Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms

Aron Laszka[1(✉)], Mingyi Zhao[2], and Jens Grossklags[2]

[1] University of California, Berkeley, USA
laszka@berkeley.edu
[2] Pennsylvania State University, University Park, USA

Abstract. Bug-bounty programs have the potential to harvest the efforts and diverse knowledge of thousands of white hat hackers. As a consequence, they are becoming increasingly popular as a key part of the security culture of organizations. However, bug-bounty programs can be riddled with myriads of invalid vulnerability-report submissions, which are partially the result of misaligned incentives between white hats and organizations. To further improve the effectiveness of bug-bounty programs, we introduce a theoretical model for evaluating approaches for reducing the number of invalid reports. We develop an economic framework and investigate the strengths and weaknesses of existing canonical approaches for effectively incentivizing higher validation efforts by white hats. Finally, we introduce a novel approach, which may improve efficiency by enabling different white hats to exert validation effort at their individually optimal levels.

Keywords: Bug-bounty programs · Vulnerability discovery · Economics of security · White hat hackers · Misaligned incentives · Crowdsourcing

1 Introduction

In recent years, many organizations have launched independent bug-bounty programs (e.g., Google and Facebook) or have joined bug-bounty platforms, such as HackerOne, Cobalt or Bugcrowd, that facilitate programs for them. These programs allow independent security researchers, so-called white hats, to evaluate the security of a website or software within a set of predefined rules. White hat hackers are encouraged to submit reports for potential vulnerabilities, which after validation by the organization will be rewarded, for example, with monetary bounties. The benefits of these programs are at least twofold. First, the organizations' products are examined by the large and diverse population of white hat hackers, which would be prohibitively expensive to employ directly. White hats' efforts effectively complement the usage of automated web vulnerability scanners, which have been shown to have only limited coverage [8,26]. Second, the public nature of the majority of these programs can signal to third parties the commitment of organizations towards continual security improvements.

I. Askoxylakis et al. (Eds.): ESORICS 2016, Part II, LNCS 9879, pp. 161–178, 2016.
DOI: 10.1007/978-3-319-45741-3_9

The scale of bug-bounty programs is sizable and growing. For example, on HackerOne, more than 20000 security vulnerabilities have been reported and fixed for hundreds of organizations as of May 2016. These contributions are based on reports from over 2500 different white hat hackers, who received bounties totaling over $7.3 M.

However, the public nature of the majority of the programs also poses a challenge since virtually anyone can participate, and organizations may be overwhelmed by low-value reports [28]. In fact, bug-bounty platforms acknowledge that the key challenge "companies face in running a public program at scale is managing noise, or the proportion of low-value reports they receive" [13]. These low-value reports include spam (i.e., completely irrelevant reports), false positives (i.e., issues that do not actually exist or have no security impact), and out-of-scope reports. For the purpose of our work, we will refer to all of these issues as invalid reports.

Invalid reports may be the result of imprecise research approaches or lack of thorough validation by white hats. For example, some hackers utilize automated vulnerability scanners in the discovery process, which typically have a high false-positive rate [8]. Since filtering out false positives is costly, some hackers may prefer to send the outputs of an automated scanner to the bug-bounty program. Further, some discoveries may initially appear to be valid, while they are actually not. For example, a hacker needs to read the participation rules for a program and validate whether an identified issue is out-of-scope. Another important facet is that the hacker needs to demonstrate that a discovered flaw can really lead to a security problem. Finally, writing a good report for a valid discovery requires effort, and can also be seen as a part of the validation process.

In practice, the number of invalid reports is significant. For example, for Bugcrowd's public programs, 34.5 % of all submissions were marked invalid (from January 2013 to June 2015) [7]. HackerOne reported that 54 % of all submissions were marked as invalid before the platform started to proactively improve the signal-to-noise ratio (in 2015) [13].

As a direct response, bug-bounty platforms have started to offer multiple policies that participating organizations can use for reducing the number of invalid reports. For example, HackerOne has introduced "Signal Requirements" and "Rate Limiter" mechanisms, which organizations can use to increase the quality of reports [13]. The former allows only those hackers to submit reports who maintain a given ratio of valid to invalid submissions, while the latter limits the number of reports that a hacker can make in some time interval. These policies aim to incentivize hackers to engage in consistent efforts to validate their reports. According to HackerOne [13], these measures together have decreased the percentage of invalid reports to around 25 %.

Unfortunately, policies may also prevent some hackers, who could contribute valid reports, from participating and may force others to waste effort by being overly meticulous. Consequently, strict policies will result not only in a reduced number of invalid reports, but also in a lower number of valid reports. In summary, finding the right policies and their optimal configuration is a challenging

problem since white hat hackers need to be incentivized to produce and submit valid reports, but at the same time, discouraged from submitting invalid reports.

With our work, we provide the first theoretical framework for modeling these policies, finding their optimal configuration, and comparing them with each other. In addition to modeling existing policies, we also propose a new policy, which directly rewards hackers for their accuracy. For each policy, we provide theoretical results on how hackers react to the implementation of the policy, and then complement our analytic results with numerical analyses comparing the policies.

The remainder of this paper is organized as follows. In Sect. 2, we discuss related work on bug-bounty programs, and vulnerability discovery. In Sect. 3, we introduce our economic model of bug-bounty programs. In Sect. 4, we study a set of canonical policies for decreasing the number of invalid reports. In Sect. 5, we present numerical results on these policies. Finally, in Sect. 6, we offer concluding remarks and outline future work.

2 Related Work

2.1 Bug Bounty and Vulnerability Markets

There has been a long-standing interest for using market approaches to address software security problems. Böhme established a terminology for organizational principles of vulnerability markets by comparing bug bounties, vulnerability brokers, exploit derivatives and cyber-insurance [5]. Among these market approaches, bug bounties have received strong attention from both industry and academia. Schechter proposed a testing competition in which multiple testers report security defects to a software company for reward [24]. Ozment further extended Schechter's testing competition into a vulnerability auction to improve its efficiency and better defend against attacks [19]. In both mechanisms, the amount of reward grows linearly with time, and resets to the initial value every time a report is accepted. This reward policy enables the firm to minimize the cost while still offering a fair price for the vulnerabilities discovered by the testers. The reward level at a given time can also serve as a measurement of software security. However, these two mechanisms did not fully consider the problem of invalid reports, which cause high cost for today's bug-bounty programs and the participating organizations. Schechter proposed to require testers to pay the transaction costs of processing reports [24]. However, this idea would prevent many hackers from submitting reports and thus is not implemented in reality. Our research focuses on real bug-bounty programs and their policies, thus complements these early proposed mechanisms.

In recent years, researchers have conducted multiple empirical analyses on bug-bounty programs. Finifter et al. empirically studied the Google Chrome vulnerability reward program (VRP) and the Mozilla Firefox VRP [11], and suggested that VRPs are more cost-effective compared to hiring full-time security researchers in terms of finding security flaws. Zhao et al. conducted a comprehensive study of two bug bounty ecosystems, Wooyun and HackerOne, to

understand their characteristics, trajectories and impact [28]. They quantitatively discussed the low signal-to-noise ratio problem which is the focus of this paper. Maillart et al. empirically studied reward distribution and hacker enrollments of public bounty programs on HackerOne and found that growing rewards cannot match the increasing difficulty of vulnerability discovery, and thus hackers tend to switch to newly launched programs to find bugs more easily [18]. Similar to [28], they suggested that a bounty program manager should try to enroll as many hackers as possible to deplete the number of vulnerabilities more effectively. However, this leads to a significant increase of invalid submissions, which we aim to address in this paper.

For other types of market-based vulnerability discovery mechanisms, Kannan and Telang theoretically demonstrated that unregulated vulnerability markets almost always perform worse than regulated ones, or even non-market approaches [15]. They further found that offering rewards for benign vulnerability discoverers is socially beneficial. Frei et al. studied a security ecosystem including discovers, vulnerability markets, criminals, vendors, security information providers and the public, based on 27,000 publicly disclosed vulnerabilities to examine the risk and impact of such an ecosystem [12]. They found that between 10 % and 15 % of the vulnerabilities of major software vendors are handled by commercial vulnerability markets, and exploits become available faster than patches on average. Ransbotham et al. empirically examined the effectiveness of vulnerability markets and concluded that market-based disclosure restricts the diffusion of vulnerability exploits, reduces the risk of exploitation, and decreases the volume of exploitation attempts [22]. Algarni and Malaiya analyzed data of several existing vulnerability markets and showed that the black market offers much higher prices for zero-day vulnerabilities, and government agencies make up a significant portion of the buyers [1]. Bacon et al. have proposed a more general market design that contains bug hunters, developers, and users [4]. Bug bounty, and vulnerability markets in general have also caused debates regarding their ethics. A recent panel discussion of such issues and their implications can be found in [10]. Finally, Libicki et al. conducted a comprehensive study of vulnerability markets and their relevance to cyber security and public policy [17].

2.2 Empirical Analysis of Software Vulnerability Discovery

Previous work has studied various software vulnerability datasets to understand vulnerability discovery. Rescorla studied the ICAT dataset of 1,675 vulnerabilities and found very weak or no evidence of vulnerability depletion. He thus suggested that the vulnerability discovery efforts might not provide much social benefit [23]. This conclusion is being challenged by Ozment and Schechter, who showed that the pool of vulnerabilities in the foundational code of OpenBSD is being depleted with strong statistical evidence [20,21]. Ozment also found that vulnerability rediscovery is common in the OpenBSD vulnerability discovery history [20]. Therefore, they gave the opposite conclusion, i.e., vulnerability hunting by white hats is socially beneficial. More recently, Shahzad et al. [25] conducted

a large-scale study of the evolution of the vulnerability life cycle using a combined dataset of NVD, OSVDB and FVDB. Their study provided three positive signs for increasing software security: (1) monthly vulnerability disclosures are decreasing since 2008, (2) exploitation difficulty of the identified vulnerabilities is increasing, and (3) software companies have become more agile in responding to discovered vulnerabilities.

More recently, researchers started to pay attention to the behaviors of vulnerability discoverers. One finding is that vulnerability discoverers are rather heterogeneous. Edmundson et al. conducted a code review experiment for a small web application with 30 subjects [9]. One of their findings is that none of the participants was able to find all 7 Web vulnerabilities embedded in the test code, but a random sample of half of the participants could cover all vulnerabilities with a probability of about 95 %, indicating that a sufficiently large group of white hats is required for finding vulnerabilities effectively. Zhao et al. conducted an initial exploratory study of white hats on Wooyun [27] and uncovered the diversity of white hat behaviors regarding productivity, vulnerability type specialization, and discovery transitions. Huang et al. uncovered that hackers at various levels of experience exist in the vulnerability disclosure ecosystem [14]. They found that hackers with different levels of accuracy have diverse strategies in selecting to which programs to contribute [14]. In this paper, we account for these studies by evaluating the effectiveness of bug bounty policies for both homogeneous and heterogeneous white hat hackers.

3 Model

In this section, we introduce our economic model of bug-bounty programs. Note that we will focus on features that are relevant to invalid reports and policies for limiting them. A list of symbols used in this paper can be found in Table 1.

Notation. We use uppercase letters to denote constants (e.g., V) and functions (e.g., $D_i(t_i)$), lowercase letters to denote variables (e.g., b), and bold font to denote vectors (e.g., $\boldsymbol{t}$). We use Lagrange's notation (i.e., the prime notation) for derivatives of single variable functions (i.e., $D_i'(t_i)$ denotes the first derivative of $D_i(t_i)$). For multivariable functions, we use Leibniz's notation (e.g., $\frac{d}{db}U_O(b, \boldsymbol{t}, \boldsymbol{v})$ denotes the first derivative of $U_O(b, \boldsymbol{t}, \boldsymbol{v})$ with respect to b). Finally, we use $^{-1}$ to denote the inverse of a function (e.g., $D_i^{-1}(t_i)$ is the inverse of function $D_i(t_i)$).

In our model, we consider an *organization* that runs a bug-bounty program and *hackers* that may participate in the program. We group hackers who have the same productivity and preferences together into *hacker types*. Since hackers of the same type will respond in the same way to the policies set by the organization, we study their choices as a group instead of as individuals.

The number of *potential vulnerabilities discovered* by hackers of type i is

$$D_i(t_i), \tag{1}$$

where t_i is the amount of time hackers of type i spend on discovery. We assume that $D_i(0) \equiv 0$ and that D_i is a non-negative, increasing, and strictly concave function of t_i. That is, we assume that the marginal productivity of discovery is decreasing, which is supported by experimental results and existing models (e.g., [2,6,29]).

On average, $\Phi_i \cdot D_i(t_i)$ of these discoveries are actual vulnerabilities and $(1 - \Phi_i)D_i(t_i)$ of them are *invalid* ($0 < \Phi_i < 1$). The number of potential vulnerabilities that are *validated* (i.e., confirmed to be valid or to be invalid) by hackers of type i is

$$I_i(v_i), \tag{2}$$

where v_i is the amount of time hackers of type i spend on validating their discoveries. We assume that $I_i(0) \equiv 0$ and that I_i is a non-negative, increasing, unbounded, and strictly concave function of v_i. The rationale behind the concavity assumption is that some discoveries are easier to validate, and a rational, utility-maximizing hacker starts validation with the easier ones. Finally, we obviously have that

$$v_i \leq I_i^{-1}\left(D_i(t_i)\right). \tag{3}$$

That is, a hacker will not waste time on validation once he has finished with all of his discoveries.

After validating his $I_i(v_i)$ discoveries, the hacker will report all $\Phi_i \cdot I_i(v_i)$ discoveries that he has confirmed to be valid vulnerabilities. Further, he will also report all $D_i(t_i) - I_i(v_i)$ unvalidated discoveries, of which $\Phi_i \cdot (D_i(t_i) - I_i(v_i))$ are valid and $(1 - \Phi_i)\left(D_i(t_i) - I_i(v_i)\right)$ are invalid. Hence, the number of valid reports made by hackers of type i is

$$\Phi_i \cdot D_i(t_i), \tag{4}$$

Table 1. List of symbols

Symbol	Description
	Constants and Functions
V	average value of a valid report for the organization
C	average cost of processing a report for the organization
W_i	value of time for hackers of type i
Φ_i	fraction of discoveries by hackers of type i that are valid vulnerabilities
$D_i(t_i)$	number of potential vulnerabilities discovered by hackers of type i
$I_i(v_i)$	number of discoveries validated by hackers of type i
	Variables
b	average bounty paid for a valid report
t_i	time spent on vulnerability discovery by hackers of type i
v_i	time spent on validating discoveries by hackers of type i
α	accuracy threshold imposed on participating hackers
ρ	report-rate threshold imposed on participating hackers
δ	validation reward for participating hackers

while the number of invalid reports is

$$(1 - \Phi_i)\left(D_i(t_i) - I_i(v_i)\right). \tag{5}$$

The utility of hackers of type i is

$$\mathcal{U}_{H_i}(b, t_i, v_i) = b \cdot \Phi_i \cdot D_i(t_i) - W_i \cdot (t_i + v_i), \tag{6}$$

where b is the average bounty that the organization pays for a valid report, and $W_i > 0$ is the hacker's utility for spending time on other activities. In other words, W_i is the opportunity cost of the hacker's time.

The organization's utility is

$$\mathcal{U}_O(b, \boldsymbol{t}, \boldsymbol{v}) = \sum_i \underbrace{(V - b)\Phi_i D_i(t_i)}_{\text{net value of valid reports}} - \underbrace{C \cdot \Big(\underbrace{\Phi_i D_i(t_i)}_{\text{valid reports}} + \underbrace{(1 - \Phi_i)\left(D_i(t_i) - I_i(v_i)\right)}_{\text{invalid reports}}\Big)}_{\text{cost of processing reports}}, \tag{7}$$

where $V > 0$ is the average value of a valid report for the organization, and $C > 0$ is the average cost of processing a report. Note that V can incorporate a variety of factors, such as a difference between the processing costs of valid and invalid reports, cost of patching a vulnerability, etc. By letting $\hat{V} = V - C$, we can express the organization's utility as

$$\mathcal{U}_O(b, \boldsymbol{t}, \boldsymbol{v}) = \sum_i (\hat{V} - b)\Phi_i D_i(t_i) - C \cdot (1 - \Phi_i)\left(D_i(t_i) - I_i(v_i)\right). \tag{8}$$

4 Analysis

In this section, we provide theoretical results on our bug-bounty model, and study how hackers respond to various policies. First, as a baseline case, we study the model without any policy against invalid reports. Then, we study two policies, *accuracy threshold* and *report-rate threshold*, which model existing practical approaches for limiting invalid reports. Finally, we propose a novel policy, *validation reward*, which incentivizes hackers to validate their discoveries instead of imposing strict limits on their actions.

4.1 Without an Invalid-Report Policy

First, we consider a baseline case, in which the organization does not have a policy for limiting invalid reports. In this case, the organization's choice is restricted to adjusting the bounty paid for valid reports. The following proposition characterizes the hackers' response to the bounty value chosen by the organization.

Proposition 1. *Without an invalid-report policy, hackers of type i will spend*

$$t_i^*(b) = \begin{cases} (D_i')^{-1}\left(\frac{W_i}{b \cdot \Phi_i}\right) & if\ D_i'(0) > \frac{W_i}{b \cdot \Phi_i} \\ 0 & otherwise \end{cases} \tag{9}$$

time on vulnerability discovery and $v_i^* = 0$ time on validating their discoveries.

Proof. First, it is easy to see that the maximum of

$$\mathcal{U}_{H_i}(b, t_i, v_i) = b \cdot \Phi_i \cdot D_i(t_i) - W_i \cdot (t_i + v_i) \tag{10}$$

is always attained at $v_i = 0$. In other words, hackers have no incentive to validate their discoveries, and their optimal decision is $v_i^* = 0$ for every type i.

Second, to find the optimal t_i for the hackers, we take the first derivative of their utility U_{H_i} with respect to t_i:

$$\frac{d}{dt_i} U_{H_i}(b, t_i, 0) = b \cdot \Phi_i \cdot D_i'(t_i) - W_i. \tag{11}$$

The maximum of U_{H_i} is attained either at the lower bound $t_i = 0$ or when the first derivative is equal to 0:

$$\frac{d}{dt_i} U_{H_i}(b, t_i, 0) = 0 \tag{12}$$

$$b \cdot \Phi_i \cdot D_i'(t_i) - W_i = 0 \tag{13}$$

$$D_i'(t_i) = \frac{W_i}{b \cdot \Phi_i}. \tag{14}$$

Since $D_i(t_i)$ is strictly concave, we have that $D_i'(t_i)$ is strictly decreasing. Consequently, if $D_i'(0) \leq \frac{W_i}{b \cdot \Phi_i}$, then Eq. (14) cannot have a positive solution $t_i > 0$, and the maximum utility is attained at the lower bound $t_i^* = 0$.

On the other hand, if $D_i'(0) > \frac{W_i}{b \cdot \Phi_i}$, then there exists a unique solution

$$\tilde{t}_i = (D_i')^{-1} \left(\frac{W_i}{b \cdot \Phi_i} \right) \tag{15}$$

to Eq. (14). Furthermore, it is easy to see that $t_i = 0$ cannot be an optimum in this case, since an infinitesimal increase to $t_i = 0$ would lead to a higher payoff due to $\frac{d}{dt_i} U_{H_i}(b, 0, 0) = b \cdot \Phi_i \cdot D_i'(0) - W_i > 0$. Thus, $t_i^* = \tilde{t}_i$ is the unique optimal choice in this case. □

4.2 Accuracy Threshold

Second, we consider programs that accept reports only from those hackers who maintain a sufficiently high ratio of valid reports (e.g., invitation-only programs or the "Signal Requirements" mechanisms of HackerOne [13]). We model these programs using a policy that imposes a restriction on the participating hackers' accuracy. We define accuracy formally as the following ratio:

$$\frac{\text{number of valid reports}}{\text{number of valid reports} + \text{number of invalid reports}} \tag{16}$$

$$= \frac{\Phi_i \cdot D_i(t_i)}{\Phi_i \cdot D_i(t_i) + (1 - \Phi_i)(D_i(t_i) - I_i(v_i))} \tag{17}$$

$$= \frac{\Phi_i \cdot D_i(t_i)}{D_i(t_i) - (1 - \Phi_i) I_i(v_i)}. \tag{18}$$

Please recall that $I_i(v_i) \leq D_i(t_i)$ by definition, which ensures that accuracy cannot exceed 1.

Based on the above definition of accuracy, we formalize the accuracy-threshold policy as follows.

Definition 1 (Accuracy-Threshold Policy). *Under an accuracy-threshold policy with threshold $\alpha \in [0,1]$, the hackers' choices must satisfy*

$$\frac{\Phi_i \cdot D_i(t_i)}{D_i(t_i) - (1-\Phi_i) I_i(v_i)} \geq \alpha. \tag{19}$$

The following proposition characterizes the hackers' responses to the accuracy-threshold policy when $\alpha > \Phi_i$ (when $\alpha \leq \Phi_i$ their responses are obviously the same as without a policy).

Proposition 2. *Under an accuracy-threshold policy, if $\alpha > \Phi_i$, hackers of type i will spend*

$$t_i^*(b,\alpha) = \begin{cases} 0 & \text{if } D_i'(0) \leq \frac{W_i}{b\cdot\Phi_i - W_i \frac{1}{I_i'(0)} \frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)}} \\ \tilde{t}_i & \text{otherwise} \end{cases} \tag{20}$$

time on vulnerability discovery, where $\tilde{t}_i$ is the unique solution to

$$D_i'(\tilde{t}_i)\left(b\cdot\Phi_i - W_i \frac{1}{I_i'\left(I_i^{-1}\left(D_i(\tilde{t}_i)\frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)}\right)\right)} \frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)}\right) = W_i. \tag{21}$$

In addition, they will spend

$$v_i^*(b,\alpha) = I_i^{-1}\left(D_i(t_i^*)\frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)}\right) \tag{22}$$

time on validating their discoveries.

Note that even though we cannot express the solution of Eq. (21) in closed form, it can be found easily numerically since the left-hand side is strictly decreasing or negative (see the proof for details). Furthermore, this also holds for the remaining propositions (Propositions 3 and 4).

Proof. Similar to the case without any policy, hackers are interested in minimizing their time spent on validating their discoveries. Consequently, for any given t_i, hackers will choose the minimum validation effort v_i^* that satisfies the accuracy-threshold constraint. Hence, we have

$$\frac{\Phi_i \cdot D_i(t_i)}{D_i(t_i) - (1-\Phi_i) I_i(v_i^*)} = \alpha \tag{23}$$

$$\Phi_i \cdot D_i(t_i) = \alpha \cdot D_i(t_i) - \alpha \cdot (1-\Phi_i) I_i(v_i^*) \tag{24}$$

$$I_i(v_i^*) = D_i(t_i)\frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)} \tag{25}$$

$$v_i^* = I_i^{-1}\left(D_i(t_i)\frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)}\right). \tag{26}$$

Note that I_i^{-1} exists since I_i is strictly increasing.

Next, we study the optimal t_i^* for the hackers. Using the above characterization of v_i^*, we can express the hackers' utility as a function of b and t_i:

$$\mathcal{U}_{H_i}(b, t_i) = b \cdot \Phi_i \cdot D_i(t_i) - W_i \cdot \left(t_i + I_i^{-1} \left(D_i(t_i) \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) \right). \tag{27}$$

In order to find the utility-maximizing t_i, we take the first derivative of the hackers' utility with respect to t_i:

$$\begin{aligned} \frac{d}{dt_i} U_{H_i} = \; & b \cdot \Phi_i \cdot D_i'(t_i) - W_i \\ & - W_i \cdot \left(I_i^{-1}\right)' \left(D_i(t_i) \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) \cdot D_i'(t_i) \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \qquad (28) \\ = & D_i'(t_i) \left(b \cdot \Phi_i - W_i \frac{1}{I_i' \left(I_i^{-1} \left(D_i(t_i) \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) \right)} \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) - W_i. \qquad (29) \end{aligned}$$

Recall that $D_i(t_i)$ is a strictly increasing function of t_i by definition. Since $\frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \geq 0$, the argument of I_i^{-1} is increasing in the formula above, which implies that the argument of I_i' is also increasing because I_i^{-1} is an increasing function. Since I_i' is a strictly decreasing function, the value of I_i' is decreasing, which implies that the fraction $\frac{1}{I_i'(\ldots)}$ in the formula above is an increasing function of t_i. Consequently, we have

$$\frac{d}{dt_i} U_{H_i} = \underbrace{D_i'(t_i)}_{\text{strictly decreasing}} \underbrace{\left(b \cdot \Phi_i - W_i \underbrace{\frac{1}{I_i'(\ldots)}}_{\text{increasing}} \underbrace{\frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)}}_{\text{non-negative}} \right)}_{\text{decreasing}} - \underbrace{W_i}_{\text{constant}}. \tag{30}$$

Since $D_i'(t_i)$ is always positive, the first term is either decreasing or negative. Therefore, the following equation has at most one solution for t_i:

$$\frac{d}{dt_i} U_{H_i} = 0. \tag{31}$$

Using an argument similar to the one used in the proof of Proposition 1, we can show that if the above equation has a solution $\tilde{t}_i$, then the unique optimal choice is $t_i^* = \tilde{t}_i$; otherwise, the unique optimal choice is $t_i^* = 0$. Since the first term on the right-hand side of Eq. (30) is either decreasing or negative, $\frac{d}{dt_i} U_{H_i} = 0$ does not have a solution if and only if $\frac{d}{dt_i} U_{H_i}$ is negative at $t_i = 0$. Therefore, $t_i^* = 0$ is the unique optimal choice if and only if

$$0 \geq D_i'(0) \left(b \cdot \Phi_i - W_i \frac{1}{I_i' \left(I_i^{-1} \left(\underbrace{D_i(0)}_{=0} \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) \right)} \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) - W_i \tag{32}$$

$$W_i \geq D_i'(0) \left(b \cdot \Phi_i - W_i \frac{1}{I_i' \left(\underbrace{I_i^{-1}(0)}_{=0} \right)} \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)} \right) \tag{33}$$

$$W_i \geq D_i'(0)\left(b \cdot \Phi_i - W_i \frac{1}{I_i'(0)} \frac{\alpha - \Phi_i}{\alpha \cdot (1-\Phi_i)}\right) \tag{34}$$

$$D_i'(0) \leq \frac{W_i}{b \cdot \Phi_i - W_i \frac{1}{I_i'(0)} \frac{\alpha - \Phi_i}{\alpha \cdot (1-\Phi_i)}}. \tag{35}$$

□

4.3 Report-Rate Threshold

Next, we consider programs that limit the number of reports that each hacker can submit in some fixed time interval (e.g., the "Rate Limiter" mechanism of HackerOne [13]). We model these programs using a policy that imposes a restriction on the participating hackers' submission rate $D_i(t_i) - (1-\Phi_i)I_i(v_i)$. In practice, programs impose these limitations on each hacker individually. To model this, we will assume in this subsection that each hacker type contains only a single hacker. Note that scaling up the analysis to a multitude of hackers is trivial, since hackers having the same parameters will make the same choices, so we can simply add their report numbers together.

We define the rate-threshold policy as follows.

Definition 2 (Rate-Threshold Policy). *Under a rate-threshold policy with threshold $\rho > 0$, the hackers' choices must satisfy*

$$D_i(t_i) - (1-\Phi_i)I_i(v_i) \leq \rho. \tag{36}$$

The following proposition characterizes the hackers' responses to the rate-threshold policy.

Proposition 3. *Under a rate-threshold policy, hackers of type i will spend*

$$t_i^*(b,\rho) = \begin{cases} 0 & if\ D_i'(0) \leq \frac{W_i}{b \cdot \rho} \\ D_i^{-1}(\rho/\Phi_i) & if\ D_i'\left(D_i^{-1}(\rho/\Phi_i)\right) \geq \frac{W_i}{b \cdot \Phi_i - W_i \cdot \left(I_i^{-1}\right)'(\rho/\Phi_i)\frac{1}{1-\Phi_i}} \\ \tilde{t}_i & otherwise \end{cases} \tag{37}$$

time on vulnerability discovery, where $\tilde{t}_i$ is the unique solution to $\frac{d}{dt_i}U_{H_i} = 0$. In addition, they will spend

$$v_i^*(b,\rho) = \begin{cases} 0 & if\ D_i(t_i^*) \leq \rho \\ I_i^{-1}\left(\frac{D_i(t_i^*)-\rho}{1-\Phi_i}\right) & otherwise \end{cases} \tag{38}$$

time on validating their discoveries.

The proof of Proposition 3 can be found online in the extended version of the paper [16].

4.4 Validation Reward

One of the primary reasons for the large number of invalid reports is the misalignment of incentives: hackers are only interested in increasing the number of valid reports, while organizations are also interested in decreasing the number of invalid reports. Existing approaches try to solve this problem by imposing constraints on the hackers' choices (e.g., imposing a threshold on their accuracy or on their report rate). Here, we propose a novel, alternative approach, which incentivizes hackers to reduce the number of invalid reports by rewarding their validation efforts. The advantage of this approach is that it does not impose strict constraints on the hackers' choices, but instead aligns their incentives with those of the organization, and allows the hackers to optimize their productivity.

A validation-reward policy can be formulated in multiple ways. For example, the organization could slightly lower bounties for valid reports, but give a bonus based on the submitter's accuracy. Alternatively, it could raise bounties, but deduct from the payment based on the submitter's rate of invalid reports. Here, we will study the latter approach since it allows us to align the hackers' incentives with those of the organization in a very straightforward way.

In practice, this policy can be easily implemented in the same way as an accuracy or rate threshold, by keeping track of each hacker's valid and invalid reports. Similar to the rate-threshold policy, we will assume for ease of presentation that each hacker type contains only a single hacker.

We define the validation-reward policy as follows.

Definition 3 (Validation-Reward Policy). *Under a validation-reward policy with incentive $\delta > 0$, a hacker's utility is*

$$\mathcal{U}_{H_i}(b, \delta, t_i, v_i) = b \cdot \Phi_i \cdot D_i(t_i) - W_i \cdot (t_i + v_i) - \delta \cdot (1 - \Phi_i)(D_i(t_i) - I_i(v_i)), \quad (39)$$

and the organization's utility is

$$\mathcal{U}_O(b, \delta, \boldsymbol{t}, \boldsymbol{v}) = \sum_i (\hat{V} - b)\Phi_i \cdot D_i(t_i) - (C - \delta)(1 - \Phi_i)\left(D_i(t_i) - I_i(v_i)\right). \quad (40)$$

The following proposition characterizes the hackers' responses to the validation-reward policy.

Proposition 4. *Let*

$$\hat{v}_i = \begin{cases} 0 & if\ I_i'(0) \leq \frac{W_i}{\delta \cdot (1-\Phi_i)} \\ (I_i')^{-1}\left(\frac{W_i}{\delta \cdot (1-\Phi_i)}\right) & otherwise. \end{cases} \quad (41)$$

Under a validation-reward policy, hackers of type i will spend

$$t_i^*(b, \delta) = \begin{cases} 0 & if\ \tilde{v}_i = 0\ and\ D_i'(0) \leq \frac{W_i}{b \cdot \Phi_i - \delta \cdot (1-\Phi_i)} \\ 0 & if\ \tilde{v}_i > 0\ and\ D_i'(0) \leq \frac{W_i}{b \cdot \Phi_i - \frac{W_i}{I_i'(0)}} \\ \tilde{t}_i & otherwise \end{cases} \quad (42)$$

time on vulnerability discovery, where $\tilde{t}_i$ *is the unique solution to* $\frac{d}{dt_i}U_{H_i} = 0$. *In addition, they will spend*

$$v_i^*(b, \delta) = \min\left\{\hat{v}_i, I_i^{-1}(D_i(t_i^*))\right\} \tag{43}$$

time on validating their discoveries.

The proof of Proposition 4 can be found online in the extended version of the paper [16].

5 Numerical Results

In this section, we present numerical results on our bug-bounty model in order to evaluate and compare the policies introduced in Sect. 4. First, in Sect. 5.1, we consider homogeneous hackers by instantiating our model with a single hacker type, and we study the hackers' responses. Second, in Sect. 5.2, we consider heterogeneous hackers and evaluate policies based on the organization's utility.

5.1 Homogeneous Hackers

For the vulnerability-discovery function $D(t)$, we use an instance of *Anderson's thermodynamic model* [3]: $D(t) = \ln(10 \cdot t + 1)$. Note that we added 1 to the argument so that $D(0) = 0$. We instantiate the remainder of our model with the following parameters: $V = 10$, $C = 1$, and a single hacker type with $W_1 = 1$, $\Phi_i = 0.2$, and $I_1(v_1) = \ln(20 \cdot v_1 + 1)$. Notice that these hackers are assumed to be relatively good at validating their discoveries since I_1 grows faster than D. Finally, note that we have experimented with other reasonable parameter combinations as well, and found that the results remain qualitatively the same.

Figure 1 shows the hackers' responses to various policies and the resulting utilities for the organization and the hackers. First, Fig. 1(a) shows that without any policy, the organization attains maximum utility at $b = 2.07$: with lower bounties, hackers dedicate significantly less time to vulnerability discovery (zero time when $b < 0.31$), while with higher bounties, the cost of running the program becomes prohibitively high. In Figs. 1(b), (c), and (d), we set the bounty value to $b = 2.07$ and study the effects of varying the policy parameters.

Figure 1(b) shows that the accuracy-threshold policy is very effective and robust: the organization's utility increases steeply with the threshold α, reaches a 70 % improvement at $\alpha = 0.74$, and declines negligibly after that. In contrast, the rate-threshold policy is considerably less reliable (Fig. 1(c)): the organization's utility is improved by 55 % at $\rho = 0.2$, but it decreases rapidly as the threshold decreases or increases, and it may reach significantly lower values than without a policy. Thus, the organization must implement this policy with great care in order to avoid suppressing productivity. Finally, Fig. 1(d) shows that the validation-reward policy is robust: even though the organization's utility does not increase until the threshold reaches $\delta < 0.66$, it increases steeply after that, reaching and maintaining a 69 % improvement.

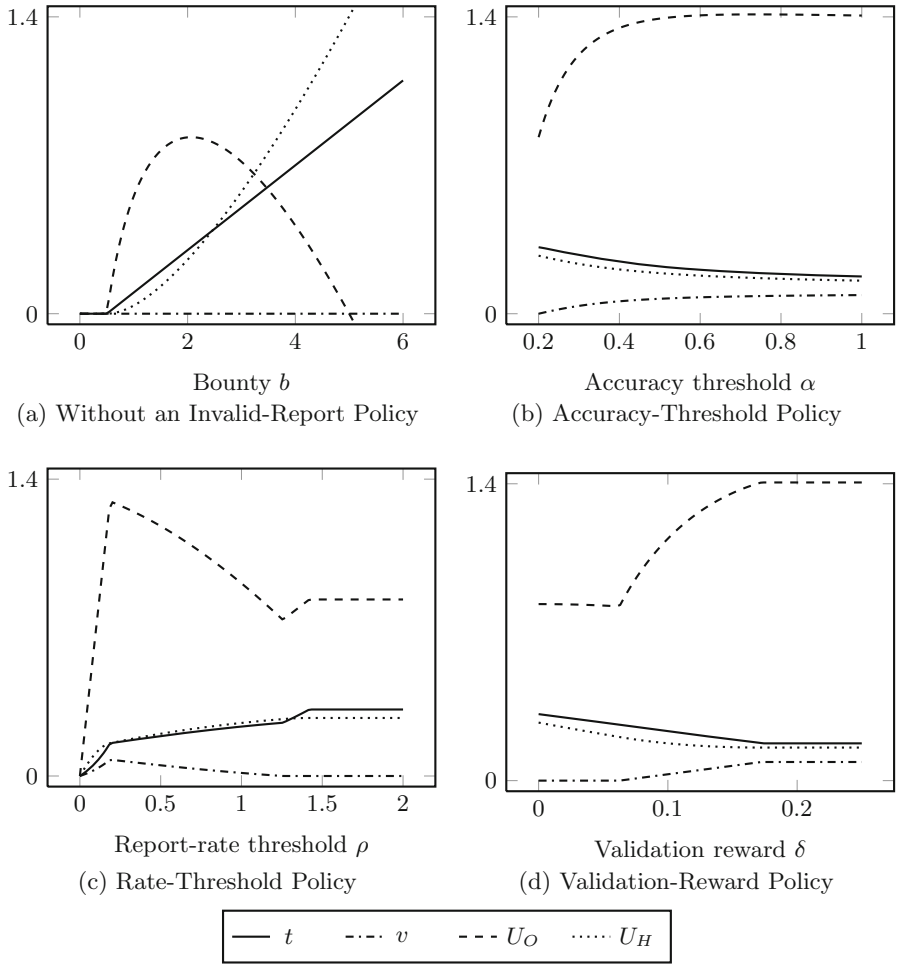
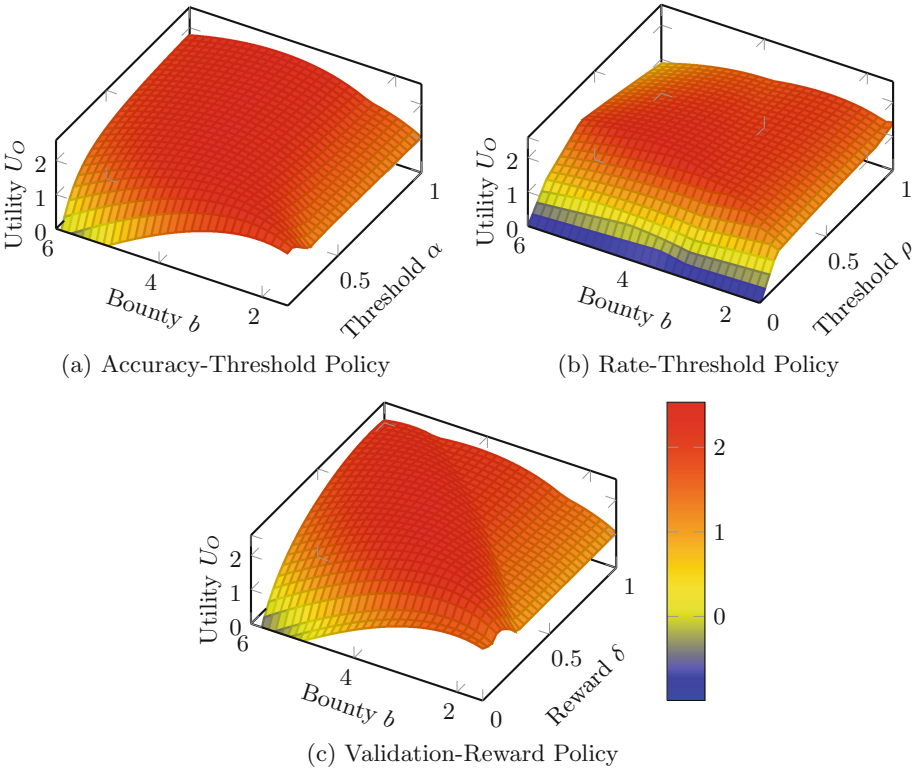


Fig. 1. The organization's and the hackers' utilities (dashed and dotted lines) and the times spent on vulnerability discovery and validation (solid and dash-dotted lines) under various policies as functions of the bounty value.

5.2 Heterogeneous Hackers

Now, we add a second type of hackers, who are worse at validating their discoveries, which we model by letting $I_2(v_2) = \ln(2.5 \cdot v_2 + 1)$ (all other parameters are the same as for the first type). Since we now have multiple hacker types, who may have different responses and utilities, we will plot only the organization's utility for clarity of presentation.

Figure 2 shows the organization's utility under various policies with two types of hackers. Similar to Fig. 1(c), Fig. 2(b) shows that the rate-threshold policy must be implemented carefully since overzealous limiting may signifi-



(a) Accuracy-Threshold Policy

(b) Rate-Threshold Policy

(c) Validation-Reward Policy

Fig. 2. The organization's utility under various policies as a function of the bounty value and policy parameter.

cantly decrease the organization's utility, while lenient limiting is ineffective. On the other hand, the accuracy-threshold and validation-reward policies (Figs. 2(a) and (c)) have large "plateaus" around the optimal values, which make them more robust to changes in configuration or parameter values. Nonetheless, if the bounty value is very low, even these policies – especially the validation-reward policy – may be too strict and deter hackers from participating.

Figure 3 shows the organization's maximum attainable utility under various policies with two types of hackers. For each policy and bounty value, we searched over possible values of the policy parameter space (i.e., $\alpha = 0.2, 0.21, \ldots, 1$; $\rho = 0, 0.05, \ldots, 5$; or $\delta = 0, 0.012, \ldots, 1.2$) and plotted the maximum utility. Since the two hacker types differ only in their validation performance, the utility values without a policy shown by Fig. 3 are proportional to the values shown by Fig. 1(a), and the maximum is again attained at $b = 2.07$. Compared to this baseline, the accuracy-threshold, rate-threshold, and validation-reward policies can attain 31 %, 13 %, and 52 % improvement, respectively. However, if the bounty value is not high enough, none of the policies can improve the organization's utility. Finally, offering validation rewards outperforms the other policies

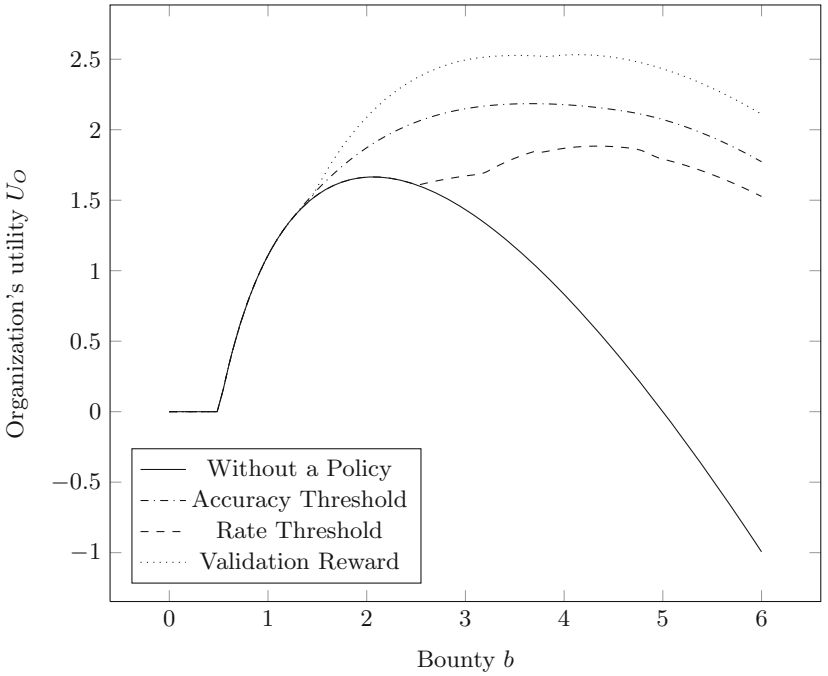


Fig. 3. The organization's maximum attainable utility under various policies as a function of the bounty value.

significantly, since it is able to incentivize heterogeneous hackers to operate at their individual maxima instead of forcing them towards a uniform strategy.

6 Conclusion

In this paper, we provided the first theoretical framework for modeling policies for reducing the number of invalid reports in bug-bounty programs. Using our framework, we investigated a set of canonical policies, and studied the hackers' responses to these policies, showing that each type has a unique response to each policy. In addition to studying existing policies, we also proposed a new policy that incentivizes hackers without restricting their actions.

Based on numerical analyses, we found that all of the considered policies may substantially improve an organization's utility, which explains their widespread use [13]. However, their effectiveness and reliability vary significantly. We found that the rate-threshold policy is not only less effective than the other two, but it must also be configured more carefully. In contrast, the accuracy-threshold and validation-reward policies are less sensitive to changes in parameter and configuration values, and they can also be more effective. However, without adequate bounties, even these policies might "backfire" and actually deter hackers from

dedicating time to vulnerability discovery. Finally, we found that the validation-reward policy may significantly outperform the other two when hackers are not homogeneous, since it allows hackers to operate at their individual optima.

In future work, we plan to extend our model and analyses by considering combinations of policies. In other words, we will consider organizations that implement multiple policies at the same time. Building on our current analysis, we will study how hackers respond to various policy-combinations, and we will explore which combinations are the most effective and robust.

Acknowledgements. This work was supported in part by FORCES (Foundations Of Resilient CybEr-Physical Systems), which receives support from the National Science Foundation (NSF award numbers CNS-1238959, CNS-1238962, CNS-1239054, CNS-1239166).

References

1. Algarni, A., Malaiya, Y.: Software vulnerability markets: discoverers and buyers. Int. J. Comput. Inf. Sci. Eng. **8**(3), 71–81 (2014)
2. Alhazmi, O., Malaiya, Y.: Modeling the vulnerability discovery process. In: 16th IEEE International Symposium on Software Reliability Engineering (ISSRE) (2005)
3. Anderson, R.: Security in open versus closed systems - The dance of Boltzmann, Coase and Moore. In: Open Source Software Economics (2002)
4. Bacon, D., Chen, Y., Parkes, D., Rao, M.: A market-based approach to software evolution. In: 24th ACM SIGPLAN Conference Companion on Object Oriented Programming, Systems, Languages, and Applications (2009)
5. Böhme, R.: A comparison of market approaches to software vulnerability disclosure. In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 298–311. Springer, Heidelberg (2006)
6. Brady, R., Anderson, R., Ball, R.: Murphy's law, the fitness of evolving species, and the limits of software reliability. Technical Report 471, University of Cambridge, Computer Laboratory (1999)
7. Bugcrowd: The state of bug bounty, July 2015
8. Doupé, A., Cova, M., Vigna, G.: Why Johnny can't pentest: an analysis of black-box web vulnerability scanners. In: Kreibich, C., Jahnke, M. (eds.) DIMVA 2010. LNCS, vol. 6201, pp. 111–131. Springer, Heidelberg (2010)
9. Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A., Wagner, D.: An empirical study on the effectiveness of security code review. In: Jürjens, J., Livshits, B., Scandariato, R. (eds.) ESSoS 2013. LNCS, vol. 7781, pp. 197–212. Springer, Heidelberg (2013)
10. Egelman, S., Herley, C., van Oorschot, P.: Markets for Zero-Day Exploits: Ethics and Implications. In: New Security Paradigms Workshop (2013)
11. Finifter, M., Akhawe, D., Wagner, D.: An empirical study of vulnerability rewards programs. In: USENIX Security Symposium (2013)
12. Frei, S., Schatzmann, D., Plattner, B., Trammell, B.: Modeling the security ecosystem - The dynamics of (in)security. In: Economics of Information Security and Privacy (2009)
13. HackerOne: Improving public bug bounty programs with signal requirements. HackerOne Blog, March 2016. https://hackerone.com/blog/signal-requirements

14. Huang, C., Liu, J., Fang, Y., Zuo, Z.: A study on web security incidents in China by analyzing vulnerability disclosure platforms. Comput. Secur. **58**, 47–62 (2016)
15. Kannan, K., Telang, R.: Market for software vulnerabilities? Think again. Manage. Sci. **51**(5), 726–740 (2005)
16. Laszka, A., Zhao, M., Grossklags, J.: Optimal policies for bug bounty programs (extended version) (2016). http://aronlaszka.com/papers/laszka2016banishing.pdf
17. Libicki, M., Ablon, L., Webb, T.: The Defenders Dilemma: Charting a Course Toward Cybersecurity. Rand Corporation (2015)
18. Maillart, T., Zhao, M., Grossklags, J., Chuang, J.: Given enough eyeballs, all bugs are shallow? Revisiting Eric Raymond with bug bounty markets. In: Workshop on the Economics of Information Security (WEIS) (2016)
19. Ozment, A.: Bug auctions: Vulnerability markets reconsidered. In: Workshop on the Economics of Information Security (WEIS) (2004)
20. Ozment, A.: The likelihood of vulnerability rediscovery and the social utility of vulnerability hunting. In: Workshop on the Economics of Information Security (WEIS) (2005)
21. Ozment, A., Schechter, S.: Milk or wine: Does software security improve with age? In: USENIX Security Symposium (2006)
22. Ransbotham, S., Mitra, S., Ramsey, J.: Are markets for vulnerabilities effective? MIS Q. **36**(1), 43–64 (2012)
23. Rescorla, E.: Is finding security holes a good idea? IEEE Secur. Priv. **3**(1), 14–19 (2005)
24. Schechter, S.E.: How to buy better testing. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) InfraSec 2002. LNCS, vol. 2437, pp. 73–87. Springer, Heidelberg (2002)
25. Shahzad, M., Shafiq, M., Liu, A.: A large scale exploratory analysis of software vulnerability life cycles. In: International Conference on Software Engineering (2012)
26. Van Goethem, T., Piessens, F., Joosen, W., Nikiforakis, N.: Clubbing seals: Exploring the ecosystem of third-party security seals. In: 21st ACM SIGSAC Conference on Computer and Communications Security (CCS) (2014)
27. Zhao, M., Grossklags, J., Chen, K.: An exploratory study of white hat behaviors in a web vulnerability disclosure program. In: 2014 ACM CCS Workshop on Security Information Workers (2014)
28. Zhao, M., Grossklags, J., Liu, P.: An empirical study of web vulnerability discovery ecosystems. In: 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS) (2015)
29. Zhao, M., Liu, P.: Empirical analysis and modeling of black-box mutational fuzzing. In: Caballero, J., Bodden, E., Athanasopoulos, E. (eds.) ESSoS 2016. LNCS, vol. 9639, pp. 173–189. Springer, Heidelberg (2016)