

Despite the fact that the challenge was far from common knowledge, the testing period was less than a month, and the reward was small, the challenge ended with a clear result. All four of the watermarking technologies were broken by a team of academic researchers from Princeton and Rice Universities [32]. In order to publish their results the researchers turned down the reward, which was conditioned on the signing of a confidentiality agreement that prohibited any discussion of their findings with the public [31].

When SDMI and the RIAA discovered that the researchers intended to publish their results, they threatened a lawsuit [78]. While a counter-suit filed on behalf of the researchers by the Electronic Frontier Foundation (EFF) failed to ensure the right to future publication [37], an agreement was eventually reached allowing the researchers to publish, but only with permission of SDMI. Unless researchers can be convinced that finding and reporting vulnerabilities will not lead to legal action, challenges will exclude many potential participants.

In *Secrets and Lies* [95], Bruce Schneier says that challenges "if implemented correctly, can provide useful information and reward particular areas of research [and] can help find flaws and correct them." He shows concern that challenges can never end with a positive security result, and that rules are rarely enforced fairly. Assuming that challenges must terminate, Schneier concludes that they do not yield useful metrics to judge security. However, challenges can provide meaningful security assurances so long as they do not terminate and the reward continues to be offered. It is the lack of known vulnerabilities combined with the offer of the reward for future vulnerabilities through which the system's developer signals the security strength of the system.

Schneier is correct that potential participants will opt out of challenges if they doubt that rules will be fairly enforced, or even if the rules take too long to understand. RSA Security spent time and money developing simple and fair rules. They have also paid out rewards which has helped build a reputation.

Unlike RSA, many organizations cannot afford the time to do this. Others have reputations that will be forever tarnished from lawsuits they filed against those who have found vulnerabilities in the past. Such organizations require another means of establishing credibility for the rewards they offer. This opens up a business opportunity for middle-men, or third parties, to play a part in the transaction. These third parties could create simple rules that remain fairly consistent for all the challenges they host. If only small changes are made for different challenges, and these are highlighted, the transaction cost of reading rules can be addressed. Because these businesses would rely upon their reputation within the security community and with software vendors, they would have strong incentives to pay out rewards in a fair and just manner. If the firm fails to pay out rewards that are deserved, future rewards offered by the firm would not attract a significant fraction of potential testers and the validity of the entire process, and its results, will be questioned. If a firm pays out when it should not, software vendors will cease to hire them.

4.5 Chapter summary

The strength of a system's security is a measure of the expected cost to breach that security. Software is efficient to test because it is an information good that is inexpensive to replicate and because one can compare the integrity of its replicas. I have shown how a market model can be used to measure the labor and other resources to breach a system's security.

Some indicators of the price at which vulnerabilities can be bought and sold already exist. Security consulting firms such as @stake sell vulnerability analysis as a service. By dividing the number of vulnerabilities found by the price paid it is possible to determine the per-vulnerability cost firms are paying to identify vulnerabilities.

Firms such as iDEFENSE offer rewards to those who report vulnerabilities in other's products, selling the right to early notification to their customer base. Other firms, such as RSA, have offered rewards for the discovery of vulnerabilities in their own systems. While the exchange of rewards for vulnerability reports has not always been successful in establishing security strength, those that most genuinely embraced the open market model have had the greatest success.

Chapter 5

Differentiating software products by security strength

A market for lemons is one in which consumers cannot determine products of quality from defective goods, or *lemons* [1]. In such a market, a producer's dominant strategy is to sell lemons, as they are less expensive to manufacturer (or obtain), and so consumers must assume that any product offered for sale is a lemon. As a result, the price consumers will be willing to pay for a product is calculated under the assumption that the product they will receive will be a lemon. Firms that invest in producing high quality products will then be driven out of the market by those that produce lemons at lower cost. Ross Anderson [9] and others have called the market for secure software a market for lemons. Without metrics through which a consumer can differentiate the more secure products from the less secure ones, there has been little incentive for software producers to increase the security of the products they sell.

To complicate the problem of differentiating software based on its security, the software product that provides the strongest security in deterring one threat scenario may be the weakest in deterring a different threat scenario. Security strength can only be measured and compared when the security requirements, and resulting threat scenarios that constitute security failures, are made explicit.

How do you compare your product's security strength to that of a competitor if your competitor is not willing to offer a reward for vulnerabilities? One approach is to explain to the consumer that, if the reward for finding a vulnerability in your competitor's software that could cause a threat scenario to be realized is \$0, then the lower bound (bid price) of the market price of a vulnerability that could be exploited to carry out that scenario is also \$0. However, comparing two lower bounds does not allow you to say with certainty that your system is more secure. Proving security superiority requires that you place an upper bound on the market price of a vulnerability in the competing product below that of the lower bound of your product's MPV.

One approach to establishing an upper bound on the security of your competitor's software is to find (or buy) a vulnerability in that software. You can then offer the vulnerability at the same price it cost you to obtain it, or any higher price. So long as the vulnerability is not fixed by your competitor, and the vulnerability information is offered to anyone who will pay for it, your asking price represents an upper bound on the product's MPV. Anyone can buy a vulnerability in the system from you at the price you are offering, regardless of whether their intent is benign or malicious. So long as the vulnerability in your competitor's software is easily exploitable and has an asking price lower than the bid price for a vulnerability in your software, you can claim that your software is the stronger of the two systems against the threat scenario in question.

An Instructive Fictional Example

Three computer manufacturers who are dissatisfied with the state of open-source operating system security form a trade group, SBU, to promote a 'Secure' variant of Berkeley Unix. This new operating system is to be targeted to customers seeking a secure network file system and the default configuration supports this application. The operating system must also allow administrative access via `ssh`. The two threat scenarios of concern are any failures that could result in a loss of confidentiality or integrity.[1]

Investing millions of dollars in testing, SBU establishes a market price of $500,000 for a vulnerability in the default configuration of its operating system that could lead to a loss of confidentiality or integrity. To establish an upper bound on the security strength of its Linux competitors, it offers $100,000 for a vulnerability common to all Linux variants (RedHat, Debian, etc.) that could be exploited by an adversary to gain root access to the system. In `unix` variants, root access gives an adversary complete control of the operating system and the ability to violate the confidentiality and integrity of its data. A vulnerability is discovered, SBU purchases it, and then offers it to anyone willing to pay the $100,000 price tag.

As none of SBU's competitors are able or willing to shell out $100,000 to discover the vulnerability, the market price of a vulnerability in SBU's operating system has been demonstrated to be $400,000 higher than that of its competitors.

If your competitor buys the vulnerability report from you, then you will need to find another vulnerability in order to restore the upper bound on the security strength of your competitor's software. If your competitor continues to buy each vulnerability that you offer for sale then he may as well be offering the reward up front in order to establish a lower bound on the security strength of his system. This is equivalent to what your competitor would need to do to compete on security strength. If your product has fewer security flaws you will need to spend less money in order to support the same security strength reward, putting you at a competitive advantage.

[1]Because the system is designed to be used internally, availability is deemed not to be an issue.

> RedHat, Debian, and IBM form an alliance to purchase and fix this vulnerability, and any other vulnerabilities that SBU might obtain to prove that Linux is less secure. Little do they know, $100,000 was a very conservative lower bound for the cost to find a vulnerability in Linux. In fact, after paying $2,000,000 for the first twenty vulnerabilities, the rate at which SBU can buy up vulnerabilities at the $100,000 price point shows no sign of slowing. Having spent all of the funds from its charter, the alliance is forced to temporarily surrender the battle to keep their product as secure as SBU's and focuses on finding a more cost effective way to remove the vulnerabilities in Linux (perhaps using the approach in Chapter 6.)

Skeptics of MPV as a measure of security strength might ask what happens when a deep-pocketed firm tries to deflate the market price of a vulnerability in the competing firm's product by buying up vulnerabilities and then dumping them on the market far below cost. Imagine, for example, a market leading firm that buys up vulnerabilities in its competitor's products and then sells them for $100,000 less than they paid. Can such market manipulation sabotage MPV as a metric?

While the strategy above could certainly be exploited by a market leader, MPV remains a valid metric. A competitor who dumps vulnerabilities on the market at a price below its cost to obtain them is acting to subsidize the adversary. If an adversary receives a subsidy that helps him attack a specific software product, that product will be less costly to attack.

> Microsoft concludes that it is prohibitively expensive to use rewards to raise the market price of a vulnerability in Windows XQ, which has over one billion lines of code, beyond the current $25,000 level. Microsoft surmises that it will be more cost effective to buy vulnerabilities in SBU's product and sell them at the $25,000 price point. While it will end up costing Microsoft at least $475,000 for each vulnerability purchased (and then likely sold to SBU), Microsoft believes that with its deep pockets it can afford to buy more vulnerabilities at the market price than SBU can afford to buy for $25,000.

There are three good reasons why a firm would want to avoid dumping vulnerabilities in its competitor's product to raise its own comparative strength. Firstly, the stronger the product is the more expensive it is to dump vulnerabilities. If the market price rises with each vulnerability purchased, the cost differential gets progressively worse for the firm doing the dumping.

If the competing software developer that is targeted by this strategy can afford to buy the vulnerabilities and repair them, then the firm doing the dumping will end up subsidizing this competitor's testing. In the long run, this will only make the competitor's product more secure.

The third consequence is that if the firm can be shown to be actively subsidizing attackers (rather than just acting as a clearinghouse for trade in vulnerabilities) than it might be held liable for aiding and abetting criminal activity.

5.1 Chapter summary

In this chapter I showed how the economic approach to measuring security strength, detailed in the previous chapter, can be applied to the software industry. By providing techniques to bound the security strength of software products in terms of the market price of a vulnerability (MPV), I have supplied the tools necessary for software producers to prove the relative strength of their product in relation to that of their competitors.

If the market price of a vulnerability is to be the security strength metric used to differentiate products, it is essential that it be economical to measure. This issue is of particular importance as continued measurement requires that a reward (the bid price) be offered for a potentially unlimited number of vulnerabilities. MPV could be a prohibitively expensive metric if rewards are claimed more often than expected. Firms will also require a proven mechanism for improving the security strength of their software. In the upcoming chapter, I will address both of these problems by introducing a market approach to establishing lower bounds on a program's security strength during the software development and testing process.

Chapter 6

Developing strong software

Despite significant spending on security, software development firms continue to release vulnerability-ridden systems. While these firms have long rewarded teams that ship products on time, few offer their employees the incentives necessary to ensure that systems are designed, built, and tested to be secure. One barrier to creating such incentives has been the problem of measuring security. By the time vulnerabilities are discovered in a released product, the developer who wrote the code and the tester who verified it may no longer be working in the same product group, or even the same company.

Even when a firm makes a public commitment to focus on security, the lack of measurement and incentives makes it difficult to follow through. For example, after delaying product development to shore up security, Microsoft released Windows 2003 only to see that a steady flow of vulnerabilities continued to be reported at an alarming rate.

If a software development firm set out to release products with measured security strength, how could it reach its goals? If the firm plans to offer vulnerability rewards to establish MPV, can it do so without paying out more money than it can afford?

This chapter describes how a software development firm can reach its security goals in an economical manner.

I describe a strategy in which security strength testing is integrated into every stage of product design and development. At any time during product development, rewards will be offered for the report of vulnerabilities in the design or existing implementation of the system. From the beginning of testing until product release, the firm will bid for vulnerability reports. While these rewards may first be offered to in-house testers, at some point before the product is released to production these rewards will be offered to the public so that anyone may test the system and report vulnerabilities. The lower bound of the market price to find a vulnerability (MPV) will be established upon release of the software to production simply by continuing to offer rewards to anyone who reports a new vulnerability.

This process can also be used to test for software defects other than vulnerabilities. While I will use the more general words, flaw and defect, in this chapter, if you are focused exclusively on security you could just as easily substitute the word vulnerability. Rather than introduce new terminology for the market price to find a general software defect, we will continue to use the acronym MPV.

6.1 Desirable properties of markets for defects

Given the opportunity, what properties would a firm build into an ideal market for purchasing reports of defects, such as vulnerabilities, from testers?

VALUE	*The price paid for each defect found should be minimized.*

If the testing budget is fixed, value is obtained by maximizing the number of defects found. If the number of defects found is fixed, value is obtained by minimizing the amount of money spent in finding them.

Speed	*Defects should be found and reported as quickly as possible.*

The sooner a defect is reported, the sooner it can be fixed and the less likely it is to cause damage before it is repaired. Given trade-offs between value and speed, I will introduce solutions that place priority on value and then show opportunities for exchanging value for speed.

Order	*The easier a defect is to find (or the cheaper the defect report is to produce), the earlier it should be reported.*

The most obvious and most commonly occurring defects are both the most easy to find and the most likely to result in damage. Defects that are difficult to find are the least likely to occur when the system is operating and thus are the least likely to cause damage [17]. Product quality can be improved more quickly if the most common defects are discovered and fixed first. For any class of security vulnerabilities that cause the same amount of damage when exploited, locating and fixing the cheapest vulnerabilities to find will increase the MPV of the system faster than if vulnerabilities that are more difficult to find are given higher priority.

For example, a defect in a router that corrupts one of every thousand packets is of more concern than a defect that corrupts one packet per billion. A software company prevents more theft if adversaries must spend millions of dollars to break a cryptographic algorithm than if a vulnerability remains that can be found and exploited for a thousand dollars. That the most frequently occurring problems are often the easiest to detect is one of the few natural laws that work to the advantage of the software tester.

If we are focusing purely on testing products that have yet to be released, order will be the least important of these properties.

6.2 Market requirements

The market for defects will have one buyer, the firm charged with improving the quality of the system, and many sellers, who we will call testers as they are charged with finding and reporting defects. This one-buyer assumption may seem misguided, especially if the defects are security vulnerabilities that, if left unrepaired, would be of value to an adversary. In Section 6.5 we will see why an adversary is not likely to buy information about a vulnerability at any price that the firm is willing to pay, and thus need not be treated as a competing buyer.

With this in mind, the firm may set the rules of the market as follows:

ACCESS	*All testers have full and equal access to the system to be tested.*

The average price of a defect report cannot fall below the average cost of finding a defect, otherwise no rational tester would search for defects. To maximize value and speed it is in the firm's best interest to minimize the testers' costs by giving them full and complete access to the system.

PRICING	*A tester reporting a unique and verifiable defect at time t, complete with test case, will receive a reward $r(t)$. A tester reporting a non-unique defect (one that was reported earlier) receives nothing.*

The firm chooses any reward function $r(t)$ such that it increases continuously with time, starting with the lowest possible cost of finding and reporting a trivial defect and ending with the MPV reward that will be continue to be offered after the system is released.

This rule frees the firm from any need to understand the testers' costs. Rather, the firm will rely upon competition between the testers to minimize the total cost of the rewards it pays out.

While not essential to this analysis, I suggest supplementing the pricing rule to require testers to pay the transaction costs of processing defect reports (still receiving their reward if the report is valid). This discourages reports that are erroneous, badly detailed, or duplicates of published existing reports. Testers are expected to integrate these costs into the price they demand for each defect reported.[1]

SELLING	*A tester may search for defects at any time, and may report a defect immediately or at any time after finding it.*

While forcing immediate reporting would appear to maximize speed and improve order, the firm does not have enough control over the market to make this happen. More importantly, doing so would interfere with the testers' ability to wait for a higher reward and make a fair profit, and would thus reduce the incentive for testers to search for defects. Instead, we will again rely on competition to achieve our goals and ensure that defects are reported in a manner that is timely enough.

INFORMATION	*All testers receive the other testers' defect reports immediately after they are received by the firm.*

After we've ensured that testers receive the maximum information available about the system (via the Access rule), testers' efficiency can be further improved by ensuring that they are not wasting effort searching for known defects. The information rule is most sensible early in the testing process, when defects are common and it

[1]These transaction costs must then be subtracted from the market price to find and report a vulnerability (MPV) if this value is to measure a true lower bound on security strength.

is likely that a defect will be reported a second time before the firm can respond to the first report with a fix. After product release, a firm to may prefer to ensure that users have a chance to patch their systems before vulnerability reports are published. Such a firm would sacrifice value by delaying the release of security defects until after they are repaired. This will be discussed in more detail in Section 6.6.

6.3 Simplifying Assumptions

The market for defects has been constructed so that once the firm has declared its desired MPV it has no further strategic choices to make. The strategy lies completely in the hands of the testers.

In order to analyze how the testers will behave, we first simplify the nature and rules of the game until the analysis becomes trivial, then remove the simplifications in order to make the game better reflect reality.

The following story summarizes our simplified game.

Storyline

Frank's system has a defect that he is willing to pay $1,000 to locate. Frank invites Ann, Bob, and others to test the system over a thousand-hour period and to provide a test case that identifies the defect. The reward for reporting the defect in the first hour is one dollar. As each hour passes, the reward for reporting the defect is increased by one dollar. However, only the first player to describe the defect will receive the reward.

Ann can find the defect with $200 of effort. Bob can find the defect with $300 of effort. No other tester can find the defect for less than $300 of effort. All of these facts are known to all the testers.

The simplifications implied in the above example can be formalized in the following statements:

(1) *There is one and only one defect.*

(2) *The act of finding a defect is atomic and takes no time.*

(3) *Each tester knows his or her cost of finding the defect.*

(4) *Each tester knows the other testers' cost of finding the defect. Each tester knows that the other testers know each other's cost of finding a defect, and so on.*

If each player knows that a defect exists and knows everyone's cost of finding the defect (perfect knowledge as defined in simplifying assumptions 3 and 4), the market can be modelled as a cooperative game with multiple sellers (the testers) and one buyer (the firm). In the language of cooperative game theory, the *marginal contribution* of a participant is the amount of value they bring to the game, in terms of total profit for all players. For example, if Ann and Bob working together earn \$50 each in profit, but with Charles in the game Ann, Bob, and Charles all earn \$60 each, the total value earned with Charles in the game is \$180 and Charles' marginal contribution is \$80 (\$180-\$100). One finding of cooperative game theory is that a player cannot expect to earn more than their marginal contribution, as if they do the other players will be better off to play the game without them.

For the rules of our simplified market, all testers are offering an information good that is a perfect substitute for the others' good as all are describing the same defect. The firm is only willing to pay for this information from one of the testers. If there are two testers with the same cost of finding the defect, they both have a marginal contribution of \$0 as either can be replaced by the other. If one tester can find the defect at a lower cost than all other testers, her marginal contribution will be positive, and will be the difference between her cost of finding the defect and the next best tester's cost of finding the defect. This is because every dollar saved in the cost of finding a defect is either a dollar of profit for the tester or a dollar saved in price of the defect report paid for by the buyer.

If, in our story above, Ann sells a defect to Frank at price p, her profit is $p - \$200$ and Frank's profit is $\$1000 - p$. These sum to an aggregate profit (the result of a characteristic function in cooperative game theory) of \$800 total between Ann and Frank. Without Frank in the story to buy the defect, there would be no transaction and no profits. Therefore Frank's marginal contribution is \$800. The aggregate profit if Frank transacts with Bob is \$700, as it costs Bob \$100 more to find the defect. Because Frank is better off buying from Ann, Bob's marginal contribution is 0. Since the aggregate profit would decrease by \$100 without Ann, her marginal contribution is \$100. Ann can expect to sell the bug report at a price between her cost, \$200, and her cost plus her marginal contribution, or \$300.

Since the firm does not have the ability to negotiate in this market (the testers report defects at the time of their choosing) and all testers are assumed to be rational, the tester with the lowest cost of finding a defect can demand a price just below the lowest cost at which any other tester could report the defect. That is, she can demand to earn a profit of any amount up to (but not necessarily including) her marginal contribution. In our story, the smallest discrete unit of currency is a dollar, and so the tester earns one dollar less than her marginal contribution.

> Ann believes that Bob and the other testers are rational and that they will not sell the defect at a loss. Since no other tester can find the defect for less than \$300, Ann can sell the defect at any price below \$300. To maximize her profit, Ann finds the defect and reports it when the reward reaches \$299. The price and timing of the sale are shown in Figure 6.1.

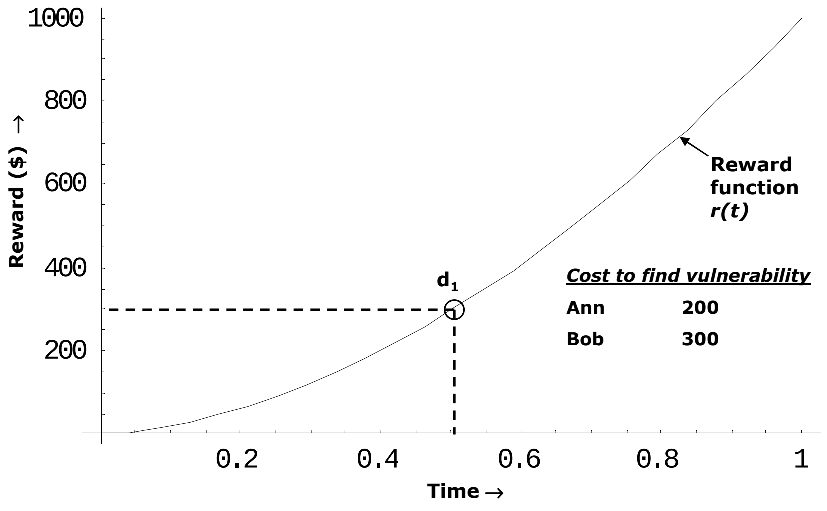

Figure 6.1: An ascending price bidding strategy is used to purchase a single defect.

6.4 Approaching Reality

In the real world, products have multiple defects, finding a defect is a time consuming operation requiring a large number of subtasks, and each tester's knowledge is far from perfect. We'll now try to replace the simplifying assumptions above with others that more closely match the realities of testing.

6.4.1 The presence of multiple defects

This bidding strategy would be of little value if it could only be used to find a single flaw in the system. This assumption must be removed, which we will represent by writing it again and crossing it out.

(1) ~~*There is one and only one defect.*~~

In order for the our bidding strategy to accommodate reports of multiple defects we must substitute a new assumption for assumption 1.

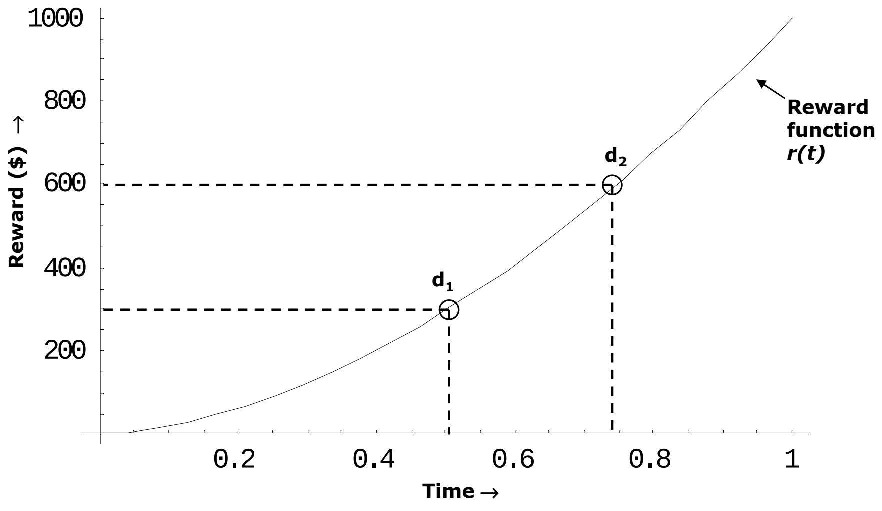

Figure 6.2: An ascending price bidding strategy is used to purchase two independent defects.

(1a) *All defects are independent. Finding one defect d in the set of all defects D neither aids nor hinders the search for another defect $d' \in D$, nor does it indicate that a tester is more or less likely to find another defect.*

With this new assumption in place, we can simultaneously bid on any number of defects in parallel, even if we don't know how many defects there are.

> There are now two defects d_1 and d_2, each of which Frank is willing to pay \$1,000 to locate. Ann's costs of finding defects d_1 and d_2 are \$150 and \$600 respectively. Bob's costs of finding defects d_1 and d_2 are \$300 and \$200 respectively. Ann reports d_1 for a reward of \$299 and Bob reports d_2 for \$599. The prices and timings of the sales are shown in Figure 6.2.

6.4.2 Knowledge about others' costs (part one)

(4) ~~*Each tester knows the other testers' cost of finding the defect.*~~

It's quite unlikely that a tester can know how hard it is for every other tester to find a defect, especially when she has yet to do the work to find the defect herself. We will replace this assumption with one that is somewhat less far fetched. This will enable us to make progress in the analysis before relaxing this assumption further in Section 6.4.4.

(4a) *Once given knowledge of a defect d, a tester with one of the two lowest costs of finding d will know the other lowest cost tester's cost to find d.*

There are now two sets of moves for each tester. First, a tester must decide if and at what price she will search for a defect. Then, if a tester has found a defect, she must decide at what price she will sell it. She can calculate the appropriate times for these actions from her corresponding choice of price by using the inverse function of $r(t)$.

If Ann knows her cost of finding a defect (Assumption 3), and we define C_a to be this cost, we say that Ann will find a defect at a time t for which the reward $r(t)$ covers her cost. That is, at some point at which $C_a \leq r(t)$. At this time t she can immediately sell the defect without any loss[2], or wait until just before the next lowest cost tester will find the defect. She can do this because once she has found the defect she will know the next lowest cost tester's cost of finding the defect. Though the story changes slightly under these new assumptions, the ending remains the same.

> Ann and Bob's costs haven't changed, but neither player starts out with any knowledge about the other's costs. When the reward price reaches \$150, Ann spends \$150 to find defect d_1 knowing that she can sell it immediately at no loss regardless of what she discovers Bob's cost of finding d_1 to be. After spending her \$150, Ann learns everything she needs to know about d_1, including that it would cost Bob \$300 to find d_1. Ann now knows, just as she did in the previous example, that she can wait to report the defect until the reward is \$299. Using the same logic, Bob will find defect d_2 once the reward reaches \$200 and will once again report it when the reward is \$599. The prices and timings of the sales remain exactly as before, and are still accurately represented by Figure 6.2.

[2]We ignore the unlikely event that Ann's cost is exactly the same as Bob's, and that they both attempt to report the defect at the same unit of time, as this becomes increasingly unlikely as our measurement of time becomes increasingly less discrete.

6.4.3 The time and cost of searching

(2) ~~*The act of finding a defect is atomic and takes no time.*~~

(3) ~~*Each tester knows her cost of finding the defect.*~~

Finding a defect can require both time and expense, and it is hard to determine the cost of finding something until you know what it is you are looking for. The following assumptions are much more realistic.

(2a) *A unit of work spent searching for a defect is an atomic task, has a fixed cost c_w, and takes no time.*

(3a) *Each tester can estimate the probability p_w that a task will reveal a previously unreported defect.*

> Ann has a test that she can design and run for \$5. The test has a 1% chance of revealing an unreported defect. Once the reward for reporting a defect has reached \$500, running the test will have an expected reward no less than $0.01 \cdot \$500 = \$5.00$. Ann, who is risk-neutral, decides to perform the test at that time.

More formally, we say that a tester will search for a defect when her expected minimum reward justifies the cost of searching. The equation below describes the requirement that cost of searching cannot exceed the expected reward, which is the probability that searching yields a reward times the amount of the reward. This is essentially the same equation that was introduced in Section 4.1.

$$c_w \leq p_w \cdot r(t) \tag{6.1}$$

Simplification 2a will not be further refined, and so our analysis will continue to be predicated on the assumption that a unit of work takes no time. Since even small units of work take time, a trade-off exists between the speed and value in testing.

The choice of testing period is one which needs to be made based on the time and budgetary constraints of firms with knowledge of the state of the art in testing tools and technology. This is thus outside the scope of this dissertation.

6.4.4 Knowledge about others' costs (part two)

(4a) ~~*Once given knowledge of a defect d, a tester with one of the two lowest costs of finding d will know the other lowest cost tester's cost to find d.*~~

Removing this assumption about knowledge makes the players' second move strategy less clear. All we can do is analyze the strategies of each player based on his or her beliefs about the costs of the other players.

Under our revised assumptions, when Ann finds a defect she does not know the likelihood that another tester will report that defect at time t for reward $r(t)$.

We will represent Ann's estimate of the probability that no other player will have reported defect d at time t as $p_a(T\backslash a, D_r, d, t)$, where $T\backslash a$ is the set of all testers except Ann, and D_r is the set of defects that have been reported.

Ann will report the defect at a time that maximizes her expected payoff function, which is once again the product of the reward times the probability of receiving the reward:

$$p_a(T\backslash a, D_r, d, t) \cdot r(t) \tag{6.2}$$

Figure 6.3 shows the decision Ann faces when she discovers a defect at the time when $r_n(t) = 0.3$, where $r_n(t)$ is the reward function $r(t)$ normalized by dividing its result by the maximum value of its range. This normalization simplifies the reward function so that the maximum reward is 1. In this particular example, in which

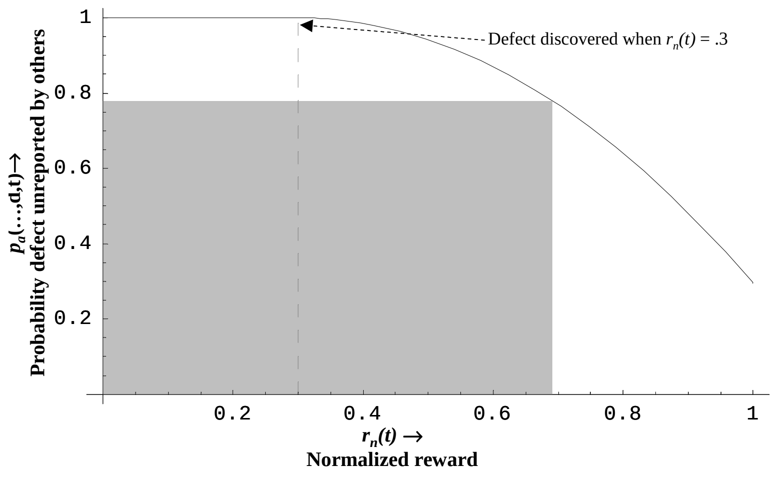

Figure 6.3: A plot of the cumulative probability distribution, $p_a(T\backslash a, D_r, d, t)$, that someone other than Ann will have reported defect d before time t to collect a reward less than $r_n(t)$. The shaded region represents $r_n(t) \cdot p_a(T\backslash A, D_r, d, t)$, or Ann's expected reward should she risk waiting until $r_n(t) \approx 0.68$ to report the defect.

arbitrary values were used for sake of illustration, the strategy that optimizes Ann's expected reward, represented by the shaded region, is to report the defect when $r_n(t) \approx 0.68$.

The better Ann's expected payoff function approximates the knowledge of assumption 4a and the behavior of the other testers who might find the same defect, the more money Ann will be able to make. The uncertainty, however, should encourage Ann to report earlier than she might if given the perfect knowledge she had with Assumption 4a.

6.4.5 Defect dependencies and learning about the skills of others

There is a strong case for removing the assumption (1a) that all defects are independent and that no useful information about the rate at which other defects will

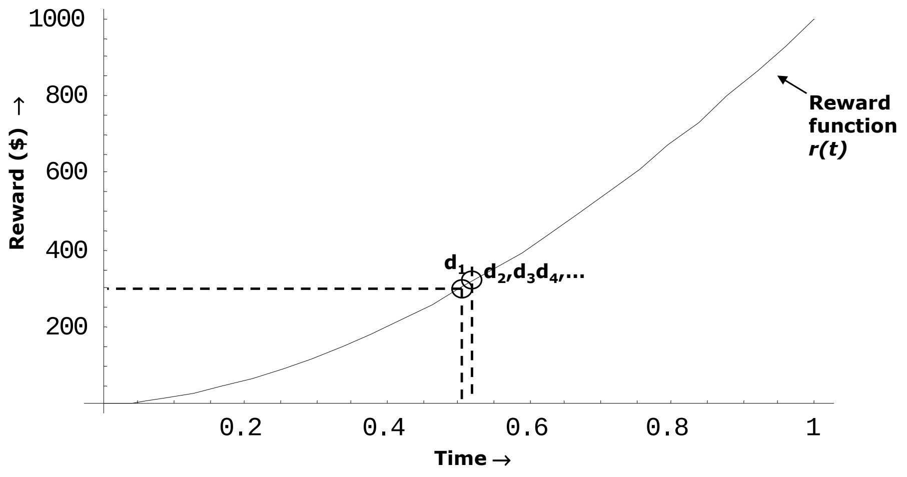

Figure 6.4: Defects are not independent as the discovery of one defect may lead to the discovery of many related defects.

be found by watching the reporting of existing defects. For starters, we would expect that if Ann discovered a number of defects only to watch Bob report each defect at a price below her estimates of his costs, Ann would revise her estimates of Bob's cost to find each additional defect.

We also can't ignore the possibility that the discovery of one defect will simplify the discovery of similar defects, as shown in Figure 6.4. Many defects are often related, so much so that it can be difficult to determine where one defect ends and another begins.

(1a) ~~*All defects are independent. Finding one defect d in the set of all defects D neither aids nor hinders the search for another defect $d' \in D$, nor does it indicate that a tester is more or less likely to find another defect.*~~

Once we acknowledge that the discovery of a defect changes every player's cost of finding another defect, adding a new assumption won't fix the rules of the game. Rather, it is our bidding strategy and not the simplifying assumptions that needs repair. The following example shows how the firm may fail to obtain value when using a monotonically increasing bidding function.

> Frank's system has three defects that are all closely related. Ann can find the first of three defects for $500, and can find each of the two remaining defects for $10 each given knowledge of the first defect. Bob can find a defect for $2000 but given knowledge of any of the defects can find the other two for only $5. Charles can find any defect for $600, but isn't aided by information about defects that have already been discovered.
>
> Ann finds the first defect when the reward has reached $500 and then instantly finds the two other defects. She reports all three defects at once when the price reaches $599, collecting $1,797.

The above example demonstrates two problems with the rules as they stand. Since Bob has a lower cost of finding the incremental defects and Ann is the one reporting them, the most efficient tester is not the one testing for the incremental defects. Since the reward function increases monotonically, the firm must grossly overpay for the incremental defects rather than paying no more than the cost of the second lowest cost tester. By the second lowest cost rule, Ann should collect $599 for reporting the first defect, and Bob should collect $9 for each of the additional defects, allowing the firm to pay $617 for the three defects instead of $1,797.

The simplest way to restore value is to reset the reward (bid) price each time a single defect is reported, lowering it down to the transaction cost of reporting a defect ($0 in our story) as shown in Figure 6.5. If the reward is reset to $0 when Ann reports the first defect, and the report is made available to all other testers, Bob will learn about the defect, find the next defect, and report it when the reward price reaches $9. The reward will once again be set to $0 and after it climbs to $9 again, Bob will report the third defect. While this approach sacrifices speed, without it value is all but nonexistent.

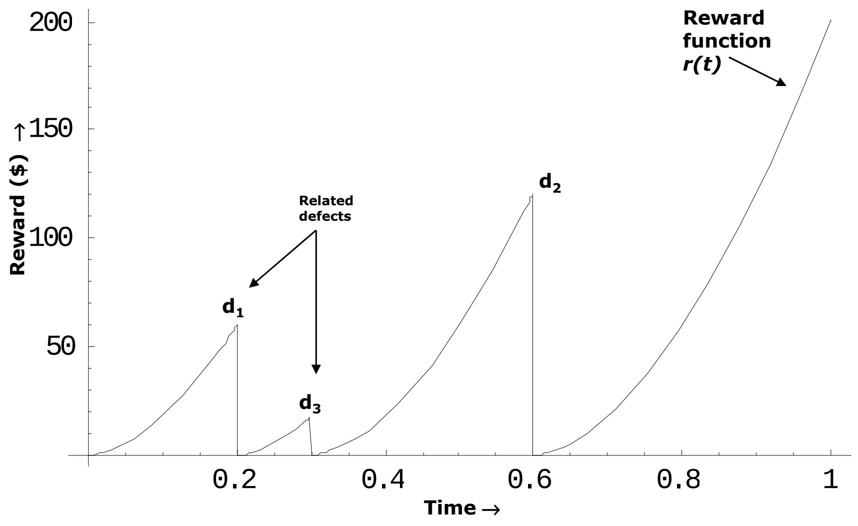

Figure 6.5: Resetting the reward to avoid overpaying for defects that are easily found after the discovery of a related defect.

The firm may want to adjust the reward function so that the bid price grows very slowly (or remains at \$0) while an outstanding defect is being fixed. This prevents a slew of related defects and test cases, which are trivial modifications of the first reported defect, from being rapidly submitted and overflowing the reporting system. Alternatively, allowing the reward to increase and a flood of related test cases to be reported may be just what is needed to build a good regression testing suite.

There is also a rather practical benefit to setting the reward back to \$0 each time a defect is reported in that it allows for more predictable budgeting. If the reward function $r(t)$ increases linearly, for example if a dollar is added to a reward 'pot' every hour and reporting a defect allows the tester to take the money in the pot, then the rate at which defects are reported does not affect the rate at which the testing budget is spent. The disadvantage is that fixing the budget removes any guarantees about the cost to find and report a defect (MPV) after any given testing period. The firm may be better off increasing rewards at faster rates when it is falling behind in meeting its security and reliability goals.

Depending on how $r(t)$ is chosen, Ann may now wish to recalculate $p_a(T\backslash a, D_r, d, t)$ for her unreported defects each time a defect is reported and the reward is reset. In doing so, she may actually determine that she may want to take a loss if she has underestimated the other testers in her past calculations. However, the same equations still guide Ann's strategy in selling the defects that she has found so long as she still seeks to maximize her expected reward.

6.4.6 Some defects are more important than others

Defects that may result in different failures, or may be exploited to carry out different threat scenarios, have different consequences. Rather than treat all defects equally, the firm will likely want to separate them into classes based on their severity. Larger rewards can then be offered for defects in classes that yield the greater threats. While the firm could create a separate bidding strategy and corresponding reward function for each class of defect, it is easier to pick one class of defect as the baseline class and measure the other classes relative to that baseline class. If the firm is willing to pay only half as much for a defect of a new class as it is for the baseline class, the reward for finding a defect from that class at time t can be set to $\frac{1}{2}r(t)$. Generally, if a defect is from a class that the firm is willing to pay x times as much as one from the baseline class, the reward for reporting such a defect should be set to $x \cdot r(t)$.

This approach is equivalent to offering parallel bids for each class of defect using separate reward functions, except that all the rewards for all defect classes are reset each time a defect of any class is reported. Since finding a defect in one class may make it easier to find one in another class, resetting all rewards is indeed necessary.

6.5 Adversaries and the one-buyer assumption

We began our analysis by assuming that the firm is the only buyer of defects during the testing process. One might be concerned that, during the testing process, adversaries may purchase reports of defects that are categorized as security vulnerabilities. If the purchased defect remained unreported after testing concluded and the software was put into use, the adversary could then exploit the defect to target vulnerable systems.

If R is the amount of the reward offered by the firm at the end of the testing process, we already assume that it may be possible for the adversary to purchase vulnerability reports at a price greater than R. Establishing a market price for finding a vulnerability in the system does not prevent an adversary from paying above this price to find a vulnerability for his own purposes.

A vulnerability is of no value to an adversary if the flaw is discovered and fixed by the firm before the product is released. An adversary who buys a vulnerability during the testing period is thus wagering that the vulnerability will remain unreported until after the product is in use. He may buy a vulnerability at a price v ($v < R$) during the testing period. A risk-averse tester might sell a vulnerability at price v rather than either accepting a smaller reward $r(t)$ offered by the firm at the same time or waiting to sell the vulnerability to either the firm (at price $r(t + \epsilon)$) or the firm's adversary. By doing so, the tester transfers to the adversary the risk that the vulnerability will be discovered by another tester.

However, there are many reasons why the adversary should be wary of any such deal. It pays for the tester to sell the adversary the defect that is most likely to be detected by another tester. It is now the adversary who would face a market for lemons [1] when buying vulnerabilities.

The difficulty faced by the adversary is more pronounced if the tester is risk-neutral, as such a tester will only sell those vulnerabilities that he does not expect to be able to sell to adversaries at a price greater than or equal to R after the testing period is over.

One might suppose that a tester could sell at a price below R and make up for the difference in volume by selling to multiple buyers. In such a case, a buyer is able to engage in arbitrage by purchasing the defect and selling it to the firm. This will result in the defect being fixed, and any exploits of the defect becoming worthless. What's more, once the first buyer has purchased the exploit he would also be able to sell the exploit to other buyers. As there would now be multiple sellers of the same information good, the price of the good would drop to zero. For both these reasons, a tester should not expect to be able to sell a security defect more than once.

In fact, the tester's ability to resell security defect information adds an additional barrier to the sale of security defects to adversaries at *any* price. Any adversary must worry that after he has paid to learn the details of the vulnerability, the seller will resell that information anonymously to the software development firm (or other buyers). Anonymity enables the tester to claim that another tester must have discovered and reported the defect. Since the only way to ensure the testers won't resell is to take them out of the game, testers who do not want to put their safety at risk will be wary of selling security defect reports to anyone who intends to use them to attack systems.

6.6 Delayed publication of reported defects

It may not always be advantageous to release defect reports immediately to all testers. This is especially true when the test is open to the public, the product is in use by customers, and the defect is a security vulnerability.

If defects are not released immediately, the firm should continue to pay only the first reporter the full reward or it should split the reward among each tester. This will cause testers to demand more for each defect, as they must bear the risk of not receiving the full reward for their efforts in searching for defects. Testers will also be wary that the firm might be tempted to create a phantom tester which it could claim reported defects before the true reporter. If the testers do not trust the firm, the firm may need to employ a trusted third party to manage the entire defect reporting and reward process. If only the first reporter receives the reward, the trusted third party may only be needed to sign and time stamp a message authentication code of each defect report submitted, thereby providing proof of who found the defect first.

The firm should never give a full reward $r(t)$ to every tester who reports a defect before the public is made aware of the defect's existence. This may seem like a good way to put speed before value and encourage as much testing as possible, but it will more likely cause the testers to collude, sharing defects to bleed cash from the firm. This is particularly serious if adversaries pose as testers and learn about vulnerabilities by taking part in such collusion.

6.7 Applying strength metrics throughout product development

Beyond using the MPV bid price settled on at the termination of the testing process to differentiate completed products, security strength and reliability metrics can also be used throughout the development process. When systems are assembled from component subsystems, financial resources can be adjusted so that the weaker components receive more testing attention. If a failure in any single component may lead to a security breach (a weakest link situation), then the goal should be to shore up the security of the weakest component.[3]

For example, consider a highly secure web server running on an insecure operating system. System builders, faced with the choice of how much to invest in the security of a component, often err by paying too much for one component and too little for another. The firm should invest in increasing the security of the component with the lowest MPV, the operating system, and not the security of the web server, which has a higher MPV.

Market measures of reliability and security strength can also be used when compensating all types of test engineers. Public testing does not eliminate the need for professional internal testers, as they will be able to perform white box testing to find defects at lower cost using information not yet available to the public. Internal testers are also best used early on as the defect reports they produce are of higher quality and

[3]Hal Varian [114] has provides a game theoretic analysis of the strategies that result when different firms control different components of a system in both a weakest link and strongest link scenario.

are thus cheaper for the firm to verify and process. Such costs may be a significant component of the transaction when defects are inexpensive to find.

Testing often suffers because the job is considered unrewarding. Test engineers are often paid less than developers and their positions are less esteemed, since building systems is more highly regarded than finding problems with them. All too often, good testers are promoted out of testing. This would be less likely to happen if testers' compensation more directly matched the contribution they make to the product. Markets for defects create a meritocracy in which the best testers are rewarded in a way that makes their jobs more desirable while ineffective testers are encouraged to find more suitable positions. One might consider whether great testers lost to development teams will start testing systems again, in their free time, to pick up extra money.[4] Some might even learn that their true calling is to return to testing.

Testers are not the only members of the team whose incentives can be better aligned through competitive testing. Software developers are often rewarded more for the speed of their work than the quality. This is in part due to the difficulty the industry has faced in measuring software quality. The market price to find and report a defect when the engineer releases his work for testing provides the measure of security strength and reliability that can be integrated into the engineer's incentives.

If development is to be out-sourced, MPV may be used to ensure the contractor has fulfilled its obligation to provide a system of a reasonable reliability and security. The lack of good metrics for software quality and security strength has caused innumerable disputes over whether or not a contractor has fulfilled its obligations. Placing a

[4]Developers should not be allowed to collect rewards for testing their own systems as doing so compromises their incentive to create as few defects as possible.

lower bound on the market price of finding a defect in the completed system may be just the metric needed to allow parties to reach agreement on quality before work begins. A wise client will pay more to ensure that the testing budget comes out of the contractor's pocket, thus ensuring that the contractor will have the maximum incentive to provide initial quality. The client should also insist that the testing itself be open to third parties that cannot be influenced by the contractor.

Perhaps most importantly, the market metrics can be used as a measure for determining when systems are ready to ship. They are especially useful when other metrics, such as Mean Time Between Failures (MTBF), are hard to measure or not applicable. Most of these other metrics fail to account for extreme conditions, such as excess load or attack by an adversary with extensive knowledge of the system, that would not otherwise occur in the laboratory and aren't detectable in typical real world use.

6.8 Chapter summary

I have described a market-based approach to software testing designed to simultaneously accomplish two different but important goals: ensuring that a product is released with a measured level of security strength and ensuring that the incentives of software testers and software developer are aligned. More generally, this approach can be used to develop software that is reliable as well as secure.

Chapter 7

Modelling security risk

A statistical model describes the relationship by which one metric, or dependent variable, depends on the values of one or more other measures, or independent variables. The more accurate the model, the better it can estimate the dependent variable as a function of the independent variables.

Outside of computer security, researchers have had some limited success in creating regression models of security risk. For example, models have been created to estimate the likelihood that a home will be burglarized, given measures that indicate the presence of factors that predispose a home to burglary or the presence or absence of safeguards. These models have been successful because the independent variables required to construct them are straightforward to measure or already available.

If a statistical model is to accurately forecast future risk, the relationships between the dependent and independent variables must remain stationary over time. Previous models have been assumed to be stationary because the threat scenarios they address, such as burglary, evolve very slowly. The challenge of modelling security risks in software systems is that these systems, and the attacks on them, are in constant flux due to the rapid appearance of new vulnerabilities and safeguards. To overcome this challenge we will isolate properties of these systems that remain stable over time and that can be correlated with the security risk metric that serves as our dependent variable, such as the annual expected losses resulting from attacks.

In this chapter I will provide a brief introduction to regression models and examine how they might be used to model the likelihood of successful attacks that result in security breaches. I will describe how others have applied models to estimate the security risk to homes from burglary. I will then address why translating this approach to computer security has proved challenging, and what can be done to overcome these challenges over the next several years.

7.1 An introduction to regression models

Regression models are tools used to estimate a measure, or *dependent variable*, as a function of a set of other measures, or *independent variables*.[1] For example, given a dependent variable Y that appears to change linearly with the value of independent variables X_1 and X_2, we might estimate Y with a value $\hat{Y}$, that is defined via a function f.

$$\hat{Y} = f(X_1, X_2) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

For a given data point i, the difference between the estimate $\hat{Y}_i$ and the measured Y_i is the error term u_i.

[1]For a more detailed introduction to regression analysis, I recommend Stock and Watson's textbook on Econometrics [108].

$$Y_i = \hat{Y}_i + u_i = f(X_{1,i}, X_{2,i}) + u_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + u_i$$

A regression is the process by which values are assigned to the β constants so that $\hat{Y}$ will best estimate Y. The β values for linear models are regressed to minimize the sum of the squares of the error terms, $\sum_{\forall i} u_i^2$, whereas the maximum likelihood estimator is used for more complicated models. Further error estimates are used to measure how well the model fits the data. One may also calculate the likelihood that a relationship between the dependent variable and one or more of the independent variables is statistically significant. To model more complex relationships, regressions can be performed on functions with non-linear terms, and even with terms in which variables interact (e.g. $\hat{Y} = \phi\left(\beta X_1 X_2\right)$).

Once the regression has selected the optimal β values, we can hold those values constant and use $f(\vec{X})$ as a functional model for estimating the value of Y.

7.2 Modelling security risk

All of the security risk metrics described in this thesis are functions of the rate (or frequency) at which security losses are expected to occur. Security risk regression models must therefore forecast the rate at which each type of attack succeeds.

Successful models can be created when the circumstances that could lead to a security breach are well understood, when the rate at which breaches occur is a function of independent variables (regressors) that are directly or indirectly measurable, and when the dependence of this breach rate on these variables remains stationary over time. If important indicators of security risk aren't measurable or simply aren't measured, the model will suffer from omitted variable bias. Even if a model can accurately estimate past security risks using historical data, the model will be of little value if the relationships on which the model relies no longer hold. This is of particular concern in security as the dominant strategy of our adversaries is often to behave counter to our models and defy our predictions.

The frequency with which a system will be successfully attacked, and security breached, is likely to depend on at least four sets of factors each of which may be represented by one or more independent variables.

First, there is the question of how many individuals are in a position to attack the system. The *number of potential adversaries* is likely to be positively correlated with the frequency of security breaches and the resulting security risk. If the class of breach being studied is one caused by an insider attack, the set of potential adversaries is likely to be smaller than for an outside attack.

Secondly, there is the question of how valuable the attack appears to be to the system's potential adversaries. The greater the *incentive to attack*, the more likely it is that a potential adversary will choose to stage an attack. Thus, the incentive to attack is also positively correlated with frequency of attacks, rate of successful attacks, and the resulting security risk. The incentive to attack must be understood in the context of the other options available to the adversary for accomplishing their personal goals and the relative attractiveness of these options.

The presence of factors that provide disincentives to attack are expected to be negatively correlated with the frequency of attacks and thus also negatively correlated with security risk. One class of disincentives is *attack risk*, or the risk of undesirable consequences to the adversary as a result of attacking the system. These consequences may include disclosure of the adversary's attack tools or techniques, reputation damage, capture and incarceration, or even physical harm.

Finally, the collective cost of equipment, effort, and other resources required to stage a successful attack also acts as a disincentive and is negatively correlated with attack frequency and security risk. The stronger the system, the greater the expected cost to successfully attack it. Thus, *security strength* is negatively correlated with attack frequency and security risk.

7.3 The scenario of home burglary

Because use of regression analysis does not date back to the castles of the middle ages, a more contemporary example is in order to replace our stand-by from Chapter 2. A maxim of English common-law is that a man's home is his castle, and so it should not be surprising that the problem of securing the modern home from the common adversarial threat of burglary has been extensively studied. Statistical data has been collected and applied to regression analyses to estimate both the risk of burglary for a given home and the effectiveness of safeguards to reduce that risk [20, 56, 98].

An example set of independent variables measured to determine their effect on burglary rates is shown in Figure 7.1. The independent variables include factors from all four of our above categories. Proximity to highway exits increases the size of the adversary population. The value of the home increases the perceived incentive to attack. Alarms increase the risk to the attacker and dead bolts increase the effort required to enter the home (security strength). Independent variables that are posed

- Is a dead bolt on the door?
- Is an alarm installed?
- Is a car always in the driveway?
- Is a light on timer or motion sensor?
- Is a radio or television on timer?
- Are mail/papers left uncollected?
- Is home on corner of block?
- Is home $\frac{1}{4}$ mile from highway exit?
- Adjacent to woods or playground?
- Value of home
- Number of children living in house
- Number of years residing in home

Figure 7.1: Independent variables from the regression analysis of the likelihood of household burglary in Hakim *et al.* [56]. Variables described by questions are boolean variables that take a value of either zero or one. Those not in the form of questions are scalar variables.

in the form of boolean (true or false) questions take the binary value of either 0, for false, or 1, for true.

The study of Shachmurove *et al.* [98], performed on data collected from the town of Greenwich, Connecticut, resulted in a model that could be used by homeowners to make better security decisions. They showed that the most effective deterrents were those that either increased a burglar's risk of being detected or that merely led the burglar to believe he was more likely to be detected. While alarms were the most effective, lights left on timers and cars in driveways were also likely to cause burglars to choose a different target. More surprising was that security measures designed to increase the difficulty of entry, such as dead bolts, did not significantly reduce the likelihood that a home would be burgled.

While surprising, the regression study is consistent with findings of Wright and Decker [119], who surveyed 108 burglars in the St. Louis area. They also reported that alarms were a strong deterrent, with 56 of 86 burglars reporting that they would never target a house known to have an alarm [119, page 125]. They report anecdotal evidence that burglars are hard to deter once they've committed to approaching a house. When confronted with dead bolts, burglars will either break them or enter the house through a window instead. Windows with a single pane can be broken without making much noise and thus without greatly increasing the risk of the burglary. In the words of one of the burglars interviewed, "as long as houses are made of wood and glass, I can get 'em" [119, page 98].

If home security practices have benefited from regression models, why hasn't the computer industry applied similar security risk models to evaluate the value of its security mechanisms? One reason is that computer systems are far more complex and heterogenous than homes. Homes are built of "wood and glass" and serve a common function. Software may be written using different algorithms coded in different languages. Because different software packages perform dramatically different functions for their users, their architectures are varied, as are the threats to which they are vulnerable. Perhaps most importantly, many computer crimes can be committed from a safe distance, making risk far less of a deterrent for these computer criminals than it is for burglars.

7.4 Regression models in computer security

7.4.1 Prior work

While we have yet to see software security regression analysis studies to estimate security risks, as we have seen for home burglary, a limited number of studies have used statistical analysis and regression models of security data for other goals.

One of the earliest applications of statistical methods to security risk is a 1990 study by Detmar Straub [110], who showed that organizational commitment to invest in computer security has a statistically significant effect. Because Straub's intent is to show that certain actions have effect in reducing risk, rather than to measure the level of security risk, he does not use a full regression model. As the study was performed well before most corporations were connected to the Internet, the results are primarily of interest when studying scenarios involving local attack. Another key limitation of this study is the poor resolution of the independent variables. For example, one independent variable represents the number of security programs installed on a system, without regard for what these programs do or how up to date they are.

Straub's ability to show statistically significant effects using such imprecise independent variables would imply that regression models using more precise measures could have promise. Indeed, regression models have been used to measure the rate of security incidents that exploit a vulnerability, as a function of the time since the vulnerability was discovered [19]. Regression analysis has also been used to show that the overall frequency of security incidents is increasing [61]. Given the potential of regression models for measuring security risks, it is not surprising that such studies have been proposed to study computer fraud [70] and insider attacks [97]. However, the resulting studies have not been forthcoming.

While not directly estimating security risk, Moitra and Konda [71, 72] used regression models to generate the constants for a stochastic model of yet another metric, system survivability. The problem with this study was not the regression model, but the stochastic model. The stochastic model assumed attacks arrived at random, and so the utility of this work is limited by the underlying assumption that attackers behave in a random, and not strategic, manner.

7.4.2 A problem of data

Why haven't regression studies been used to measure the security risk of software systems and the effects of the safeguards designed to protect them? Successful studies not only require a sizeable amount of data, but it must be the right data. This data may not always be readily available or easy to measure.

When protecting a system, one will want to invest in safeguards in a manner that reflects the presence, and severity, of different threats for that system. For example, a payroll database faces very different threat scenarios than a web server. The payroll data is likely to be well isolated from outside attack, but may be a target of insiders who might seek to change the size of a pay check or learn their colleagues salary. Because the web server must respond to outside requests, it will not be as well isolated from outside attack. Remote adversaries may target the web server with the goal of using the machine as a stepping stone from which to access the rest of the network. Because different systems face different threats, predictive models of security risk should be created for each threat scenario.

To build a regression model for a threat scenario, such as was done for home burglary, one must design and distribute a questionnaire, collect the results, and analyze them. This becomes a daunting task when one tries to address the full range of threat scenarios faced by computer systems. One reason is that, for each scenario, the survey must measure independent variables that will remain predictive over time. Many questions, such as the number of attacks witnessed and dollar value of yearly losses, will need to be repeated for each scenario. If these questions are not repeated for each scenario, and responders are instructed to aggregate data for all scenarios into a single response, then predictive models at the scenario level will be less reliable or impossible to build.

One reason given for the lack of quantitative security studies is the lack of data. One of the most well known data sets comes from a joint survey [29] run by the Computer Security Institute (CSI) and the Federal Bureau of Investigation (FBI). The survey questions ask about the presence of ten types of safeguards, and breaks down losses into twelve categories, based on scenarios. Survey questions also inquire as to predisposition to attack, such as whether the organization has a web site and whether that site is used for electronic commerce. Unfortunately, the CSI has yet to use the data collected for a regression study, and confidentiality concerns have prevented the data from being released for outside study [82]. What's more, the CSI/FBI study asks its questions at the organizational level. Because organizations employ a variety of different software packages on myriad machines, it is difficult to infer the effects of safeguards at a system level from organizational data.

Because the CSI/FBI study is not intended to help firms choose between different products, the survey cannot be used to determine which software products are more prone to attack or which safeguard is the best of its class. The survey reports the number of insider attacks and the losses that resulted, but there is no data that would enable us to determine which human resources practices, such as employee screening and monitoring, could reduce the likelihood of these incidents. Nor is data collected on other aspects of security strategies, such as budget or person-hours dedicated to monitoring networks.

Even if surveys asked the right questions, critical indicators of security are not being measured by the organizations that respond to these surveys. While one can determine the presence of active safeguards, these safeguards are not the only factor that deter adversaries. Other security investments, such as employing stronger software, can also act to deter or thwart attacks, but the presence and effects of these investments are not measured. These effects must be measured if defensive resources are to be allocated effectively.

To explain why some scenarios can be modelled using existing measures of security while others require new measures, I introduce two distinct threat scenarios. Using existing measures we can model rate of security breaches that result from insider abuse or negligence. New measures are required if we are to predict the reduction in security risk resulting from strategies to thwart remote network attacks that target software vulnerabilities.

7.5 Insider threats vs. network attacks

The breaches resulting from an insider threat occur outside the boundary of the software system. Software systems cannot eliminate the possibility of these insider threats because these systems rely on instructions and data provided that come from outside their boundaries. Breaches at the organizational level can occur if system administrators abuse their powers, corrupted data is provided as input, or rules for confidentiality handling system outputs are disregarded.

Like burglars, insiders face significant risk when they attack a system. Their role within the organization alone makes insiders suspect. Many insider attacks must also be executed from within close proximity to the target system. While an insider may already have access to the system he is attacking, gaining access often requires identifying himself and consenting to the monitoring of his actions.

Fortunately, the number of potential inside attackers may be significantly smaller than the number of potential burglars that might attack a home. The smaller the number of individuals granted access that could be used to carry out an attack, the lower the risk that an individual who would abuse this access will be among those selected. All else being equal, the number of potential adversaries will be negatively correlated with the likelihood of a breach.[2]

If insiders already have the access required to attack a system, then risk, and not the cost of effort or resources, will deter the insider from staging attacks or acting negligently. Because perceived risk is strongly correlated with the presence of safeguards and other system properties, models of insider threats may be built using survey questions that indicate which of these factors are present. Because the presence of safeguards is easy to determine, and the predisposing factors are easy to measure, the most significant barrier to conducting a survey is convincing responders to part with the data.

In contrast to insider threats, remote attacks via networks are less likely to be

[2]This correlation is why security practitioners rely upon the principle of least privilege, introduced by Saltzer and Schroeder in 1975, which states that “every program and every user of the system should operate using the least set of privileges necessary to complete the job” [89]. The principle of least privilege acts to minimize the number of adversaries the system faces in each threat scenario.

deterred by the introduction of safeguards that increase the risk of attack. Global networks, such as the Internet, enable adversaries to stage attacks with little risk of retribution. Whereas it was once the case that criminals could flee to foreign countries after a crime, a network enables an adversary to seek refuge first and then to attack from a safe distance. Foreign jurisdictions that do not consider the attack to be a crime, or may not have the interest or resources to help identify and extradite the perpetrator, are ideal for such purposes.

Staging an attack without being identified is preferable to staging an attack and avoiding prosecution. Network attackers regularly hide their identities by routing their communicating through a sequence of distant systems. The longer the trail, the harder it is to trace its source. If each step in the route lies in a different jurisdiction, and the trail can only be unravelled one step at a time, tracking the adversary will require gaining the cooperation of each jurisdiction. If enough steps lie between the adversary and the target, tracking the source of the attack is likely to take far more time than the attack itself. Unless meticulous logs are kept by all compromised parties, the trail will disappear before it can be traced.

When attacks come from a global network, risk of retribution is not likely to be a significant deterrent in the attacker's decision to target a system. A system's defenses can do little to increase the risk that an attacker will be identified, let alone punished. When an adversary can attack with impunity, the time, effort, and other resources required to stage such attacks become significant factors in the adversary's choice to target a system.

There are many ways to make systems stronger, and thus more difficult to attack.

Perhaps the most obvious is to ensure that all known vulnerabilities are countered with patches or other safeguards. While neutralizing known vulnerabilities will not stop all attacks, it will force outside attackers to undergo the potentially costly process of finding and exploiting new vulnerabilities. As those defending systems have become more vigilant in keeping their systems patched, attackers have started to employ *zero-day* exploits, which target new vulnerabilities discovered by these adversaries. Such attacks are expected to increase [57, 63, 68].

How well software stands up to attempts to find new vulnerabilities is a question of security strength. If regression models of software security are to be adapted for threat scenarios in which zero-day exploits are employed, the security risk model will need to include measures of security strength such as bounds (bid and ask) on the market price of a new vulnerability (MPV). Economic strength metrics also have the desirable property that their influence on the decisions of the adversary remains relatively consistent over time, whereas metrics that measure the presence of specific safeguards become meaningless as the strength of these safeguards fluctuates.

Unfortunately, today's software products are released without any measure of their strength, and thus without any measure of their resilience to the threat of network attack via zero-day exploits.

One tempting alternative to measuring security strength is to create models in which one or more independent variables are used to estimate it. Since strength is a function of the systems being used, one might propose creating an independent variable for each system and regressing to estimate strength. Because different versions have different strength, we would in fact need an independent variable for each version as well. Even if it were practical to perform regression analysis on functions of the thousands of indicator variables that would be required, the resulting model would only apply to the system versions for which they were measured. Not knowing the strength of a new version when it arrives, we could not understand how it would perform until its strength had once again been tested by time.

7.6 The growing significance of security strength

The decline in effectiveness of security strategies that leave vulnerabilities unguarded, and rely on risk to deter the adversary, is inevitable and already underway. Systems with unpatched vulnerabilities, for which exploits have been written, are now compromised within minutes of being exposed to the Internet [111]. The steady increase in network attacks seen in recent years is easily explained by the decline in both the risk and the cost of attack. The automation of attack by tools such as attack scripts has reduced the cost to attack systems. The steady increase in the number of vulnerabilities discovered, and left unpatched, has ensured a steady supply of attack scripts. As home users have acquired high speed, always-on, Internet access, their poorly guarded machines have become available for computer criminals to route traffic through, making it even easier for these attackers to lower their risk.

Is it reasonable to expect safeguards to be updated frequently enough to counter attacks on known vulnerabilities? Vulnerabilities have gone unpatched because the process of patching systems has been both labor intensive and poses its own risks [13]. In the past, patching has required administrators to watch mailing lists for patch announcements and to download and install the patches themselves. Risk is inevitable because introducing new code into a system in which it has not yet been tested may result in the creation of new vulnerabilities, or new flaws that may make cause a previously reliable system to fail. Because patches are hurriedly developed in the race to repair systems before exploits can target them, fears of patch-induced failure are not necessarily unjustified.

Beattie *et al.* [13] analyze the trade off between the choice to patch and the risk of leaving a vulnerability exposed. They describe how decisions can be made based on the time of discovery, the time in which an exploit is first seen, and the time at which a worm starts spreading. However, these risks are in constant flux. Staniford *et al.* [106] described how self-reproducing malware, or *worms*, can spread in a matter of minutes. This was later demonstrated by the Slammer worm and documented by Moore *et al.* [74]. If such a worm is released before an organization patches the vulnerability that the worm exploits, the organization will not have time to respond to patch the system before the worm infects the organization's systems. What's more, worm development kits promise to reduce the time and skill required to create a worm to exploit a new vulnerability. Hybrid pathogens now bypass organizations' firewalls by travelling through email like a virus (which, unlike worms, require human interaction to spread), then attack organizations from the inside exploiting vulnerabilities to once again spread like a worm (without human intervention).

As these network pathogens have grown more infectious, security strategies that don't include immediate patching have become less and less effective at reducing security risk. As the threat posed by worms continues to develop, the incentive for organizations to patch will only increase.

However, patching vulnerabilities is also becoming less burdensome on the organization. Operating systems such as Windows XP can now fetch a customized list of applicable patches automatically. Administrators no longer need to spend hours to read alert lists, check version numbers, download patches, verify their authenticity, and install them. The process can now take place without human intervention. Whereas it was once the case that only critical systems were patched, operating systems may soon to be shipped with automatic patching turned on by default [104].

Patches are not the only safeguards that can neutralize a vulnerability. Other safeguards, especially those that do not increase the risk of failure in systems that requires high reliability, may be desirable. Internal firewalls will protect systems from others on the same network. Proxies can be placed between clients and servers to interpret requests and filter out attacks before they can reach a system that would otherwise be vulnerable. For example, Microsoft's Shield project [116] is a network transport layer filter that removes network traffic that has the potential to exploit a known vulnerability. These filters can close vulnerabilities without the need to interrupt the execution of the vulnerable program or modify its code base.

When the vast majority of systems are protected from known vulnerabilities, adversaries will turn to zero-day exploits. These exploits require the adversary to obtain a new vulnerability – one that has not yet been reported and patched by the manufacturer. Once such an exploit has been put into widespread use, it is likely to be detected, reported, and patched in a matter of days. The adversary who discovers the vulnerability, and can control who else will have access to it during this window of time, will have the power to profit from it.

The only way to discourage such attacks against your systems is to make sure that you choose systems in which vulnerabilities are hard to find. These systems are those that have the greatest security strength. As a result, the decline in the availability of widely exploitable known vulnerabilities and the rise of the importance of using strong systems go hand in hand.

Even with accurate measures of software security strength, there is still a significant barrier to using regressions on organizational data to estimate security risk. Automated attacks, such as those using scripts, worms, or viruses, enable an adversary to indiscriminately target a large number of target systems and organizations at once. Even if an organization's systems contain little that is of value to outsiders, these system may still be struck. From the adversary's perspective, there may be no reason to assess the value of accessing the incentive to attack a system when he's already amortized away the cost of finding the vulnerability that gets him inside.

Thus, large scale attacks, especially those that have yet to be seen, must be modelled if we are to understand changes in the threat environment and the effect these change may have on security risk. This will be the topic of Chapter 8.

7.7 Chapter summary

While previously unnoticed by the computer security research community, models that forecast the security risks posed by a threat scenario are not only possible, but already available outside of computing in contexts such as home burglary. However, these models cannot be re-purposed for use in network and computer security without a significant overhaul. Remote attackers' targeting decisions are more likely to be influenced by the resources required to stage an attack (the system's security strength) than the personal risk to the attacker. Until security strength is measured, security risk models will suffer from omitted variable bias and are prone to rely upon statistical relationships that do not remain stationary.

Chapter 8

Anticipating new threats

The use of statistical models that use historical data alone to forecast future security risks can lead to a false sense of security. Left unaccounted for may be new technological advances or attack strategies that introduce unforseen threat scenarios. When security risk statistics are generated at an organizational level, the threat from scenarios that span multiple target organizations may also be underestimated. Of particular concern are attacks that are infrequent, making statistics hard to aggregate, but that target a large number of victims, making their potential effect on the user community significant. These threats have the potential to be quite profitable for our adversaries, as the cost of finding a vulnerability in a system can be amortized over a large number of target organizations in which the affected system has been installed.

For example, virus and worm authors have been known to unleash new pathogens for the purpose of using infected systems to relay spam, such as was the case with `Sobig.F` [103]. For each additional machine that is infected and controlled by the author, more spam can be relayed and more money collected from the author's clients. While the financial payoff from infecting any single machine or organization may be less than the cost to write the virus, the endeavor becomes quite profitable in aggregate because the virus need only be written once.

An organization cannot consider itself safe simply because it would cost a thief more to find and exploit a new vulnerability in the software used by the organization than the thief could gain by exploiting the vulnerability. The thief evaluates his potential financial gains in a larger context that includes all possible targets. The thief will estimate the desirability of obtaining and exploiting a new vulnerability based on the benefit of attacking all organizations that use the vulnerable software. Unless an organization can afford to build every system it uses from scratch, it must also measure its security in a global context.

To this end Michael D. Smith and I introduced *economic threat modelling*, a process for understanding threats posed by adversaries motivated by financial gain [92] for which little data is available. Our first such model, which I will present in this chapter, focuses on those thieves outside the target organization. The scenarios covered include those in which the outsider breaches security of many organizations via an unreported vulnerability in a software system. This model can then be used to estimate what these thieves are willing to pay for system vulnerabilities and how secure the system needs to be to make theft an unprofitable proposition.

I will also detail how the global nature of security also works to the advantage of the organization, since the community of all potential victim organizations has similar interests in thwarting attacks. For example, I argue that one way organizations can lower the damage resulting from unreported vulnerabilities is by sharing information about recent attacks. Monitoring companies and federally funded Information Sharing and Analysis Centers (ISACs) such as CERT can help organizations to collect and disseminate such information.

I explain the threat scenario of outside theft in Section 8.1 and introduce two subcategories (child scenarios). Serial theft, a scenario in which a criminal makes repeated uses of the same exploit to rob one victim after another, is analyzed in Section 8.2. Parallel theft, a scenario in which thieves automate the process of exploiting vulnerabilities in order to attack many systems at once, is addressed in Section 8.3. The chapter concludes in Section 8.4 with related work.

8.1 The threat of outside theft

One cannot determine how much security is required to protect against adversaries without making assumptions about who the adversaries are, what motivates them, and what resources are at their disposal. In this chapter I will focus primarily on thieves outside an organization. *Outside thieves* are individuals, or groups of individuals, who are unaffiliated with the target organization and who attack the organization's systems for financial gain. I assume that thieves, in contrast to adversaries such as terrorists, behave rationally in so far as they will not stage attacks that are expected to lead to their own financial loss.

Thieves have many ways to profit from exploiting system vulnerabilities. The most obvious method is for the thief to steal information, such as trade secrets, customer lists, credit card numbers, or cryptographic keys, and then sell that information to others. However, a thief may not actually need to take anything during an attack to create a situation where he can sell something for profit. For example, the thief may create back doors in the systems he attacks and then later sell that access to the highest bidder [91]. Alternatively, the thief may only change the state (e.g. a bank balance) on the target machines. Young and Yung [120] describe a state-changing attack that involves encryption of the data on the victim's machine; the thief profits by ransoming the decryption key.

8.1.1 Serial thieves

Serial thieves exploit an unreported vulnerability in a software program to attack system after system. By concentrating on one (or a small number) of victims at a time, the serial thief can carefully survey the value and location of what each victim is protecting and then maximize the loot obtained while also focusing on minimizing his risk of being detected.

A serial thief's crime spree ends when he is caught, when the vulnerability he has learned to exploit has been detected and patched on all target installations, or when the reward of committing another theft with this exploit no longer outweighs the risk to the thief of losing the loot he has already collected.

8.1.2 Parallel thieves

Parallel thieves automate their attack to penetrate a large number of targets at the same time. Automation has the clear appeal of multiplicatively increasing the potential loot beyond that obtainable by a more meticulous serial thief in the same time period. In fact, this may be the only practical way to rob a large number of victims if exploiting the vulnerability is likely to reveal it to the defense, and if once the vulnerability is revealed the manufacturer of the system is quick to release patches to repair the vulnerability.

There are, however, three costs that must be accounted for when automating attacks. First, an automated attack will need to locate valuables, such as which data to steal or modify, without human guidance. There is no doubt that valuables can often be located automatically (e.g., by looking for files of a particular type) but such an attack will not always capture the maximum value that could be obtained by attacking each target serially. Second, automating an attack requires the thief to make a limited set of assumptions about the vulnerability and its surrounding defenses. If the assumptions are incorrect, the system may repel a parallel attack when it would have failed to repel a more customized attack. The scale of the attack may significantly increase the chance that the thief is discovered and later punished. Finally, defensive systems are more likely to notice a flood of automated attacks that occur in a relatively short time period than they would be to detect a slow trickle. More and more intrusion detection tools, such as network telescopes [73], look for such statistically significant abnormalities in network traffic. Yet another risk of targeting a large number of systems is that, if a network or financial transaction is required to monetize the loot, the likelihood of detection will increase with each transaction.

8.2 Serial Theft

A target's attractiveness to a serial thief is the expected income to the thief from attacking it. This is in turn a function of the probability of success and the amount of loot that may be protected by the target. To simplify the initial analysis, I start by assuming that a thief will attack the most attractive targets, and that there is an unlimited number of homogeneous targets that are all equally attractive. We later extend the analysis by removing this assumption and modelling the unique qualities of each target.

8.2.1 Homogeneous Targets

I model the choices of the serial thief using a number of variables to represent properties of his environment. The motivation for each crime is the amount of loot that the thief expects to capture if successful.

ℓ	Loot, or the value obtained from a successful theft.

For every theft, there is a chance that the thief will be caught, convicted, and punished. We thus define the probability of being convicted and attempt to quantify in dollar terms the value of the punishment.

$\mathcal{P}_c$	Probability of being caught, convicted, and punished for the theft.

F	Fine paid by the thief if convicted, or the dollar equivalent of other punishment issued in the event of conviction, including the value of any confiscated loot.

Catching and convicting a serial thief may be difficult, especially if the thief is in a foreign jurisdiction. Another way to stop a thief who is taking advantage of an unknown vulnerability in a system is to detect how he is breaking in. Each time the thief uses an exploit to break into a system, there is a chance that an intrusion detection system or monitoring firm will be able to observe the thief's means of attack. Once the vulnerability is discovered, intrusion prevention systems can be trained to detect these attacks and the vulnerable system's manufacturer can distribute a patch. The serial thief will no longer be able to use this exploit to attack organizations that patch their systems.

$\mathcal{P}_d$	Probability that use of the exploit will expose it to the defense and the vulnerability will be patched as a result.

For simplicity, assume that if the thief is caught and convicted, his methods will be divulged. When $c \Rightarrow d$, $\mathcal{P}_c \leq \mathcal{P}_d$.

Finally, there is always a chance that a given attack will fail for reasons other than that the vulnerability was detected and a patch was put into place. Perhaps the target's intrusion detection system had learned just enough from reports of previous attacks at other targets to enable it to recognize a new attack. Perhaps a behavior-based intrusion detection system recognized the theft as unusual activity and stalled while monitoring personnel were notified.

$\mathcal{P}_f$	Probability that the attack fails, possibly because it is repelled before the criminal can reach the loot.

Note that detection and failure are independent events. An attack may succeed but the thief may be caught and convicted; an attack may fail but the thief may not be captured or convicted; or an attack may succeed and the thief may escape capture and conviction.

To simplify our notation, the probability that the exploit will not be divulged is written $\overline{\mathcal{P}}_d = 1 - \mathcal{P}_d$. The probability that the attack does not fail is written $\overline{\mathcal{P}}_f = 1 - \mathcal{P}_f$.

The expected profit from the ith theft, assuming the exploit has not yet been detected and patched, is the expected loot, $\overline{\mathcal{P}}_f \ell$, minus the expected fine if convicted, $\mathcal{P}_c F$:

$$\overline{\mathcal{P}}_f \ell - \mathcal{P}_c F$$

The expected profit from the ith and all additional thefts is labelled $E_{i\to\infty}$. We account for the additional value of the future thefts by adding the expected value of those thefts on the condition that the exploit can be used again, $\overline{\mathcal{P}}_d E_{i+1\to\infty}$.

$$E_{i\to\infty} = \left(\overline{\mathcal{P}}_f \ell - \mathcal{P}_c F\right) + \overline{\mathcal{P}}_d E_{i+1\to\infty}$$

Expanding reveals a common recurrence of the form $x^0 + x^1 + x^2 + \ldots$, which for $0 < x < 1$ is equal to $\frac{1}{1-x}$.

$$\begin{aligned} E_{i\to\infty} &= \left(\overline{\mathcal{P}}_f \ell - \mathcal{P}_c F\right) \cdot \left(1 + \overline{\mathcal{P}}_d + \overline{\mathcal{P}}_d^2 + \overline{\mathcal{P}}_d^3 + \ldots\right) \\ &= \left(\overline{\mathcal{P}}_f \ell - \mathcal{P}_c F\right) \cdot \frac{1}{1 - \overline{\mathcal{P}}_d} \end{aligned}$$

Thus the value of an exploit to a serial thief attacking homogeneous targets is:

$$E = E_{1\to\infty} = \frac{\overline{\mathcal{P}}_f \ell - \mathcal{P}_c F}{\mathcal{P}_d} \tag{8.1}$$

This result is quite revealing for the defense. It shows that even if you can't increase the chance of convicting a thief and can't thwart attacks that haven't been seen before, you can cut the thief's expected revenue in half by doubling the probability, $\mathcal{P}_d$, that you can detect how the thief is breaking in. That is, by doubling the probability that the vulnerability used in the attack will be revealed to the defense and subsequently repaired, the amount of loot that the thief can expect to extract from society is reduced by half. Deploying security tools to thwart attacks is also an effective deterrent. Halving the probability that an attack succeeds at each target will also reduce the value of the exploit, by at least if not more than half.

When evaluating different approaches for changing the probabilities in Equation 8.1, an organization should remember that customized security tools (i.e., ones not readily available to the adversary) are the ones that will be most effective. Unfortunately, these tools are also the most expensive to implement. Security tools that are mass produced may not be as effective for this purpose. If such packages are available to the thief then he will have had every opportunity to customize the exploit tool in an attempt to circumvent the defense and to avoid detection during an attack. This is a strong argument for employing more customized network monitoring systems, or monitoring firms that use custom systems, the behavior of which cannot be fully understood by outside adversaries.

8.2.2 Unique Targets

Instead of viewing all targets as homogeneous, now assume that each is unique. Each potential target organization has two goals in formulating its defense: to minimize the profitability of attack and to make itself the last target a thief would choose to attack. Action towards the first goal reduces the likelihood that thieves will attack. The justification for the second goal is that, should a spree of thefts occur, the later the organization falls in the set of desirable targets the more likely it is that the spree will end before the organization is attacked.

We label each target t_i where t_1 is the first target attacked, t_2 the second, and so on. An ordered set of targets $t_1, \ldots, t_i$ is written $\mathbf{T}_i$. The loot for each target is defined as ℓ_i and the punishment for the ith attack is F_i.

$\mathbf{T}_i = t_1, \ldots, t_i$	An ordered set of attack targets.

ℓ_i	The loot obtained by attacking target t_i.

F_i	The punishment if caught and convicted after the ith attack.

The probability that the vulnerability used to attack the system is detected is defined two different ways.

$\overline{\mathcal{P}}_d\left(t_i \vert \mathbf{T}_{i-1}\right)$	Probability that, if targets t_1 through t_{i-1} have been attacked, and the exploit has yet to be discovered, that it will not be discovered when t_i is attacked.

$\overline{\mathcal{P}}_d\left(\mathbf{T}_i\right)$	Probability that the thief can attack all targets, t_1 through t_i, in order, without the exploit being discovered and fixed.

The latter probability can be expressed inductively in terms of the former: the probability that all i attacks remain undetected is the probability that the first $i-1$ attacks remained undetected multiplied by the probability that the *ith* attack remains undetected.

$$\begin{aligned}
\overline{\mathcal{P}}_d\left(\mathbf{T}_i\right) &= \overline{\mathcal{P}}_d\left(t_i|\mathbf{T}_{i-1}\right)\overline{\mathcal{P}}_d\left(\mathbf{T}_{i-1}\right) \\
\overline{\mathcal{P}}_d\left(\mathbf{T}_1\right) &= \overline{\mathcal{P}}_d\left(t_1|\emptyset\right) = \overline{\mathcal{P}}_d\left(t_1\right) \\
\overline{\mathcal{P}}_d\left(\mathbf{T}_0\right) &= \overline{\mathcal{P}}_d\left(\emptyset\right) = 1
\end{aligned}$$

We also need functions to describe the probabilities that the thief will or won't be caught and that the attack won't fail.

$\mathcal{P}_c\left(t_i \vert \mathbf{T}_{i-1}\right)$	Probability that, if targets t_1 through t_{i-1} have already been attacked and the thief was not caught, the thief will be caught and convicted because he attacks t_i.

$\overline{\mathcal{P}}_f\left(t_i\middle\|\mathbf{T}_{i-1}\right)$	Probability that, if targets t_1 through t_{i-1} have already been attacked, the attack on t_i will *not* fail to produce loot.

The expected income from attacking unique targets is an extension of the same recurrence shown in the homogeneous target case.

$$\begin{aligned}
E_{1\to n} &= \overline{\mathcal{P}}_f\left(t_1\right)\ell_1 - \mathcal{P}_c\left(t_1\right)F_1 + \overline{\mathcal{P}}_d\left(t_1\right)E_{2\to n}\\
&= \overline{\mathcal{P}}_f\left(t_1\right)\ell_1 - \mathcal{P}_c\left(t_1\right)F_1\\
&\quad + \overline{\mathcal{P}}_d\left(t_1\right)\left[\overline{\mathcal{P}}_f\left(t_2|\mathbf{T}_1\right)\ell_2 - \mathcal{P}_c\left(t_2|\mathbf{T}_1\right)F_2 + \overline{\mathcal{P}}_d\left(t_2|\mathbf{T}_1\right)E_{3\to n}\right]\\
&= \overline{\mathcal{P}}_f\left(t_1\right)\ell_1 - \mathcal{P}_c\left(t_1\right)F_1\\
&\quad + \overline{\mathcal{P}}_d\left(t_1\right)\left[\overline{\mathcal{P}}_f\left(t_2|\mathbf{T}_1\right)\ell_2 - \mathcal{P}_c\left(t_2|\mathbf{T}_1\right)F_2\right]\\
&\quad + \overline{\mathcal{P}}_d\left(t_1\right)\overline{\mathcal{P}}_d\left(t_2|\mathbf{T}_1\right)E_{3\to n}\\
&= \overline{\mathcal{P}}_d\left(\mathbf{T}_0\right)\left[\overline{\mathcal{P}}_f\left(t_1|\mathbf{T}_0\right)\ell_1 - \mathcal{P}_c\left(t_1|\mathbf{T}_0\right)F_1\right]\\
&\quad + \overline{\mathcal{P}}_d\left(\mathbf{T}_1\right)\left[\overline{\mathcal{P}}_f\left(t_2|\mathbf{T}_1\right)\ell_2 - \mathcal{P}_c\left(t_2|\mathbf{T}_1\right)F_2\right]\\
&\quad + \overline{\mathcal{P}}_d\left(\mathbf{T}_2\right)E_{3\to n}
\end{aligned} \tag{8.2}$$

The recurrence simplifies to the following summation where n is the number of thefts.

$$E = E_{1\to n} = \sum_{i=1}^{n} \overline{\mathcal{P}}_d\left(\mathbf{T}_{i-1}\right)\left[\overline{\mathcal{P}}_f\left(t_i|\mathbf{T}_{i-1}\right)\ell_i - \mathcal{P}_c\left(t_i|\mathbf{T}_{i-1}\right)F_i\right] \tag{8.3}$$

As long as there are systems for which the thief's chosen vulnerability has yet to be patched, he will attack another target if the next term in Equation 8.3 is positive, increasing the expected profit. That is, attacking target t_{n+1} increases E so long as:

$$\overline{\mathcal{P}}_f\left(t_{n+1}|\mathbf{T}_n\right)\ell_{n+1} - \mathcal{P}_c\left(t_{n+1}|\mathbf{T}_n\right)F_{n+1} > 0 \tag{8.4}$$

How can an organization make itself a less attractive target for a serial thief? Most already know that if they have a good relationship with law enforcement, a thief will be more likely to be caught and face conviction. The likelihood of conviction is represented above via $\mathcal{P}_c$.

By taking account of the actions of past victims in our definition of $\mathcal{P}_c$, we can also quantify the effect of providing leads to law enforcement and other members of the defense that may not help the victim directly, but may lead to the thief's capture when others are attacked. For example, if a victim organization has a reputation for sharing information to help protect others, it can reduce the expected income to be gained by the thief by including the organization in the thief's set of targets. As a result, organizations that share information with others make less attractive targets than those that keep information to themselves.

Even if leads do not result in the thief being caught, they may still reduce the thief's expected income from future attacks. The defense may use information learned from one victim to help protect others or to allow the next victim to learn more about how it has been attacked. Such information might include suspicious traffic patterns or a port number the thief is suspected to have used. The results of such efforts would appear in our equations as decreases in $\overline{\mathcal{P}}_d(t|\mathbf{T})$ and $\overline{\mathcal{P}}_f(t|\mathbf{T})$ for potential future targets t.

Similarly, having a reputation for detecting exploits used and cutting off any future revenue from using such an exploit once it is patched will also deter future attacks. As organizations add intrusion detection and response capabilities, thus increasing $\mathcal{P}_f$, they will also make themselves less attractive targets.

An organization may wish to gauge its attractiveness to a thief in comparison with another organization and pose the question of who would be attacked first. Recall that the later an organization falls in a list of potential victims, the more likely it is that the string of thefts will be brought to an end before the organization is targeted. We can approximate the answer by viewing the world as if there were only two targets, a and b. We write that attacking b followed by a is expected to be more profitable than the reverse ordering by writing:

$$E_{b,a} > E_{a,b}$$

The expected profit from each ordering may be expanded.

$$\begin{aligned}
E_{b,a} &= \overline{\mathcal{P}}_f(t_b)\,\ell_b - \mathcal{P}_c(t_b)\,F_b + \overline{\mathcal{P}}_d(t_b)\left[\overline{\mathcal{P}}_f(t_a|t_b)\,\ell_b - \mathcal{P}_c(t_a|t_b)\,F_a\right] \\
E_{a,b} &= \overline{\mathcal{P}}_f(t_a)\,\ell_a - \mathcal{P}_c(t_a)\,F_a + \overline{\mathcal{P}}_d(t_a)\left[\overline{\mathcal{P}}_f(t_b|t_a)\,\ell_a - \mathcal{P}_c(t_b|t_a)\,F_b\right]
\end{aligned}$$

Given a cost c to find a vulnerability and develop an exploit, we can separate out the benefits and costs from the Equation 8.3 to write the equation for the prospective attacker's ROI.

$$\text{ROI}_{\text{attacker}} = \frac{\sum_{i=1}^{n} \overline{\mathcal{P}}_d(\mathbf{T}_{i-1})\,\overline{\mathcal{P}}_f(t_i|\mathbf{T}_{i-1})\,\ell_i}{c + \sum_{i=1}^{n} \overline{\mathcal{P}}_d(\mathbf{T}_{i-1})\,\mathcal{P}_c(t_i|\mathbf{T}_{i-1})\,F_i} \tag{8.5}$$

The likelihood that such an attack will be undertaken depends on the existence of attacks of similar scale with more favorable expectations on ROI. If we could collect enough data, we might be able to derive an expected frequency that a threat scenario will be realized given the scale of attack and its expected ROI.

Whether an organization will be effected is both a function of the frequency that the scenario will be realized and the likelihood that the organization will be targeted before the attacks stop. More research, including quantitative studies, will be required to determine the best techniques for estimating the frequency that an organization will be effected by an attack and integrating this into existing security risk statistics.

8.3 Parallel Theft

Thieves undertake parallel attacks in an attempt to penetrate as many systems as possible before the defense can construct a patch for the vulnerability. We assume that the parallel attack threat scenario ensures that even if the attack is detected, a patch cannot be created and deployed in time to prevent the remaining targets from being penetrated. Hence, the following analysis does not consider $\mathcal{P}_d$, as defined in the previous section.

Thieves benefit from every target attacked in so far as each target increases the total potential loot. For n targets, $t_1, t_2, \ldots, t_n$ define the marginal loot for the ith target as ℓ_i and refer to the total loot as L_n.

ℓ_i	Marginal potential loot from attacking the ith target.

$L_n = \sum_{i=1}^{n} \ell_i$	Total potential loot held by n targets.

Increasing the set of targets also increases the risk of being caught, convicted, and punished. Though detection of a parallel attack may not help the defense at the time of attack, it may prevent the thief from obtaining the loot (for example, if compromised machines are re-secured before the loot is taken) or enable the defense to recover all of the stolen loot at the time of conviction. Thus, a failure or conviction that results from an attack on a single target will often result in the loss of the loot from all targets. Our analysis incorporates this assumption by setting the total potential loot, L_n, and the total fine paid if caught, F, to be equal.

Given this assumption, one should not think about the probability of failure, $\mathcal{P}_f$, and the probability of conviction, $\mathcal{P}_c$, as separate probabilities, but instead simply consider the marginal risk for the ith target as the single quantity r_i. I will refer to the probability that the thief successfully captures all of the loot as P_n.

r_i	Marginal increase in the probability that all loot is lost due to the attack on the ith target.

$P_n = 1 - \sum_{i=1}^{n} r_i$	Probability that an attack on n targets succeeds and does not lead to detection or conviction.

Using these definitions, the expected profit from the attack, E_n, is the potential loot L_n times the probability that the attack succeeds, P_n.

$E_n = P_n \cdot L_n$	Expected profit from the attack.

The expected profit from an additional attack would take into account the marginal loot and marginal risk.

$$E_{n+1} = P_{n+1}L_{n+1} = (P_n - r_{n+1})\,(L_n + \ell_{n+1})$$

The thief benefits from expanding the set of targets by one if $E_{n+1} > E_n$.

$$\begin{aligned}
E_{n+1} &> E_n \\
(P_n - r_{n+1})(L_n + \ell_{n+1}) &> P_n L_n \\
P_n L_n + P_n \ell_{n+1} - r_{n+1} L_n - r_{n+1}\ell_{n+1} &> P_n L_n \\
P_n \ell_{n+1} - r_{n+1}\ell_{n+1} &> r_{n+1} L_n \\
P_{n+1}\ell_{n+1} &> r_{n+1} L_n
\end{aligned}$$

The equilibrium point that describes the number of targets attacked, n, is the point at which:

$$P_n \approx \frac{r_n L_{n-1}}{\ell_n} \tag{8.6}$$

Once the equilibrium point is found, the resulting loot and risk summations, L_n and P_n, can be multiplied to find the expected profit from the attack E_n. Theft is a losing proposition if E_n is less than the cost to find and exploit a new vulnerability. Even if the thief is expected to profit from finding a vulnerability in the system, a given organization may avoid attack if the thief perceives the marginal risk of attacking that organization to be too high.

However, most current automated attacks, such as worms and viruses, do not discriminate on the targets that they attack. Instead, those who control worms and viruses later decide which of the attacks they want to monetize with further action. Even if an system is not specifically targeted, it may still suffer damage. For example, an individual may lose all his data to a cryptovirus even if the author of the malware has no intention of risking a small transaction to sell the individual his data back. For threat scenarios such as these indiscriminate worm and virus attacks, we must assume that the probability that a vulnerable system will be damaged in the attack is 1. The likelihood that a system will be damaged is thus only a function of whether the attack occurs and whether it succeeds in affecting the system in question. To estimate the likelihood that an attack occurs, ROI can be calculated as the benefit E, calculated at the equilibrium state, over the cost of obtaining a vulnerability and writing the exploit.

To help develop intuition for our equilibrium state, consider the case in which all targets contain the same amount of loot ℓ. We can then revise Equation 8.6 by taking into account that $\ell_n = \ell$ and $L_{n-1} = (n-1)\ell$.

$$\begin{aligned} P_n &\approx \frac{r_n \cdot (n-1)\ell}{\ell} \\ P_n &\approx (n-1)r_n \end{aligned}$$

Dividing by the marginal risk yields the equilibrium state for the case of homogeneous loot:

$$n = \frac{P_n}{r_n} + 1 \tag{8.7}$$

The conclusion drawn from both the distinct and homogeneous loot cases is the same: maximizing the marginal risk of each additional attack is key to deterring the parallel thief.

The marginal risk of an additional attack will be low if organizations are defended by homogeneous defenses (packaged systems), as once one system using such a defense is attacked the marginal risk of attacking another system that uses this same defense exclusively will be extremely small. Using a set of systems, each customized to have unique properties, will be much more effective in keeping marginal risk high. This is not surprising as animal species use a diversity of defense mechanisms, customized in each organism, to survive potent threats from parallel attacks by microorganisms.

Sharing of information by potential targets can also act to increase the thief's marginal risk of attacking an additional system. If the body of knowledge of each member of the defense grows with the number of targets attacked, so will the marginal risk of attack. If organizations do not share information, the body of knowledge of each one will be constant and will not affect marginal risk. If they do share information, then the potential knowledge of each potential victim grows with each attack.

As mentioned above, the thief must not only avoid being traced while breaking in, but must also retrieve the loot and exchange it for payment. To keep his marginal risk of being discovered and convicted low, the thief will need to be able to monetize attacks as anonymously as possible. The thief will only benefit from anonymous digital cash systems if these systems do not contain back doors through which authorities can revoke anonymity. Anonymous networking techniques can be used to protect a limited amount of data traffic between the thief and target systems, but as the amount of data grows so does the likelihood that traffic analysis will expose the thief's location. Thieves may try to mitigate these risks by finding ways to group cash transactions or reduce the bandwidth and number of network transactions. For example, the author of a cryptovirus could force multiple victims choose a single delegate to transact on their behalf.

Once marginal risk has been addressed, the remaining tool in thwarting the parallel attack is limiting the amount of loot lost to the thief. Backing up data both frequently and securely is an excellent way to limit the risk posed by data integrity attacks, such as data ransom (cryptoviruses) and bank balance manipulation.

8.4 Further reading

The study of the economics of crime owes a great deal to Becker, who wrote the seminal paper on the subject nearly a quarter century ago [14]. Among his contributions is a model for the expected utility of committing a crime that serves as a foundation for this chapter. He defines this value by adding to the utility of the crime the product of the utility of the punishment (a negative number) and the probability of conviction.

Ehrlich's work [35] examines both a theory of crime as occupational choice and empirical evidence for this theory. In later work [36] he proposes a market model of criminal offenses, where the supply of offenses is a function of the benefits and costs of committing them, including the opportunity cost lost from legitimate labor.

The question of whether it behooves an organization to report information when it is attacked has been posed by Gordon, Loeb, and Lucyshyn [54] in the context of Information Sharing and Analysis Centers (ISACs) such as CERT. A similar question was previously addressed in the context of reporting of household burglaries by Goldberg and Nold [49]. Their empirical analysis found that those households that contained indicators that they would report burglaries were less likely to be robbed.

In Section 8.2 our model suggests that reporting forensic information discovered in response to an attack by a serial thief should help to deter future theft.

8.5 Chapter summary

I have introduced a model for estimating the value of a software vulnerability and exploit to an outside thief. This model takes into account investments in intrusion detection and response, both internally and by outside monitoring firms. Using the model, an organization can gauge its attractiveness to outside thieves and determine how much security is required in the software it purchases. Beyond choosing how much to spend on security, analysis of the social aspects of intrusion detection and response strategies, such as information sharing, can be evaluated for their effectiveness in deterring future attacks.

One of our first applications of this approach was to model the threat of worms and viruses that opened up back doors on infected hosts [92, 91]. The premise was that the author of the worm or virus could sell access to these hosts to the highest bidder, or use the hosts for organized criminal activities. We showed that this threat scenario was attractive to a virus or worm author as it posed little personal risk and had a high potential payoff. Following publication, these threats have begun to emerge in the form of viruses, such as `Sobig.f` [103], that have used back doors on infected hosts to relay spam.

Chapter 9

Conclusion

The dearth of quantitative methods for measuring, forecasting, and improving computer security has left those of us who depend on information systems in a precarious state. The software we rely upon is developed and tested without any meaningful measure of its security strength. Those of us charged with securing systems are unable to differentiate more secure software products from less secure ones. Even if we knew that one software product was stronger than its alternative, we have lacked tools with which to forecast the financial benefits of choosing the stronger product. Because quantitative metrics have not been available, our security decisions have instead relied upon the opinions of those believed to be experts, anecdotal evidence, and other heuristics.

While security will never be an exact science, I have shown in this dissertation it is possible to answer questions of security with quantitative metrics that provide answers in meaningful terms. These techniques lay groundwork that can be used for quantitative and economic studies of security. I described how, given inputs that are measurable and stationary, statistical models can be built to forecast security risk. These forecasts can measure the benefits of different prospective security investments or strategies, and then be used in a cost-benefit analysis to better optimize our security decisions.

I argued that models of security risk for networked systems must measure the effects of deterrents other than the personal risk of capture or harm posed to those who attack these systems. While measuring the risk retribution is sufficient when modelling deterrents to certain threat scenarios, as it was for home burglary, computer networks provide a vector of attack for which the level of risk may become small enough to be overshadowed by other deterrents. Security strength is the measure of these other deterrents, which include the cost of time and resources to carry out a threat scenario.

Security strength is a familiar concept to cryptographers, who have long estimated the computational costs to break codes and forge signatures. Security strength is important to the security of encrypted messages because eavesdropping can often be achieved with low risk. Once a message is intercepted a cryptanalyst can attempt to decipher it from a safe location. If one believes the computational assumptions on which a cryptosystem is built, then one believes that the strength of that cryptosystem can be increased exponentially with a small increase in the key size and computational costs. Because key size is the primary choice that needs to be made when choosing a cryptosystem, it has not been necessary to perform detailed analysis of the trade-offs between security risk and the costs of increasing security strength.

Increasing the strength of a general software system is far more expensive than increasing a cryptosystem's key length. It is thus important to understand how security strength influences the security risk faced by software systems. In order to accomplish this, it was first necessary to redefine security strength in the context of software systems and determine how it could be measured.

I introduced an economic measure of software security strength, with respect to a threat scenario, in Chapter 4. The key observation was that security strength depended most on whether the adversary knew of the existence of an exploitable vulnerability in the system. If a vulnerability is known and can be exploited cheaply, the security strength of the vulnerable system is negligible. If no vulnerability is known, the security strength of the system is bounded by the cost to find a new vulnerability and to develop a means for exploiting the vulnerability to accomplish one's goals. For systems without known vulnerabilities, I used the market price to find a new vulnerability (MPV) as a measure of security strength.

I provided a means of establishing upper and lower bounds for MPV, using markets in which new vulnerabilities reports can be purchased or sold. These bounds can be used to differentiate products that have stronger protection against a threat scenario from those that are weaker. One can also use these bounds to estimate security strength, and these estimates can be input into models that forecast security risks.

There will be little value to measuring security strength if stronger software can't be developed at a reasonable price. By extending the market approach to measure security strength throughout the development process, the techniques I have introduced can enable software developers to improve security strength before their products are released. This enables firms to make trade-offs between their product's testing costs, testing time before release, and its security strength at the time of release.

Finally, I have introduced economic threat models for forecasting the potential risks posed by new threat scenarios for which the data required for regression models does not exist. I included the analysis of two outsider threat scenarios, serial attack and parallel attack. These approaches have already proven their value in predicting the arrival of 'back doors' in viruses and worms used for financial gain.

With this groundwork in place, many new research opportunities await. Many threat scenarios await regression models for forecasting security risk. Economic threat models should be better integrated with these models. Processes for measuring security strength must be put into place before regression models of security risk, which require this indicator, can be constructed.

While there is much work to be done, software developers can begin using the tools presented in this dissertation to formalize the list of threat scenarios they expect their software to address. They can increase the strength of these systems as they develop them by using market methods, and release systems with measured levels of strength.

By signalling our intent to factor security strength into our software purchasing decisions, those of who depend on our software to be secure can encourage software vendors to apply this quantitative approach.

Appendix A

A brief history of fault, threat, and attack trees

Fault trees are the oldest of the tree based methods, dating back to their invention in 1961 by H. A. Watson [117], sometimes credited with A. Mearns [64], at Bell Laboratories for a contract to study the launch control system for the Minuteman missile.

In his superb history of fault trees, Clifton Ericson [64] describes how the use of Fault Tree Analysis (FTA) took off.

> Dave Haasl, then at the Boeing Company, recognized the value of this tool and led a team that applied FTA to the entire Minuteman Missile System. Other divisions within Boeing saw the results from the Minuteman program and began using FTA during the design of commercial aircraft. In 1965 Boeing and the University of Washington sponsored the first System Safety Conference.

Ericson goes on to describes how, after the fatal Apollo 1 launch pad accident of 1967, NASA adopted fault tree analysis for the entire Apollo system. After a system fails, fault trees may be used to determine what events could cause the failure. Ericson describes how fault trees were thus used in analyzing what went wrong at Three Mile Island nuclear power plant and with the space shuttle Challenger [64]. More recently, NASA has used fault tree analysis yet again in its investigation of the loss of the space shuttle Columbia [12].

While the earliest papers on fault trees are hard to come by, Fussell's work on the construction and application of fault trees [46, 47] are excellent examples of early work. The Nuclear Regulatory Commission's fault tree handbook [115], from the early 1980s, reflects the maturation of this research area. A bibliography of fault tree research is available from NASA [75], but the selection of references may be biased towards those works that are relevant to the space agency.

Threat trees are described in depth by Amoroso [4], who recognizes their heritage from fault trees and cites previous work by the U.S. Department of Defense in the late 1980s [77] and Jonathan Weiss in the early 1990s [118]. More recently, the technique has been popularized by a chapter in Microsoft's text, *Writing secure code*, by Howard and LeBlanc [62], and its further treatment in Ross Anderson's text *Security Engineering* [8].

Schneier first described attack trees in collaboration with others in 1998 [88], which was followed by an article in Dr. Dobb's Journal [93] and a chapter in his book, *Secrets & Lies: Digital Security in a Networked World* [94]. In Schneier's attack trees, the events at each node are referred to as goals. This reflects Schneier's approach of taking the adversary's perspective when constructing trees. Whereas fault trees originated from the study of safety where events were the result of natural causes, in a security perspective the base threat event at the root of the tree is the adversaries goal.

A more detailed comparison of fault, threat, and attack trees is presented in the dissertation of Nathalie Foster [44], in which she contends that threat trees and attack trees are specific types of fault trees. Foster creates her own trees, which she calls RAE trees, for use in protocol analysis. The examples she provides are some of the most extensively detailed trees available.

Bibliography

[1] George A. Akerlof. The market for 'lemons': Quality, uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84:488–500, August 1970.

[2] Christopher J. Alberts and Audrey J. Dorofee. An introduction to the OCTAVE^{SM} method. http://www.cert.org/octave/methodintro.html, January 30, 2001.

[3] Christopher J. Alberts and Audrey J. Dorofee. *Managing Information Security Risks: The OCTAVE^{SM} Approach.* Addison-Wesley, June 2002.

[4] Edward G. Amoroso. *Fundamentals of computer security technology.* Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[5] Alison Anderson, Dennis Longley, and Lam For Kwok. Security modeling for organisations. In *Proceedings of the 1994 ACM Conference on Computers and Communications Security*, November 1994.

[6] Alison M. Anderson. Comparing risk analysis methodologies. In David T. Lindsay and Wyn L. Price, editors, *Proceedings of the IFIP TC11, Seventh International Conference on Information Security, IFIP/Sec '91*, IFIP Transactions, pages 301–311. Elsevier, May 15–17, 1991.

[7] Ross J. Anderson. Why cryptosystems fail. *Communications of the ACM*, 33(11):32–40, November 1994.

[8] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley & Sons, Inc., first edition, 2001.

[9] Ross J. Anderson. Why information security is hard, an economic perspective. In *17th Annual Computer Security Applications Conference*, December 2001.

[10] James Bamford. *The Puzzle Palace: A Report on America's Most Secret Agency.* Penguin Books, New York, NY, 1983.

[11] James Bamford. *Body of Secrets: Anatomy of the Ultra-Secret National Security Agency.* Random House, New York, NY, April 2001.

[12] Jim Banke. Columbia investigation enters new phase, air force picture adds intrigue. *SPACE.COM*, February 7, 2003.

[13] Steve Beattie, Seth Arnold, Crispin Cowan, Perry Wagle, Chris Wright, and Adam Shostack. Timing the application of security patches for optimal uptime. In *Proceedings of The 16th USENIX Systems Administration Conference (LISA 2002)*, November 3–8, 2002.

[14] Gary S. Becker. Crime and punishment: An economic approach. *The Journal of Political Economy*, 76(2):169–217, March–April 1968.

[15] Bob Blakley. An imprecise but necessary calculation. *Secure Business Quarterly: Special Issue on Return on Security Investment*, 1(2), Q4, 2001. A publication of @stake.

[16] Bob Blakley. The measure of information security is dollars. In *The First Workshop on Economics and Information Security*, May 16-17, 2002.

[17] Robert M. Brady, Ross J. Anderson, and Robin C. Ball. Murphy's law, the fitness of evolving species, and the limits of software reliability. http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/babtr.pdf, 1999.

[18] Phillip J. Brooke and Richard F. Paige. Fault trees for security system design and analysis. *Computers & Security*, 22(3):256–264, 2003.

[19] Hilary K. Browne, William A. Arbaugh, John McHugh, and William L. Fithen. A trend analysis of exploitations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 214–229, May 2001.

[20] Tracey Budd. Burglary of domestic dwellings: Findings from the british crime survey. Technical report, United Kingdom Home Office Crime Reduction Programme Unit, April 1999.

[21] Shawn A. Butler, P. Chalasani, Somesh Jha, Orna Raz, and Mary Shaw. The potential of portfolio analysis in guiding software decisions. In *Proceedings of the First Workshop on Economics-Driven Software Engineering Research (EDSER-1)*, May 1999.

[22] William J. Caelli, Dennis Longley, and Alan B. Tickle. A methodology for describing information and physical security architectures. In Guy G. Gable and William J. Caelli, editors, *Proceedings of the IFIP TC11, Eigth International Conference on Information Security, IFIP/Sec '92*, volume A-15 of *IFIP Transactions*, pages 277–296. Elsevier, May 27–29, 1992.

[23] L. Jean Camp and Catherine Wolfram. Pricing security. In *Proceedings of the CERT Information Survivability Workshop*, pages 31–39, October 24-26, 2000.

[24] Graham Chapman, John Cleese, Eric Idle, Terry Gilliam, Terry Jones, and Michael Palin. Monty Python and the Holy Grail, 1975.

[25] Leonardo Chiariglione and The Secure Digital Music Initiative. An open letter to the digital community. http://www.sdmi.org/pr/OL_Sept_6_2000.htm, September 6, 2000.

[26] Clifford C. Cocks. A note on 'non-secret encryption'. Technical report, British Communications-Electronics Security Group (CESG), November 20, 1973.

[27] Computer Security Institute. Fourth annual CSI/FBI computer crime and security survey, 1998.

[28] Computer Security Institute. Fifth annual CSI/FBI computer crime and security survey, 1999.

[29] Computer Security Institute. Eigth annual CSI/FBI computer crime and security survey, 2003.

[30] Don Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *Journal of Cryptology*, 10(4):233–260, November 1997.

[31] Scott Craver, Patrick McGregor, Min Wu, Bede Liu, Adam Stubblefield, Ben Swartzlander, Dan S. Wallach, and Edward W. Felten. SDMI challenge FAQ. http://www.cs.princeton.edu/sip/sdmi/faq.html.

[32] Scott A. Craver, Min Wu, Bede Liu, Adam Stubblefield, Ben Swartzlander, Dan W. Wallach, Drew Dean, and Edward W. Felten. Reading between the lines: Lessons from the SDMI challenge. In *Proceedings of The 10th USENIX Security Symposium*, August 13–17, 2001.

[33] Dorothy E. Denning. The limits of formal security models. National Computer Systems Security Award Acceptance Speech, October 18, 1999.

[34] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[35] Isaac Ehrlich. Participation in illegitimate activities: A theoretical and empirical investigation. *The Journal of Political Economy*, 81(3):521–565, May–June 1973.

[36] Isaac Ehrlich. Crime, punishment, and the market for offenses. *The Journal of Economic Perspectives*, 10(1):43–67, Winter 1996.

[37] Electronic Frontier Foundation. Security researchers drop scientific censorship case. February 6, 2002. http://www.eff.org/IP/DMCA/Felten_v_RIAA/20020206_eff_felten_pr.html

[38] James H. Ellis. The possibility of non-secret encryption. Technical report, British Communications-Electronics Security Group (CESG), 1970.

[39] James H. Ellis. The history of non-secret encryption. Technical report, British Communications-Electronics Security Group (CESG), 1987.

[40] Carl Ellison and Bruce Schneier. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal*, 16(1), 2000.

[41] Mike Fisk. Causes & remedies for social acceptance of network insecurity. In *The First Workshop on Economics and Information Security*, May 16-17, 2002.

[42] Philip E. Fites, Martin P. J. Kratz, and Alan F. Brebner. *Control and Security of Computer Inforamtion Systems*. Computer Sccience Press, Inc., Rockville, MD, 1989.

[43] The Association for Payment Clearing Services (APACS). Card fraud – the facts 2002. http://www.cardwatch.org.uk/html/card_fraud_facts.html, 2002.

[44] Nathalie Louise Foster. *The application of software and safety engineering techniques to security protocol development*. PhD thesis, University of York, September 2002.

[45] Lady Antonia Fraser. *Mary Queen of Scots*. Random House, London, UK, 1989.

[46] Jerry B. Fussell. A formal methodology for fault tree construction. *Nuclear Science and Engineering*, 52(1), September 1973.

[47] Jerry B. Fussell, Gary J. Powers, and R G. Bennetts. Fault trees—a state of the art discussion. *IEEE Transactions on Reliability*, R-23(1):51–55, April 1974.

[48] Daniel E. Geer, Jr. Making choices to show ROI. *Secure Business Quarterly: Special Issue on Return on Security Investment*, 1(2), Q4, 2001. A publication of @stake.

[49] Itzhak Goldberg and Frederick C. Nold. Does reporting deter burglars?–an empirical analysis of risk and return in crime. *The Review of Economics and Statistics*, 62(3):424–431, August 1980.

[50] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. http://www.cs.ucsd.edu/users/mihir/papers/gb.pdf, April 2001.

[51] Shafi Goldwasser and Sylvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, April 1984.

[52] Lawrence A. Gordon and Martin P. Loeb. The economics of information security investment. *ACM Transactions on Information and System Security*, 5(4):438–457, November 2002.

[53] Lawrence A. Gordon and Martin P. Loeb. Return on information security investments: Myths vs. realities. *Strategic Finance*, pages 26–31, November 2002.

[54] Lawrence A. Gordon, Martin P. Loeb, and William Lucyshyn. An economics perspective on the sharing of information related to security breaches: Concepts and empirical evidence. In *The First Workshop on Economics and Information Security*, May 16-17, 2002.

[55] Peter Gutmann. PKI: It's not dead, just resting. *IEEE Computer*, 35(8), August 2002.

[56] Simon Hakim, George F. Rengert, and Yochanan Shachmurove. Knowing your odds: Home burglary and the odds ratio. Technical Report CARESS Working Paper 00-14, University of Pennsylvania Center for Analytic Research in Economics and the Social Sciences, September 2000.

[57] Eric Hines. Analysis of the ntdll.dll WebDAV exploit. Technical report, Fate Research Labs, March 25, 2003.

[58] F. H. Hinsley and Alan Stripp, editors. *Codebreakers: The inside story of Bletchley Park*. Oxford University Press, Oxford, UK, 1993.

[59] Kevin J. Soo Hoo. *How Much Is Enough? A Risk-Management Approach to Computer Security*. PhD thesis, Stanford University, June 2000.

[60] Kevin J. Soo Hoo, Andrew W. Sudbury, and Andrew R. Jaquith. Tangible ROI through secure software engineering. *Secure Business Quarterly: Special Issue on Return on Security Investment*, 1(2), Q4, 2001. A publication of @stake.

[61] John D. Howard. *An Analysis Of Security Incidents On The Internet: 1989 - 1995*. PhD thesis, Carnegie Mellon University, April 7, 1997.

[62] Michael Howard and David LeBlanc. *Writing secure code*. Microsoft Press, Redmond, Washington, 2nd edition, December 2002.

[63] Reorge V. Hulme. Zero-day attacks expected to increase. *Information Week*, March 24, 2003.

[64] Clifton A. Ericson II. Fault tree analysis – a history. In *Proceedings of the 17th International System Safety Conference*, August 16–21, 1999.

[65] Robert V. Jacobson. What is a rational goal for security? *Security Management*, 44, December 2000.

[66] Joseph. E. Kaufmann, H. W. Kauffmann, and Robert M. Jurga. *The Medieval Fortress: Castles, Forts and Walled Cities in the Middle Ages.* DaCapo Press, 2001.

[67] Nina Lewis. Workshop framework response. In *Proceedings of the 3rd International Computer Security Risk Model Builders Workshop.* National Institute of Standards and Technology (NIST), August 21–23 1990.

[68] Shawna McAlearney. Zero-day IE exploit just the beginning. *Information Security Magazine: Security Wire Digest*, 5(74), October 2, 2003.

[69] Robert I. Mehr and Bob A. Hedges. *Risk Management: Concepts and Applications.* Richard D. Irwin, Inc., Homewood, Illinois, USA, 1974.

[70] Lindsay C. J. Mercer. Fraud detection via regression analysis. *Computers & Security*, 9(4), June 1990.

[71] Soumyo D. Moitra and Suresh L. Konda. A simulation model for managing survivability of networked information systems. Technical Report ESC-TR-2000-020, Carnegie Mellon Software Engineering Institute, December 2000.

[72] Sournyo D. Moitra and Suresh L. Konda. The survivability of network systems: An empirical analysis. Technical Report CMU/SEI-2000-TR-021, Carnegie Mellon Software Engineering Institute, December 2000.

[73] David Moore. Network telescopes: Observing small or distant security events. Invited Talk at the 11th USENIX Security Symposium, August 8, 2002.

[74] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the Slammer worm. *IEEE Security and Privacy*, 1:33–39, July 2003.

[75] National Aeronautics and Space Administration. *Fault Tree Analysis: A Bibliography.* National Aeronautics and Space Administration (NASA) Scientific and Technical Information Program, Hanover, Maryland, July 2000.

[76] National Institute of Standards and Technology (NIST). An introduction to computer security: The NIST handbook. National Insitute of Standards and Technology (NIST) Special Publication 800-12, U.S. Government Printing Office, October 1995.

[77] Department of Defense. System security engineering program management requirements. MIL-STD-1785, June 20, 1988.

[78] Matthew J. Oppenheim, Recording Industry Association of America, and The SDMI Foundation. Letter to professor edward felten. http://www.cs.princeton.edu/sip/sdmi/riaaletter.html, April 9, 2001.

[79] Thomas R. Peltier. *Information Security Risk Analysis.* CRC Press LLC, 2000 N.W. Corporate Blvd., Boca Raton, Floirda 33431, 2001.

[80] John Arthur Thomas Pritchard. *Risk Management in Action.* NCC Publications, The National Computing Centre Limited, Oxford Road, Manchester M1 7ED, England, 1978.

[81] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for computer science, January 1979.

[82] Robert Richardson. personal correspondence, November 2003.

[83] Ronald L. Rivest, Adi Shamir, and Len A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[84] RSA Security. The new RSA factoring challenge. http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html.

[85] RSA Security. The RSA challenge numbers. http://www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html.

[86] RSA Security. The RSA factoring challenge FAQ. http://www.rsasecurity.com/rsalabs/challenges/factoring/faq.html.

[87] RSA Security. The RSA laboratories secret-key challenge. http://www.rsasecurity.com/rsalabs/challenges/secretkey/index.html.

[88] Chris Salter, O. Sami Saydjari, Bruce Schneier, and Jirn Wallner. Toward a secure system engineering methodology. In *Proceedings of The New Security Paradigms Workshop.* ACM Press, September 22–25, 1998.

[89] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1728–1308, September 1975.

[90] Stuart E. Schechter. Quantitatively differentiating system security. In *The First Workshop on Economics and Information Security*, May 16-17, 2002.

[91] Stuart E. Schechter and Michael D. Smith. Access for sale: A new class of worm. In *Proceedings of the Workshop on Rapid Malcode (WORM)*, October 27, 2003.

[92] Stuart E. Schechter and Michael D. Smith. How much security is enough to stop a thief? The economics of outsider theft via computer systems and networks. In George Davida, Yair Frankel, and Owen Rees, editors, *Lecture Notes in Computer Science: Proceedings of the Seventh Financial Cryptography Conference*, volume 2742, pages 73–87. Springer-Verlag, LNCS 2437, January 27-30, 2003.

[93] Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, December 1999.

[94] Bruce Schneier. *Attack Trees*, chapter 21, pages 318–333. In [95], 2000.

[95] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World.* John Wiley & Sons, 2000.

[96] Bruce Schneier. No, we don't spend enough! In *The First Workshop on Economics and Information Security*, May 16-17, 2002.

[97] E. Eugene Schultz. A framework for understanding and predicting insider attacks. *Computers & Security*, 21(6), October 2002.

[98] Yochanan Shachmurove, Gideon Fishman, and Simon Hakim. The burglar as a rational economic agent. Technical Report CARESS Working Paper 97-07, University of Pennsylvania Center for Analytic Research in Economics and the Social Sciences, June 1997.

[99] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[100] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:398–403, 1948.

[101] Simon Singh. *The code book: the evolution of secrecy from Mary Queen of Scots to quantum cryptography.* Random House, New York, 1999.

[102] Alan Gordon Smith. *The Babington Plot.* Macmillan, London, UK, 1936.

[103] `W32.Sobig.F@mm`.
http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.f@mm.html.

[104] Scott Spanbauer. Microsoft's patch policy pickle: Blaster attacks could prompt default enablement of automated updates. *PC World*, November 2003.

[105] Mosler Anti-Crime Bureau Staff. Risk management: A systems approach. In Robert R. Rosberg, editor, *A Practitioners Guide for Security Risk Management.* Dorison House Publishers, Inc., 824 Park Square Buliding, Boston, Massachusetts 10016, 1980.

[106] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, August 7–9, 2002.

[107] A. Francis Steuart, editor. *Trial of Mary Queen of Scots.* Wiliam Hodge, London, UK, 1951.

[108] James H Stock and Mark W. Watson. *Introduction to Econometrics.* Pearson Education, Boston, MA, 2003.

[109] Gary Stoneburner, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems: Recommendations of the national institute of standards and technology. National Insitute of Standards and Technology (NIST) Special Publication 800-30, U.S. Government Printing Office, October 2001.

[110] Detmar W. Straub. Effective IS security: An empirical study. *Information Systems Research*, 1(3):255–276, January 16, 1990.

[111] The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community.* Addison-Wesley, 2001.

[112] United States Congress. Computer security act of 1987. Public Law 100-235 (H.R. 145), January 8, 1988.

[113] Hal R. Varian. Managing online security risks. *The New York Times*, June 1, 2000.

[114] Hal R. Varian. System reliability and free riding. In *The First Workshop on Economics and Information Security*, May 16-17, 2002.

[115] William E. Vesely, F. F. Goldberg, Norman H. Roberts, and David F. Haasl. *Fault Tree Handbook.* Systems and Reliability Research Office of Nuclear Regulartory Research, U.S. Nuclear Regulatory Commission, Washington, D.C. 20555, January 1981.

[116] Helen J. Wang, Chuanxiong Guo, Daniel R. Simon, and Alf Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. Technical Report MSR-TR-03-81, Microsoft Research, February 2004.

[117] H. A. Watson. Launch control safety study. Technical report, Bell Telephone Laboratories, Murray Hill, New Jersey, 1961.

[118] Jonathan D. Weiss. A system security engineering process. In *Proceedings of the 14th National Computer Security Conference*. National Institute of Standards and Technology/National Computer Security Center, October 1–4, 1991.

[119] Richard T. Wright and Scott H. Decker. *Burglars on the Job: Streetlife and Residential Break-ins.* Northeastern University Press, Boston, 1994.

[120] Adam Young and Moti Yung. Cryptovirology: Extortion-based security threats and countermeasures. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 129–140, May 6-8, 1996.