Bug Auctions: Vulnerability Markets Reconsidered

Andy Ozment
Computer Laboratory
University of Cambridge
Andy.Ozment@cl.cam.ac.uk

Abstract

Measuring software security is difficult and inexact; as a result, the market for secure software has been compared to a 'market of lemons.' Schechter has proposed a vulnerability market in which software producers offer a time-variable reward to free-market testers who identify vulnerabilities. This vulnerability market can be used to improve testing and to create a relative metric of product security. This paper argues that such a market can best be considered as an auction; auction theory is then used to tune the structure of this 'bug auction' for efficiency and to better defend against attacks. The incentives for the software producer are also considered, and some fundamental problems with the concept are articulated.

Keywords: computer security, information security, auction theory, cost to break, security metrics, testing, economics

1 Introduction

The security of software is difficult to measure, leaving software producers no ability to make credible claims about it. As a result, they are not able to effectively charge a premium for software that is more secure and so have little incentive to invest in creating secure products.[1] The software market has thus been described as a market of lemons (in this case, insufficiently secure products), in which producers lack incentive to create high quality software [And01].

Although some software security metrics exist, they are inadequate. For example, estimates of software size or complexity could give an indication of the probable number of vulnerabilities (security defects) in a product, but software complexity is itself difficult to measure. The number of vulnerabilities patched

[1] Security, in this sense, is the lack of design or implementation defects that permit unexpected or undesired activity which threatens the confidentiality, availability, or integrity of an organization's information or resources.

Presented at the:
Workshop on Economics and Information Security. May 13-14, 2004: Minneapolis, MN, USA.

(fixed) during a given time period is quantifiable and readily observable, but this metric depends in part on the level of usage for the software and is also apparent only after the software has been in use for a period of time. Finally, one could measure the amount of effort put into security when designing and implementing the software, but such effort does not always translate to results.

A better metric would quantify software security such that buyers could effectively assess their risk. It might also allow producers to provide a quantitative assurance for their product. One such metric would be the 'cost to break' a system. A number of security firms have contests in which they offer prizes for 'breaking' their own product or the mathematical foundation upon which their product is built [RSA][Gol04].

Stuart Schechter proposes that firms create a vulnerability market in order to ascertain the cost to break of their system. A *producer* (the entity that has created and is selling the product to be tested) would offer rewards to the first *testers* (persons or organizations who identify vulnerabilities in return for payment) to inform it of new vulnerabilities in its product. If, after time, the reward is no longer being claimed, the producer can argue that the cost to break for its product must be greater than the size of the reward [Sch02a][Sch02b][Sch04].

In his exposition of this idea, Schechter characterizes it as a vulnerability market (VM). However, it can also be characterized as an auction. When considered this way, auction theory provides a number of useful tools for assessing its effectiveness. In section 2 I describe the current situation with respect to testing and vulnerability identification. Section 3 reviews Schechter's vulnerability market and notes some attacks that it anticipates and against which it defends. In section 4, I review the pertinent literature in auction theory and consider Schechter's vulnerability market as an auction. In section 5, this perspective is used to identify a number of improvements to the original design. Section 6 identifies two attacks by testers that are applicable to Schechter's vulnerability market and articulates alterations to the bug auction to defend against these attacks. Attacks by the producer are considered in section 7. In section 8, I note some fundamental problems with both the vulnerability market and the bug auction. Section 9 discusses some areas of future work and section 10 summarizes the results. Throughout the course of the paper, a vulnerability market as characterized by Schechter will be referred to as a *VM*. A *bug auction* will refer to a VM that has been modified as suggested in this paper.

2 Vulnerability Finding

Before critically examining the VM, it is useful to elaborate on the current situation with respect to the identification of vulnerabilities. A product's life can be divided into three distinct phases. In the *pre-release* phase, the product is still under development and testing; it has not yet been commercially released. *Post-release*, the product is commercially available and used by customers. *Post-depreciation*, the producer is no longer interested in actively improving the product or its security, usually because a successor product has

become available. The VM is a tool for finding vulnerabilities during the pre-release and post-release phase.

Currently, producers' pre-release testing occurs both in-house and using external volunteers (beta testers). During the post-release phase, producers learn about vulnerabilities in four general ways (in order of decreasing preference on the part of the producer): in-house testing, directly from 'white hat' security analysts, full-disclosure fora, or exploits 'in the wild.'

The first of these, testing, is the most attractive to producers. Unfortunately, for any reasonably complex software product, testing may find some vulnerabilities but is unlikely to find all of them [BAB99].

Second, white hat security analysts identify vulnerabilities after the product is in production and available to consumers. As the term is used here, a security analyst is an individual or organization that identifies vulnerabilities for non-malicious purposes—either for their own reputation gain or from a genuine pleasure derived from hacking—and reports these vulnerabilities to a responsible organization like CERT-CC or to the producer itself.

The third source of vulnerability reports is full-disclosure fora: mailing lists (e.g. Bugtraq), newsgroups, or any other forum in which individuals or organizations post vulnerabilities that may be observed by the general public. (Security analysts, as defined here, typically post to those forums but first wait for the producer to create a patch remediating the vulnerability.) The producer learns about the vulnerability at the same time as the general public and must then race to create a patch before the vulnerability can be exploited.

Fourth, the producer can learn about a vulnerability when it is exploited. For example, an exploit circulating in the black market might be detected when it is used against a system. This scenario is generally the worst for the producer, as it can result in prominent negative publicity.

3 The Vulnerability Market

Schechter envisages the VM as a tool both to improve a product's security and to quantitatively assess this security. Producers would ideally recoup the expenses they incur from the VM by the increased sales they achieve from having a more secure product. (Or, conversely, producers would recoup their expenses by retaining those customers they would otherwise lose due to the negative publicity and other effects of security failures.) He also notes that the VM can be extended to improve a product's quality, in addition to security.

Schechter argues that the producer is most interested in three properties: the cost of finding vulnerabilities ('value'), the speed with which they are found ('speed'), and the order in which they are found ('order'). The last of these goals is based on the assumption that bugs are not stochastically distributed; rather, the most common and obvious vulnerabilities should be found first, because these are the bugs most likely to become evident when the product is used [BAB99]. (See section 8.5 for a discussion of a different perspective recently put forward by Eric Rescorla.) By offering a monetary reward for vulnerabilities,

Schechter argues that producers will find them more rapidly (e.g. earlier in the production process, when they are less expensive to fix) and that the most obvious vulnerabilities will be identified first.

Schechter refines his initial proposal in a number of ways, by including: a continuously increasing reward, a reward spectrum, and a trusted third party.

Pre-release, the reward R is small when first offered; it then grows over time until it is claimed. After each new vulnerability is reported and verified, the reward amount is reset to R_0, the minimum reward value. If a vulnerability is reported more than once, only the first reporter receives the reward. The continuously increasing reward scheme maximizes value at the expense of speed: it trades potential delays in the reporting of vulnerabilities in return for monetary savings. A tester can choose to report a vulnerability at any time; waiting longer increases the reward she will receive, but it also increases the probability that another tester will report the same vulnerability and thus deprive her of the reward. (The post-release phase may differ and is described below.) If the bug is genuine and unique, the reward will be reset [Sch02a].

A producer may also create a spectrum of rewards, so it does not pay a large reward for an unimportant flaw. To reduce complexity, only one reward is published: the category of the vulnerability ascertains the fraction of the reward that the tester receives. During the pre-release phase, this refinement can also be used to improve the product's quality: the reward can also be offered for non-security related defects (bugs). For example, a report of an important security defect (e.g. a remote root vulnerability) might have a multiple of one and thus earn R, a less important security defect (e.g. a local privilege escalation vulnerability) might earn $\frac{4}{5}R$, while a minor quality bug (e.g. a user interface defect) might earn $\frac{1}{10}R$. Because defects can be related, the reward is reset after each report, regardless of the defect's classification [Sch02a].

Producers and testers interact through a trusted third party (TTP) who ensures that the producer pays the reward when appropriate and that the testers may remain anonymous. Involving the TTP prevents the producer from accepting vulnerability reports but not paying the testers because it falsely claims that the reports are non-unique. The (pseudo)anonymity it provides testers allows them to submit a report without fearing retaliation by either the producer or the black market. If a reward spectrum is used, the TTP will rank the submitted vulnerabilities according to severity and thus determine the proportion of the reward that the tester receives; using the TTP prevents the producer from minimizing the severity of a vulnerability in order to pay a smaller proportion of the reward. (Other attacks by the producer are considered in section 7.) However, the TTP will have less knowledge of the product than the producer; as a result, verifying reports will be more difficult for it. In order to ease the burden of verification, all vulnerability reports must include an example program that exploits the vulnerability. This requirement will result in rapid verification and minimize the expense of employing a TTP. Additionally, to reduce the frequency of 'frivolous' reports, testers may be charged the transaction cost of submitting a report; that transaction cost is the minimum R_0 [Sch02a].

The VM is thus first implemented pre-release, using the continuously increas-

ing reward; both the producer's employee testers and some number of external testers participate (section 8.4 explains how testers obtain the product). If this approach is taken, the producer needs to ensure that the existence of the market is incentive compatible with respect to those salaried testers directly employed by it. The goal of this phase is to identify and fix those vulnerabilities that are easy to find. The product is not commercially released until the reward for finding a vulnerability has gone unclaimed for some period of time.

Perhaps the best way to describe the VM is to use a (best case) hypothetical example:

> Software producer SecureCompany builds the closed-source SecureProduct. After SecureProduct is finished, the quality assurance division of SecureCompany tests the product and identifies a number of defects. These defects are fixed and SecureProduct is deemed ready for wider testing. So, SecureCompany engages TTP, a trusted third party, to oversee a vulnerability market. SecureCompany sets $R_0 = \$100$ and decides that R will increase at a rate of \$1/day. Quality defects are worth $\frac{1}{10}R$, while security defects are worth R. The TTP then distributes beta (pre-release) copies of SecureProduct to testers Alice and Bob.
>
> After ten hours of testing, Alice identifies a quality defect. She submits the defect to TTP. TTP verifies the defect and pays Alice $R = (\frac{1}{10})[100 + (1)(10)] = \$11$. TTP then resets the reward to $R_0 = \$100$. This process continues for two months, with a number of quality and security defects identified and fixed. At the end of two months, the reward has climbed to $R = 100 + (1)(24)(14) = \$436$ because no defects have been found in the past fourteen days.

The product is then made commercially available, and the post-release phase of the VM begins. R is reset to a new R_0, the value of the security assurance the producer is offering its customers (the reward is now offered only for security vulnerabilities).[2] Because the reward continues to grow and then be reset when vulnerabilities are reported, the security assurance is dynamic but provides a constant assurance at the level of R_0. Alternately, R can be set to a constant level (R does not increase) in order to provide a stable security assurance. The choice to use a continuously increasing reward or a stable reward can be made based on the producer's perception of the product's quality and the perceived reputation benefits of a continuously increasing reward. Regardless of whether or not the reward is increasing, its magnitude when unclaimed is the lower bound on the product's cost to break.[3] The product can safely be used to protect information whose total worth, summed across each customer and installation, is less than or equal to the magnitude of the unclaimed reward. Any rational, risk-neutral criminal who discovers a vulnerability would rather report it to the

[2]The security assurance is not a guarantee: the producer is not offering to recompense customers if they suffer losses due to a security breach.

[3]In actuality, the magnitude of the unclaimed reward only approximates the cost to break: it also includes transaction costs, risk, etc. However, for the purposes of the VM, this approximate cost is sufficient.

producer and receive the reward than risk the legal consequences of an attack that could at most reap the same monetary benefit.

Of course, for a widely used product, the cumulative value of information that relies on that product will almost certainly be greater than the assurance provided by the producer. However, a potential attacker faces the difficulty of converting access to that information into monetary gain: the reward may still be the most attractive source of renumeration for locating a vulnerability. Moreover, the real value of the reward is comparative: it highlights the security of the product relative to its competitors.

> SecureCompany decides that SecureProduct is now secure and commercially releases it. SecureCompany decides to also use a VM in the post-release phase. It again contracts with TTP. Now SecureCompany sets R_0 = \$250 and advertises that SecureProduct has a \$250 security assurance. R still increases at a rate of \$1/day, but the reward is now offered only for security vulnerabilities.
>
> SecureProduct is widely adopted. It is used in a variety of environments, and as a result four new vulnerabilities are identified and reported. Then, no vulnerabilities are identified for thirty-two days. SecureCompany sends press releases to industry journals to highlight the fact that the reward stands at \$1018 and remains unclaimed. Sales of SecureProduct increase dramatically.

As mentioned in section 1, the number of vulnerabilities found is one currently used metric of software security. As a result, a producer that employs a VM may be vulnerable to public relations attacks from a competitor who does not.

> Now, BigCompetitor enters the market. It advertises that its CompetingProduct is more secure than that of SecureCompany. BigCompetitor notes that four different vulnerabilities have been identified in SecureProduct. Although those vulnerabilities have been fixed, BigCompetitor argues that these vulnerabilities indicate the poor quality of programming in SecureProduct. In contrast, no vulnerabilities have been identified in CompetingProduct! Sales of SecureProduct plummet.

Schechter notes that any producer can bootstrap the use of the VM in order to differentiate itself from competitors. The producer offers at least R_0 for vulnerabilities in its product, thus establishing a lower bound on the cost to break its own product. It then employs a TTP to purchase, at a lower price, a single vulnerability for its competitor's product (establishing an upper bound on the cost to break that competitor's product). The producer has proven that its product is more secure than the competitor's: the lower bound on the cost to break of the producer's own product is higher than the upper bound on the cost to break of its competitor's product [Sch02b]. Although the value of the reward and the quantitative assurance may thus be low when compared to the total value of information that relies on the product, the use of the VM enables

the producer to quantitatively assert that its product is *relatively* secure. That is, the product is more secure than those offered by competitors.

> SecureCompany decides that it must respond to BigCompetitor. So, SecureCompany pays TTP to offer a $100 bounty for any vulnerabilities in CompetingProduct. A vulnerability is soon reported to TTP and the reward is paid. Because it is employed by SecureCompany, TTP does not report the vulnerability to BigCompetitor (or to SecureCompany).
>
> Now SecureCompany advertises that it offers a security assurance of $250 on SecureProduct, while vulnerabilities for CompetingProduct can be purchased for just $100! SecureProduct is thus more secure than CompetingProduct!
>
> BigCompetitor has no choice but to respond. It employs TTP to run a vulnerability market for CompetingProduct. It matches the $250 security assurance provided by SecureCompany. However, BigCompetitor decides that the reputation advantage of an increasing reward is not worth the cost. So, its $250 security assurance is constant, rather than continuously increasing.

4 The Vulnerability Market as an Auction

The auction literature in economics provides useful insight into the VM. Although a bug auction would be monopsonistic, I will follow convention in this section by using monopolistic auctions for my general examples. In a monopolistic auction, many buyers compete to purchase from just one seller, while in a monopsonistic auction many sellers compete to sell to just one buyer.

Auctions can largely be divided into two types: first price and second price. The most common example of a first-price auction is the Dutch, or open first-price descending, auction. In a Dutch auction the seller sets an exceptionally high price for the good being sold. That price then begins to descend automatically. When a the price reaches a point at which a buyer considers it reasonable, she signals the seller that she will purchase the good at that price; the first buyer to signal the seller thus obtains the good at the first price she is willing to pay for it.

The commonly used example of a second-price auction is the English, or open second-price ascending, auction. In an English auction, buyers compete with each other by offering increasingly higher bids for the good being sold. When only two buyers remain in competition, the bids will increase until one bid is equal to the lower of the two remaining bidders' valuations.[4] The winning buyer will then place a bid one unit higher than the losing buyer's valuation and thus obtains the good at the second price: the price at (or just above) the valuation of the item by the *second* highest bidder. The winning buyer's valuation may be much greater than the price he pays.

[4]That is, the price reaches the value that he places upon the item. Below this value, he would purchase the item and earn a profit. Above this value, if he purchases the item he suffers a loss. At this value, he is indifferent as to whether or not he obtains the item.

First-price and second-price auctions can also be held as closed auctions. In this instance, the bids would be sealed when submitted and then opened simultaneously by the seller. The winner would then pay either the price of her offer, or the price of the second-highest offer, for the respective types of auctions.

Although these four auctions are different in important ways, they are *revenue equivalent*: under standard conditions they provide on average the same revenue to the seller of the item, because rational bidders adjust their behavior according to the auction's structure [Vic61][RS81]. Those conditions are [MM87, 706]:

A. All bidders are risk neutral.

B. Each bidder's valuation of the good is private and independent of the good's valuation by other bidders.

C. Bidders are symmetric: they draw their values from the same probability distribution.

D. The seller's revenue comes entirely from the bids themselves; it receives no payments for the privilege of bidding and no revenue from the use of the good being sold.[5]

Schechter constructs a vulnerability *market*, perhaps because bootstrapping may require producers to offer rewards for finding vulnerabilities in *competitor's* products. However, he notes that when the producer is willing to reward anonymous testers, that producer rapidly becomes the only important buyer because of its unique ability to remediate the vulnerability with an update for the product [Sch02a]. Any vulnerability available on the black market will eventually be reported to the producer. Because the reward is offered to anybody, either the black market buyer or seller of a vulnerability would be tempted to engage in arbitrage and increase their profit by also selling the vulnerability to the producer. Indeed, since "the only way to ensure the testers won't resell is to take them out of the game, testers who wish to avoid sleeping with the fishes will be wary of selling security defects to anyone other than the [producer]" [Sch02a, 11]. The only vulnerabilities a producer will not learn about are those a tester identifies and then exploits himself (because he values the exploitation of the vulnerability more highly than the reward), or those vulnerabilities with black market value greater than the reward.

Thus, when a vulnerability is widely available on the black market or has been identified by an individual not interested in illegal gains, it will be sold to the producer at a price established by the continuously increasing reward function. When considered this way, the situation can be accurately described as an *auction*. This auction has one buyer, the producer (through its proxy, the

[5]This condition does not imply that bidders have no costs for participating in the auction. Bidders can pay both entry costs and bidding costs in a standard auction; the condition only requires that these payments are not received by the seller. For example, bidders typically pay some transaction cost to enter an auction, e.g. traveling to the auction site.

trusted third-party auctioneer) and a potentially unlimited number of sellers, the testers. If modeled after the VM, it is a reverse Dutch auction, or open first-price ascending auction, in which the price is set to be exceptionally low and then rises continuously until it is accepted by a seller.

Before a more complex analysis of this auction can be performed, it is first necessary to compare it with the standard auction described above. Requirement B is fulfilled: testers' valuations will in part depend upon the amount of work they put into identifying the bug (their costs) and will thus be both private and independent (unless testers collude, an occurrence that will be discussed in section 6.2). Requirement A is also fulfilled: no information extant suggests that testers will be risk averse or risk prone. Requirement D is met, as well: although the bidders must pay entry fees (elaborated upon below), those fees are not paid to the producer.

Requirement C, symmetry, is more interesting. If the auction is implemented according to the rules of the VM, bidders are asymmetric because testers from different countries will draw their valuations from different probability functions. McAfee and McMillan note that a typical example of asymmetry is a procurement auction in which both domestic and foreign bidders participate [MM87, 706]. In first-price auctions, asymmetry of values can result in the seller capturing less than all of the available value because the bidder with the highest valuation may not place the highest bid [MR85] [MM87].

Given this shortcoming, it is worth considering the value of using the current broad format of the VM (Dutch) as opposed to the other main formats (English, sealed first or second price). For this environment, the Dutch auction has one overriding structural advantage: a reward is always offered, ensuring that vulnerabilities are reported immediately if they are being traded on the black market. Sealed auctions must be periodic or the bidders can assume that they are the only bidder or one among few bidders (and thus bid more conservatively, causing the producer to overpay). English auctions require a waiting period after each bid to ensure that other bidders have the opportunity to counter. The delays necessary in these alternative models could cause a tester to first sell a vulnerability on the black market. The vulnerability could then be in use for the entire interval required by that auction format before the tester has the opportunity to sell it to the producer. Producers concerned about security may be risk averse; if so, then they also prefer first-price auctions to second-price auctions (although they would prefer a first-price closed auction over one that is open) [WHR98]. The Dutch auction has one other advantage: it conveys no information about the number of bidders (see section 6.2).

Because the reverse Dutch auction seems most suitable to the demands of the environment, the producer might attempt to remove the asymmetry through other means. Unfortunately, any means of remedying the asymmetric distribution of values would first require that testers with different probability distributions be distinguished; this requirement seems overwhelming in an environment in which anonymity is required and one in which any individual is a potential

participant.[6] The producer may thus have to accept the inefficiencies that result from asymmetric bidders.

We have so far considered only those auctions used to sell a single item. Sellers with more than one item to sell can employ sequential auctions or simultaneous multi-unit auctions. Although not found in the initial exposition of the concept, multi-unit auctions can still be revenue equivalent if the goods to be sold are homogeneous [MR89]. However, unlike single-unit auctions, designing multi-unit auctions to achieve efficient outcomes is difficult [Kle04a]. The bug auction is a sequential auction, in which the auctioneer will sequentially purchase bugs from the bidders.

The bidders in the bug auction must pay for the costs they incur in finding vulnerabilities (or attempting to find vulnerabilities), whether or not they actually win the auction. These expenses initially seem to resemble the all-pay auction, in which bidders pay an amount to the seller based on their bid, whether or not they win the auction: e.g. government procurement [KLSW02], political lobbying [Kru74] [BKdV93], and waiting lines [JS82].

However, this initial similarity is misleading. In all-pay auctions, the bidders must pay according to the magnitude of their bids. By dropping out early, a bidder can alter the amount he must pay. In the bug auction, the costs are incurred by each bidder whether or not he is even able to bid; a bidder can incur costs from attempting to enter the auction (searching for a bug) but fail in his attempt. Furthermore, a bidder's costs are not based on his bid, but on the effort required for him to identify the bug. A better mechanism for modeling these costs in a bug auction is to consider them to be entry costs.

The VM can be thus characterized as an auction: a sequential open first-price ascending auction with asymmetric independent private value bidders, high entry costs, and minimal bid costs.

5 Efficiency Enhancements

When the VM is considered as an auction, rather than a market, auction theory provides a number of useful tools for optimizing its structure. In addition, other enhancements become more clear when the VM is considered from this perspective. The alterations suggested below are intended to improve the efficiency of the auction; those alterations intended to cope with attacks are elaborated upon in section 6 and 7.

5.1 Endogenous Entry

Most of the standard auction models assume that the number of bidders is fixed and known. However, this assumption does not fit many real world auctions, in which potential bidders first decide whether or not to participate in an auction at all. For real world auctions, the importance of attracting a sufficient number

[6] Any user may by chance discover a vulnerability and then report it; in effect, some bidders may have zero costs for discovery.

of bidders cannot be overemphasized: the addition of a single bidder can benefit the auctioneer more than optimizations using reservation prices and entry fees [BK96].

Unfortunately, the bug auction's high entry cost is a significant disincentive to potential bidders.[7] A further disincentive exists for those testers who do not participate from the start of the sequence. Learning the product in order to test it is a non-trivial entry barrier, but it is one that must be incurred only once. Testers who incur this sunk cost thus face lower entry costs for the remaining auctions of the sequence, and they can then amortize the initial learning cost over the entire sequence.

This may be one area for which speed is more important to producers than value. Producers can benefit from being first to market and thus may prefer as short of a pre-release testing phase as is prudent. The post-release life of a product is also usually limited (e.g. by the arrival of future versions of the product). If speed is of import to the producer, then it can adjust its spending accordingly to induce high initial participation without entirely sacrificing the cost savings derived from the continuously increasing reward function:

Enhancement 1 *For the first auction or auctions in the sequence, set the initial value of the reward to a high level. For the following auctions, the producer should commit to offering a reasonably high minimum initial value for the reward.*

Alternatively, a producer might have the reward increase at a more rapid rate or with discontinuous jumps instead of offering a high starting reward value. Creating a significant initial incentive is of particular import in order to 'jump-start' the first few auctions in the series and draw a large pool of testers. Once these testers have invested in learning the product, they will have lower entry costs for participating in future auctions in the sequence. The producer expends its budget more rapidly in the beginning of the sequence in return for rapidly increasing the number of bidders. Increased participation should increase the rate with which vulnerabilities are identified and thus may well result in more vulnerabilities reported, at a lower cost, in the long run. Of course the lower costs that could result from this increased competition might decrease the incentive for bidders to continue searching for bugs; eventually a stable participation equilibrium would be reached . (The potential for producers to cheat is considered in section 7.)

Even without the inducement of a VM, the security analysts described in section 2 identify vulnerabilities and report them directly to producers, although they receive no direct financial reward. Instead, their reward seems to be an increase in their reputation, an incentive that can be used in combination with the monetary reward to induce entry.

[7] Menezes and Monteiro argue that entry costs simply discourage low-valuation bidders and are thus actually optimal from the seller's perspective [MM96]. However, in their model bidders pay a fixed entry cost at a time when they already know their valuation: in the bug auction, the entry cost is dependent upon each bidder and may influence their valuation.

Enhancement 2 *Combine the monetary reward with a reputation reward.*

When the producer releases a patch to remediate the vulnerability, it should also highlight the pseudonym of the tester that reported the vulnerability. The auctioneer, acting as TTP, could reveal the true identity of a pseudonym at that individual's request (for example to use that reputation to gain employment).

5.2 Variable Demand

Another important attribute of the bug auction is that it exhibits variable demand (or variable supply in the terminology of the standard monopolistic auction model). Each tester has a bug he has discovered. If he is not the first to claim the reward this auction, he can claim it during the next auction or the one after. However, at any time a second tester may identify the same bug and claim the reward before the first tester. If this situation occurs, the first tester (who has not yet claimed the reward) discovers that the demand for his bug has disappeared.

Neugebauer and Pezanis-Christou perform an extensive analysis and experimental examination of monopolistic sequential first-price auctions both with and without demand uncertainty. They find that demand uncertainty causes high-value bidders to bid less aggressively at first (i.e. they 'wait and see') and then with increasing aggressiveness; low-value bidders, on the other hand, initially bid aggressively in a seeming attempt to take advantage of early uncertainty. Casting their results in the terms of a monopsonistic action like the bug auction, they find that average prices are lower with uncertainty than without and that prices tend to decline over the series of auctions [NPC03]. (The latter trend may be countered by the generally increasing difficulty of finding each new bug as the most obvious or common bugs are remediated [BAB99].) In the bug auction, demand uncertainty thus seems to benefit the producer, resulting in lower costs.

However, from the perspective of the tester, two types of demand uncertainty exist in the bug auction: the anticipated number of auctions that will occur before another tester discovers the same bug and the total remaining number of auctions. The latter type of demand uncertainty can have a strongly negative impact on entry. As noted in section 5.1, each tester pays some one-time cost in order to familiarize herself with the product; this cost can to some degree be separated from the cost of finding a specific bug. The longer the sequence of auctions (chronologically), the more opportunity for the tester to amortize the cost of learning the product.[8]

However, the VM is designed in such a way as to make forecasting its cost relatively easy. Although a sequence of infinite length would undoubtedly result in the most secure product, all products have official lifetimes after which the producer is no longer interested in selling, supporting, or enhancing them. An

[8]The former type of demand uncertainty will also undoubtedly affect entry; however, it is necessary to provide the competition incentive that ensures the producer obtains value for its testing budget.

approximate budget can be created using this maximum duration $\bar{t}$, the rate of increase r, and an estimation of the total number of vulnerabilities v: the budget equals $r\bar{t} + vR_0$. The producer's knowledge and approximate budget can be used to eliminate the negative effect on entry of sequence-length demand uncertainty.

Enhancement 3 *The producer commits to the bug auction until a certain minimum amount of reward money has been claimed or until a minimum chronological period has elapsed, whichever occurs first.*

6 Attacks by Bidders

In his exposition of VMs, Schechter ignores or minimizes the impact of two important attacks: resale and collusion. This section considers each attack in turn and suggests alterations to the structure of the bug auction to minimize their potential impact. It is important to note, however, that neither of these attacks is fully defended against: their existence must be weighed against the benefits provided by the auction in order to ascertain its utility to any given producer.

6.1 Resale

Although it exists without the VM, the VM is likely to exacerbate the problem of resale: the potential for testers to sell vulnerability reports on the black market, perhaps to malicious hackers. The producer's willingness to pay the reward will ensure that it is eventually informed of vulnerabilities, but it does not ensure that no other entity is so informed. A smart tester sell the vulnerability after reporting it to the producer. He could truthfully advertise it as having been reported, but as-yet unremediated. Buyers could then pay a price commensurate with their utility for a short-lived vulnerability . The VM does not have any real solution to this dilemma. The potential result of a producer offering a reward for vulnerabilities found in a commercially available software product could thus be a temporary *decrease* in the security of that product. The reward could induce more testers to locate vulnerabilities which would then each be unremediated for some period of time and available on the black market. (This concern is related to recent work considered in section 8.5.) This problem cannot be entirely solved, and Schechter too readily dismisses it [Sch02a, 11]. However, an alteration to the bug auction could decrease the likelihood of resale.

Enhancement 4 *If a bug is found to be exploited before the producer can release a patch, the reward is reduced by some fraction.*

In order to reduce the likelihood of the tester reselling a vulnerability, the producer pays its reward conditionally: some fraction of the reward is withheld if attacks using this vulnerability are found 'in the wild' before the patch is released. This is an incentive for the tester to not resell the vulnerability, but

only if the resale value is less than the amount withheld from the reward. Some portion of the reward must always be paid to ensure that vulnerabilities available on the black market are reported to the producer.

6.2 Collusion

Klemperer argues that the two most important aspects of practical auction design are the encouragement of entry and the deterrence of collusion [Kle04b]. The former goal has already been addressed. This section considers the problems of both employee-tester collusion and tester-tester collusion.

Schechter largely ignores the problem of employee-tester collusion (e.g. an engineer involved with creating the product could team up with a tester not employed by the producer in order to create subtle vulnerabilities and then report them for mutual gain). This problem exists even without the market. A competitor could pay an engineer to insert a defect or back door into a product. However, the VM creates more opportunities (with lower transaction costs) for this attack. Schechter does suggest, in a different context, that the feedback from the VM be used to drive an incentive system for engineers; the more rapid feedback provided by the VM could enable rewards based on the quality of that engineer's work. A quality incentive system of this type might reduce the incentive for 'cheating' by colluding with external testers. Enforcing legal or contractual penalties for those engineers found to be collaborating with external testers is another means of reducing this risk. Sting operations, in which security officers employed by the producer pretend to be testers and attempt to entice engineers would also increase the risk to engineers of cheating. Nonetheless, it seems unlikely that this risk could be completely ameliorated, and at least to the extent that it is a risk without the VM, it remains a risk with it.

Even if employee-tester collusion is limited, the producer must still concern itself with collusion between testers (who could, for example, agree not to submit reports until the price reached some minimum). One important tool for preventing such collusion is to keep the number of testers (bidders) unknown [Kle04b]. No group of colluding testers can be sure that they will be the only testers, thus limiting their ability to control the auction. Furthermore, because the participants for any single auction in the series are unknown, no colluding group can identify and punish a defecting member through retaliatory bidding [CS02].

Enhancement 5 *The exact number of testers will not be public, although when the number is small an approximate count may be published to entice participation.*

The producer may choose to indicate the approximate number of bidders when that number is small, because broadcasting this fact can induce entry by other bidders.[9] Because the number will be approximate, it need not remove

[9]Menezes and Monteiro assert that if the bidders are risk neutral, knowledge of the number of bidders does not affect expected revenue from the auction [MM96]. See note 7 for a discussion of how their model differs from this one.

the collusion deterrence effect of having an unknown number of bidders.

However, in a VM the producer has no way of determining the number of potential testers: it does not even know how many active testers exist. In addition to its use in inducing entry, the producer could use knowledge of this number to tune the starting point and rate of reward change.

Enhancement 6 *Testers benefit by registering in advance with the trusted third party.*

This enhancement must be implemented carefully. Testers should not be *required* to register, because a user (who is not already a tester) who discovers a bug must be able to report it for the reward. Moreover, the benefit should be targeted only at testers: otherwise, if individuals not interested in testing can receive the benefit, then they will register and the count will be inaccurate. Although the benefit can vary with the situation, an example would be to reduce the transaction fee charged to testers that have been registered for at least three months. This benefit provides an incentive for individuals who are actively planning to test the product and claim the reward; on the other hand, individuals who discover a bug by chance still have an incentive to submit that bug (because R_0 is greater than the transaction fee). Testers register with the TTP, not the producer, so that they can maintain their (pseud)anonymity.

Even if the producer chooses not to attempt to register testers, enhancement 5 highlights the importance of maintaining (pseudo)anonymity and keeping secret any knowledge of participation numbers. The producer should keep the potential for collusion firmly in mind. Many incidental enhancements that might be considered to improve testers' efficiency could also lead to collusion: for example, supplying an official newsgroup for communication between testers.

7 Attacks by Producers

Schechter largely fails to consider that the producer may be motivated to 'cheat' in order to save money; it has five primary avenues through which to attack the bug auction: not paying for bug reports, releasing an exploit into the wild to reduce the payout to a bidder, abbreviating the length of the auction sequence, reseting a large reward, and falsely employing a large R_0 to jump-start the auction sequence. However, it is important to note that all of these attacks are against the testers; the producer has no viable means of altering the auction results in a way that cheats its *customers*.

The first attack is the simplest against which to defend: bidders refuse to participate unless the producer employs a TTP tasked with verifying bugs, ranking their severity, and paying the reward [Sch04].

The second opportunity to cheat is to decrease a claimed reward by planting an exploit (to take advantage of enhancement 4). However, the existence of exploits in the wild can result in significant damage to the product's reputation. This disincentive to cheating would likely dominate any potential savings, with the possible exception of a vulnerability that is almost trivial (at which point

the savings are also diminished, because the reward spectrum guarantees that the payout is also trivial).

The third attack, abbreviating the length of the auction sequence, is more problematic. Enhancement 3 proposes that producers commit to a minimum amount of reward money and a minimum chronological duration. The producer can then legitimately halt the auction once *either* of these minima has been surpassed. This attack has an obvious defence:

Enhancement 7 *The trusted third party should hold in escrow those funds that the producer has publicly committed to the auction.*

Of course, if no bugs are found in the product or the last reward is unclaimed at the end of the chronological limit, the producer can be refunded the remaining money. (Although bidders cannot require a producer to implement this enhancement, they can adjust their strategies if it is not implemented.)

The producer may attempt to cheat if it is no longer interested in spending those funds previously committed to the auction. It is prevented from arbitrarily halting the sequence because of the reputation damage it would sustain. The alternative is for the producer to employ an intermediary to report bugs of which it was already aware until the auction funds are exhausted. A number of disincentives reduce the risk of this attack. It requires that the producer introduce these bugs, either in patches or before the product's release. Introducing the bugs in patches creates a risk that testers may recognize that the producer is cheating (due to the extraordinarily high rate of bug introduction in patches) and cause the producer to suffer a reputation penalty. Introducing bugs before the product's release (or leaving unfixed some of the bugs identified by internal testers) would require sufficient foresight. Either of these scenarios creates a risk that the bugs will be identified and reported by an independent tester, thus causing the producer to pay a penalty (the reward). Even if the producer succeeds in ending the auction without arousing suspicion, it suffers a reputation penalty with respect to the number of bugs identified and reported in order to end the auction—unless the producer is confident that the full amount of money committed to rewards will be claimed before the chronological end of the sequence. Essentially, the risk of the producer cheating to end the auction early is significant only when the product is of poor quality, an attribute that can be observed by testers. Testers can thus factor this information into their strategies in order to compensate for the risk of a premature end to the auction.

The fourth attack is for the producer to report (via a proxy) previously planted bugs in order to reset the reward when it reaches too large of a value. In essence, the producer is implementing a reservation price (a cap on the reward value). Although Schechter designed the VM with a reservation price, the producer has a disincentive to employ an explicit reservation price because such caps discourage entry [LS94]. However, several disincentives seem likely to discourage the producer from cheating. Pre-release, if the reward has gone unclaimed for a long period, then the software must be reasonably secure and bug-free; the producer should simply end that phase of the auction in anticipation of releasing the product. The producer can then legitimately reset the

reward. If the producer cheats to reset the reward in the post-release phase, then it suffers a double blow to its reputation: it decreases the security assurance and also further suffers the reputation penalty of a bug having been found (although the latter disincentive could be somewhat decreased if the reported vulnerability is minor).

A further disincentive to reseting the reward lies in the tension between value and speed. Reseting the reward maximizes value at the expense of speed, because testers who have identified high-value bugs will wait for the reward to reach a high level again before reporting them. However, as noted in section 1, producers also value speed: they are motivated to rapidly release their products and to create the impression of security at as early of a stage in the product's life as possible. In addition, reseting the reward may not result in any real savings for the producer (unless the producer is attempting to prematurely end the auction). If testers are waiting for the reward to reach a certain level before they will report bugs, then they are unlikely to dramatically alter their reserve price because the producer has reset the reward. The producer could continuously reset the reward, but only at the cost of an increased perception of product insecurity.

The fifth opportunity to cheat arises if the producer employs enhancement 1. It could advertise a large R_0 in order to create a high level of initial participation and then cheat (by having a proxy report a previously planted bug) to avoid paying the cost of inducing such participation. However, it might thus fail to achieve the high level of participation that is its goal, and it further suffers the reputation penalty of a bug having been discovered. (Again, unless it is confident that the product is of such poor quality that all of the funds committed to the auction will be claimed before the end of the sequence; however, if it already possesses this knowledge, then it should simply decrease the funds committed to the auction or not employ a bug auction at all.)

If the producer employs 7, a final disincentive for the latter three of these methods for the producer to attack the auction is legal and financial. If it uses a proxy to cheat and claim back some of the funds already given to the TTP, then general accounting practices and taxation laws may make it difficult for the producer to explain the reclaimed funds. Furthermore, the proxy still has to identify himself to the TTP, which may identify a link between him and the producer.

In general, producers that are concerned about their reputation have incentives that discourage cheating. If the producer is no longer concerned with its reputation, there exists no real defense against its cheating. However, testers would likely have other indications of this disregard and factor it into their strategies.

8 Fundamental Problems

The previously mentioned attacks can be defended against both through the structural suggestions made above and through legal, cultural, and managerial

tools employed during the course of the auction sequence. However, Schechter's market suffers from five fundamental shortcomings: its expense, the potentially harmful effect on the producer's reputation, the loss of free testing, the potential for copyright infringement, and the possibility that vulnerability finding does not lead to security.

8.1 Expense

Any producer can employ the VM to enhance its pre-release testing: it may find that the VM, in combination with limited in-house 'white-box' testing, is more cost effective than relying entirely on in-house testers. Deriving the competitive benefit of the security assurance is more difficult, although Schechter does provide for bootstrapping the market to force your competitor to also use a VM (see section 3).

However, in order to effectively advertise a quantitative level of security (e.g. a constant stable reward at $20,000), the producer must always be willing to pay at least that amount for vulnerabilities reported after the product is made commercially available; if a rash of unique vulnerabilities are reported, this guarantee could be rather costly to the producer.

Schechter implies that few vulnerabilities will be reported after the product is made commercially available, because he assumes that the VM has been used extensively during the pre-release testing of the product. As a result, Schechter assumes that, post-release, the reward will not be claimed with any great frequency.

One problem with these assumptions is the value most producers see in being first to market. While Microsoft claims to be delaying product releases to ensure a high enough level of security [Bek02], they are also dominant in these markets. Producers that lack the luxury of dominating their markets may be unwilling to employ the VM in testing for an open-ended period (i.e. until testers stop claiming the reward).

Another problem lies in the nature of testing. As testing continues, the effort and time required to find new bugs increases, *provided* that the environment is constant. After a complex product is commercially released, the producer often receives a flurry of new bug reports because the commercial users employ the product in a wide variety of new environments. The producer's testers were unable to anticipate or replicate all of these environments and so did not find these bugs [BAB99]. Again, this factor implies that the producer may find the reward it offers after the product is commercially released to be more costly than expected.

8.2 Reputation

The problems described in section 8.1 may impact more than just the amount of money the producer spends on rewards. If a number of bugs are reported in the period immediately after the product is released, the patches released to fix those bugs may create the appearance of insecurity. (Indeed, as mentioned in

section 1, the number and rate of patches released is one currently used metric of software security.)

Schechter notes that, for *testers* considering whether or not to participate in the VM, the *perception* of security can be more important than the reality [Sch04, 47]. However, he does not fully address the impact of perception on the *customers* of the producer. Although a product may be more secure if the VM is used during testing and after release, customers might still base their perception of security on the number of patches disseminated. Reputation may thus be as important as the three properties already mentioned: value, speed, and order.

One area of future research would be to identify an auction structure that can better account for reputation while still minimizing the costs to the producer. The producer could perhaps 'cluster' its fixes, waiting until is has corrected more than one vulnerability before issuing a patch. However, it would need to be conscious that delays in patching any single vulnerability might enable that vulnerability to be sold on the black market and exploited.

8.3 Loss of Free Testing

The existence of Schechter's market also alters the interaction that occurs between the producer, security analysts, full-disclosure fora, and the black market that was described in section 3. One negative effect is that the producer will have to pay for the vulnerability reports that it previously received for free from security analysts. One the other hand, the VM will potentially draw more analysts and thus the product may more rapidly approach a secure state, in which most vulnerabilities, and particularly the most obvious ones, have been identified and remediated (assuming such a state exists—see section 8.5). The existence of resale means that the full-disclosure model of identification will not be significantly impacted, although again the producer will be paying for reports that it previously received for free.

8.4 Copyright Infringement

Schechter specifies that all testers must have "full and complete" access to the product [Sch04, 66]. This stipulation is necessary to ensure that the product is tested by the maximum possible number of testers and that their costs are minimized. For open-source products, full and complete access is not problematic. For closed-source products, full and complete access will realistically be interpreted as access to the product in its executable form, not its source code. Even so, this requirement may be problematic for closed-source products. If implemented poorly, it could be used as a means of copyright infringement: individuals who have no intention of testing could acquire a free copy of the product for personal use. Schechter notes the possibility of this infringement occurring; to prevent it, he suggests that the testing copies could have some functionality removed or disabled so that they would be less useful to a would-be infringer. He argues that illegal copies of most software are widely available;

as a result, those copies put into distribution for testing would not greatly increase the availability of free copies for those unwilling to pay. The effectiveness and importance of this attack (and the countermeasure of disabling functionality) will differ according to the product. If a product is not already widely available from illegal sources, the producer may choose to disable functionality. If that solution is also not practical, the producer may choose not to employ the VM with that product.

8.5 Vulnerability Finding and Security

Eric Rescorla challenges the assumption that work invested in finding vulnerabilities in software is socially useful: he argues that such efforts fail to decrease the total cost of security breaches [Res04]. If this challenge is correct, then using a bug auction in the post-release phase does not result in a more secure product (although it still provides a quantitative security assurance and means of comparing products). The analysis that results in this conclusion is based upon the assumption that vulnerabilities are stochastically distributed in a product.

This work is predicated on the opposing assumption: that some vulnerabilities are easier to find than others, a viewpoint for which the literature provides support [BAB99]. If vulnerabilities are ordered this way, then each tester with an as-yet unreported vulnerability must assume that another tester may find the same vulnerability. Even if vulnerabilities are stochastically distributed, shared vulnerability finding heuristics may create this ordering effect.

More investigation into the distribution of vulnerabilities is clearly warranted. If further investigation confirms Rescorla's assumption, then the bug auction would still retain its pre-release utility and its value for qualifying the relative security of competing products. However, if vulnerabilities are not ordered, then any tester with a vulnerability need not fear that the vulnerability will be found by another tester. As a result, the competition incentive is removed and an increasing reward may not be useful.

9 Future Work

Further work in this area includes formally modeling the bug auction. The bug auction would be modeled as a sequential open first-price ascending auction with an unknown number of asymmetric independent private-value bidders, variable demand, high entry costs, and minimal bid costs. A formal model could also compare the value to the producer of an asymmetric open first-price ascending auction with an asymmetric closed second-price auction. This paper has assumed that the drawbacks of a closed second-price auction are more significant than those of an open first-price auction, but formal verification would be useful.

Identifying a means of compensating for bidder asymmetry would increase the value of the bug auction to producers.

Also, the costs and efficiency of current quality assurance methods might provide an interesting comparison with the bug auction. How much do compa-

nies currently spend for their quality assurance testers to identify a single bug? Do those testers typically identify bugs of only a certain magnitude (e.g. are user interface bugs found and remote security vulnerabilities not found)?

10 Conclusion

Schechter's VM is an innovative solution to a vexing problem in software security; although his proposal is daring, the cost-to-break challenges currently offered by firms demonstrate that it is not inconceivable for software firms to employ this idea. However, the expense of implementing the VM is not trivial: a robust theoretical understanding of the concept will encourage those firms who might consider implementing it.

This paper has identified areas where auction theory can improve the efficiency of the VM. It has highlighted attacks that were not considered or were too readily dismissed by Schechter. Finally, it has noted four fundamental weaknesses of the VM that can not readily be corrected. I argue that the VM is an important concept, but one that can be better considered as a bug auction. Considering the VM as an auction provides a more firm theoretical foundation for understanding the idea. Further, it enables the producer to fine-tune the auction to increase its effectiveness.

A bug auction would provide producers with: the assurance that it will learn about those vulnerabilities that are available on the black market, the ability to significantly increase scrutiny of a product (particularly early in the development life cycle when it is otherwise difficult to achieve this goal), and the ability to provide a qualitative assurance of security to their customers.

The arguments in favor of using a bug auction in the pre-release phase seem clear: the producer benefits from both more testing and, more importantly, from testing in a wide variety of environments. This benefit is gained at the expense of a potentially more costly testing program and also a potential delay in releasing the product. While neither of these drawbacks are trivial, the current market emphasis on security seems likely to make them bearable for some producers. Assessing the use of the bug auction in the post-release phase of a product's life is more difficult, but it has the potential to revolutionize the market for secure products. Although the implementation of a bug auction would require significant changes to the management culture of producers and consumers, it nonetheless seems uniquely suited to providing a relatively strong security assurance and an efficient means of improving product quality and security.

Acknowlegements: Stuart Schechter was gracious enough to provide a draft of his unfinished dissertation. This paper would not be possible if he had not already done the most difficult work: conceiving of the vulnerability market. I also thank Alessandro Acquisti, Ross Anderson, Daniel Immerwahr, and the anonymous referees for their helpful comments and advice. This work was supported by grants from the Marshall Aid Commeration Commission.

References

[And01] Ross Anderson. Why information security is hard - an economic perspective. In *17th Annual Computer Security Applications Conference*, December 2001. New Orleans, LA, USA.

[BAB99] Robert M. Brady, Ross J. Anderson, and Robin C. Ball. Murphy's law, the fitness of evolving species, and the limits of software reliability. Technical Report 471, University of Cambridge Computer Laboratory, September 1999.

[Bek02] Scott Bekker. Windows .net server delays complicate longhorn schedule., April 2002. `http://www.entmag.com/news/article.asp?EditorialsID=5316`.

[BK96] Jeremy Bulow and Paul Klemperer. Auctions versus negotiations. *The American Economic Review*, 86(1):180–194, 1996.

[BKdV93] Michael R. Baye, Dan Kovenock, and Casper G. de Vries. Rigging the lobbying process: An application of the all-pay auction. *The American Economic Review*, 83(1):289–294, 1993.

[CS02] Peter Cramton and Jesse A. Schwartz. Collusive bidding in the FCC spectrum auctions. *Contributions to Economic Analysis and Policy*, 1(1), 2002.

[Gol04] M. Corey Goldman. Code that can't be cracked. *The Star*, January 2004. `http://www.thestar.com/NASApp/cs/ContentServer?pagename=thestar/Layout/%Article_PrintFriendly&c=Article&cid=1074467408185&call_pageid=968350072197`.

[JS82] Charles A. Holt Jr. and Roger Sherman. Waiting-line auctions. *The Journal of Political Economy*, 90(2):280–294, 1982.

[Kle04a] Paul Klemperer. A survey of auction theory. In Paul Klemperer, editor, *Auctions: Theory and Practice*, chapter 1A. Princeton University Press, 2004.

[Kle04b] Paul Klemperer. What really matters in auction design. In Paul Klemperer, editor, *Auctions: Theory and Practice*, chapter 3A. Princeton University Press, 2004.

[KLSW02] Todd Kaplan, Israel Luski, Aner Sela, and David Wettstein. All-pay auctions with variable rewards. *The Journal of Industrial Economics*, L(4):417–430, 2002.

[Kru74] Anne O. Krueger. The political economy of the rent-seeking society. *The American Economic Review*, 64(3):291–303, 1974.

[LS94] Dan Levin and James L. Smith. Equilibrium in auctions with entry. *The American Economic Review*, 84(3):585–599, 1994.

[MM87] R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.

[MM96] Flavio M. Menezes and Paulo K. Monteiro. A note on auctions with endogenous participation. Technical Report 9610003, Washington University in St. Louis, September 1996. Economics Working Paper Archive, Microeconomics.

[MR85] Eric S. Maskin and John G. Riley. Auction theory with private values. *The American Economic Review*, 75(2):150–155, 1985.

[MR89] Eric S. Maskin and John G. Riley. Optimal multi-unit auctions. In Frank Hahn, editor, *The Economics of Missing Markets, Information, and Games*, pages 312–335. Oxford University Press, 1989.

[NPC03] Tibor Neugebauer and Paul Pezanis-Christou. Bidding at sequential first-price auctions with(out) supply uncertainty: A laboratory analysis. Technical Report 558.03, Unitat de Fonaments de l'Anlisi Econmica (UAB) and Institut d'Anlisi Econmica (CSIC), February 2003.

[Res04] Eric Rescorla. Is finding security holes a good idea? In *Workshop on Economics and Information Security*, May 2004. Minneapolis, Minnesota.

[RS81] John G. Riley and William F. Samuelson. Optimal auctions. *The American Economic Review*, 71(3):381–392, 1981.

[RSA] `http://www.rsasecurity.com/rsalabs/challenges/`.

[Sch02a] Stuart Schechter. How to buy better testing: Using competition to get the most security and robustness for your dollar. In *Infrastructure Security Conference*, October 2002. Bristol, UK.

[Sch02b] Stuart Schechter. Quantitatively differentiating system security. In *Workshop on Economics and Information Security*, May 2002. Berkeley, CA, USA.

[Sch04] Stuart Schechter. *Computer Security: A Quantitative Approach*. PhD thesis, Harvard University, (work in progress) January 13 2004.

[Vic61] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[WHR98] Keith Waehrer, Ronald M Harstad, and Michael H Rothkopf. Auction form preferences of risk-averse bid takers. *The RAND Journal of Economics*, 29(1):179–192, 1998.