Devising Effective Policies for Bug-Bounty Platforms and Security Vulnerability Discovery

Mingyi Zhao[1], Aron Laszka[2], and Jens Grossklags[3]
[1]Snap Inc. [2]Vanderbilt University [3]Technical University of Munich

Abstract

Bug-bounty programs have the potential to harvest the effort and diverse knowledge of thousands of independent security researchers, but running them at scale is challenging due to misaligned incentives and misallocation of effort. In our research, we discuss these challenges in detail and present relevant empirical data. We develop an economic framework consisting of two models that focus on evaluating different policies for improving the effectiveness of bug-bounty programs. Further, we discuss regulatory-policy challenges and questions related to vulnerability research and disclosure, such as mandatory bug bounties and the relation to other cyber-security policies.

Keywords: Bug-bounty programs, Vulnerability discovery, Economics of security, White-hat (ethical) hackers, Misaligned incentives, Crowdsourcing.

1 Introduction

Despite significant progress in software-engineering practices, software utilized for desktop and mobile computing remains insecure. At the same time, the consumer and business information handled by these software products is increasing in its richness and monetization potential, which triggers significant privacy and security concerns.

Numerous challenges stand in the way of secure code. Feature-rich complex software products are inherently difficult to develop securely, in particular, if time-to-market, outdated legacy code bases, and other business realities impede careful engineering practices.[1] Instead, companies are increasingly harvesting the potential of external (ethical) security researchers to crowdsource efforts to find and ameliorate security vulnerabilities.[2] These so-called white-hat researchers are often rewarded with monetary payments (i.e., bounties) and publicly recognized.

1. Ross Anderson, "Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore," in *Proceedings of Open Source Software: Economics, Law and Policy* (2002); Robert Hahn and Anne Layne-Farrar, "The Law and Economics of Software Security," *Harvard Journal of Law & Public Policy* 30, no. 1 (2006): 283–353.

2. Matthew Finifter, Devdatta Akhawe, and David Wagner, "An empirical study of vulnerability rewards programs," in *USENIX Security Symposium* (2013); Mingyi Zhao, Jens Grossklags, and Peng Liu, "An

Broadening the appeal of crowdsourced security, several commercial bug-bounty platforms have emerged (e.g., HackerOne[3], BugCrowd[4], Cobalt[5]) and successfully facilitate the process of building and maintaining bug-bounty programs for companies and organizations. For example, on HackerOne, over 44,000 security vulnerabilities have been reported and fixed for hundreds of organizations. Contributions came from over 5000 different white-hat hackers, and paid bounties have surpassed the US$10M threshold (data from May 2017).

Recently, researchers have begun to systematically study these platforms from an empirical perspective to evidence their growing popularity and practical contributions to the security of deployed code.[6] In particular, researchers found that vulnerabilities are increasingly hard to find in the code of participating companies. This implies that bug-bounty programs make a significant contribution to code security, and that these companies are less likely to fall victim to cybercriminals and nation-state sponsored security compromises as a result of zero-day exploits.

The problem area we are addressing with our work is how to scale the promise of crowdsourced security to a more significant number of companies and white-hats. To achieve this goal, it is important to design effective policies for bug-bounty, and we will discuss two classes of policies, *engagement policy* and *regulatory policy*. Engagement policies are developed and enforced by bug-bounty programs and platforms to encourage white-hats to make valuable contributions to organizations, whereas regulatory policies shape the bug-bounty ecosystem from the outside.

As a major part of our work, we discuss several key challenges in the bug-bounty ecosystem which we are addressing by designing and evaluating engagement policies. First, a major obstacle to scaling bug-bounty platforms to a larger number of white hats is inherent in the concept of crowdsourced security. White-hat security researchers are interested in receiving as many bounties as possible while minimizing effort. However, these efforts include the validation of findings before reporting them to companies; a lack thereof results in a higher number of invalid reports which significantly burden participating organizations. The percentage of invalid reports is currently significant, ranging from 35% to 55% on different platforms.

To address this first problem area, we devise a theoretical model for evaluating approaches for reducing the number of invalid reports. We then investigate the strengths and weaknesses of canonical approaches taken by bug-bounty platforms. These existing policies increase reports quality by restricting participation. However, such policies might “backfire" and actually deter hackers from dedicating time to vulnerability discovery. We introduce a novel validation-reward policy, aiming to further improve efficiency by enabling different white hats to exert validation effort at their individually

empirical study of web vulnerability discovery ecosystems,” in *22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2015).

3. `https://hackerone.com/`
4. `https://bugcrowd.com/`
5. `https://cobalt.io/`
6. Mingyi Zhao, Jens Grossklags, and Kai Chen, “An Exploratory Study of White Hat Behaviors in a Web Vulnerability Disclosure Program,” in *2014 ACM CCS Workshop on Security Information Workers* (2014); Zhao, Grossklags, and Liu, “An empirical study of web vulnerability discovery ecosystems”; Thomas Maillart et al., “Given Enough Eyeballs, All Bugs are Shallow? Revisiting Eric Raymond with Bug Bounty Markets,” in *Workshop on the Economics of Information Security (WEIS)* (2016).

optimal levels.

Second, with a growing number of participating companies, the problem of efficiently allocating and distributing the valuable, but scarce effort, of white-hat researchers is becoming paramount. On the one hand, it is self-evident that newly joined organizations harbor the greatest potential for security improvements. On the other hand, organizations who already benefited from crowdsourced security wish to continue reaping the benefits of white hats' scrutiny even though finding new vulnerabilities becomes increasingly difficult.[7] The need arises to devise sophisticated economic policies to manage this trade-off with various measures, including optimal adjustments to monetary bounties.

Responding to this second challenge, we introduce an economic model to study the allocation of hackers in bug-bounty programs. Our model captures the vulnerability discovery process by hackers with an emphasis on capturing the diversity of hackers' capability in identifying vulnerabilities. The model illustrates why more participation is not always better, and how bug bounty programs can design more effective hacker allocation plans.

In addition to engagement policies, bug-bounty programs can potentially benefit from clear regulatory policies of vulnerability research and disclosure. We discuss several challenges for creating such regulatory policies. We also discuss how existing policies, such as the Wassenaar arrangement, could potentially affect bug-bounty negatively.

Roadmap: We proceed as follows. In Section 2, we further illustrate the highlighted challenges to bug bounty platforms with selected data. We develop and analyze the model to evaluate policies for incentivizing validation efforts in Section 3. We then present our model to evaluate policies for allocating discovery efforts in Section 4. In Section 5, we discuss regulatory challenges faced by bug bounty programs. We discuss related work in Section 6. We summarize our results and offer concluding remarks in Section 7.

2 Engagement Policy Challenges

2.1 Bug-bounty Rules

Bug-bounty is often perceived as a risky approach for improving security, because an organization is asking largely anonymous and independent hackers from all over the world to probe and test its software systems remotely. Organizations could worry that some white-hats might damage the production system or steal user data when doing vulnerability research, or disclose vulnerabilities found to other parties, or even to the public. On the other hand, white-hats would also worry that they might face legal charges for vulnerability research and disclosure. One might expect to have regulatory policies to control the risk. However, as of today, regulatory policies for bug-bounty are a murky area, and we will discuss them in more detail in Section 5.

7. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems"; Maillart et al., "Given Enough Eyeballs, All Bugs are Shallow? Revisiting Eric Raymond with Bug Bounty Markets."

At present, bug-bounty programs, or bug-bounty platforms that host individual programs, enforce several types of *engagement policies* for bug-bounty participants to reduce risk. The most common form of engagement policy is the development of *bug-bounty rules*. These rules specify which part of the software system can be assessed, and what kind of actions are permitted.[8] For example, some organizations will exclude websites that belong to newly acquired subsidiaries, and many organizations prohibit the usage of automated vulnerability scanners which could affect normal business traffic. In addition, the program rules also state that if white-hats comply with the program policy during vulnerability research, the organization will not take legal action against the white-hats. In other words, bug-bounty programs become a safe(r) harbor for white-hats to conduct vulnerability research.

The intention of having bug-bounty rules is not only to reduce risk, but also to improve the quality of reports submitted by white-hats. By clearly stating what scope and what types of security issues are of interest, organizations can better concentrate white-hats' valuable manpower on the most important areas of its attack surface. However, the effect of bug-bounty rules on report quality is rather unclear, and more research is needed. If we look at some data from today's bug bounty program in Figure 1, we can easily see two major challenges: a high percentage of invalid reports, and a significant amount of duplicated effort. To address these challenges, organizations and bug-bounty platforms have designed and implemented additional kinds of engagement policies. We will discuss these engagement policies in the remainder of this section.

2.2 Incentive Policies and Invalid Reports

In addition to defining rules, bug-bounty programs also enforce various kinds of incentive policies to motivate white-hats. For example, a bug bounty program usually gives out rewards in proportion to the severity of the vulnerability found, in order to encourage white-hats to search for more important issues. Rewards could be money, reputation points, or a place in the hall of fame.

Previous research has found that the average (monetary) bounty amount is positively correlated with the number of valid reports received.[9] However, at the same time, organizations that run bug-bounty programs could be easily overwhelmed by invalid reports (also referred to as noise), which 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 (i.e., issues that do exist but are explicitly excluded by bug-bounty program rules, for example, such as bugs in software products from recently acquired companies or affiliates). 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".[10]

In practice, the number of invalid reports is significant. Figure 1 shows some relevant

8. See, for example, the bug-bounty rules for Twitter on the HackerOne platform: `https://hackerone.com/twitter`.

9. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."

10. HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*, *HackerOne Blog*, `https://hackerone.com/blog/signal-requirements`, March 2016.

statistics of two independent bug-bounty programs run by Facebook[11] and Google,[12] and two bug-bounty platforms, HackerOne and BugCrowd.[13] We can observe that the percentage of valid reports is lower than 25% for all programs and platforms. The value is particularly low for the independent bug-bounty programs ran by Facebook and Google, and higher for bug-bounty programs hosted on the platforms. One reason behind this difference is that the platforms have better identification practices for participants, accumulate rich data about white-hat researchers, and can thus enforce various kinds of quality-control policies. In addition, some bug-bounty programs on these platforms are private, meaning that they are only open to well-established and experienced white-hats, who are more likely to submit valid reports.

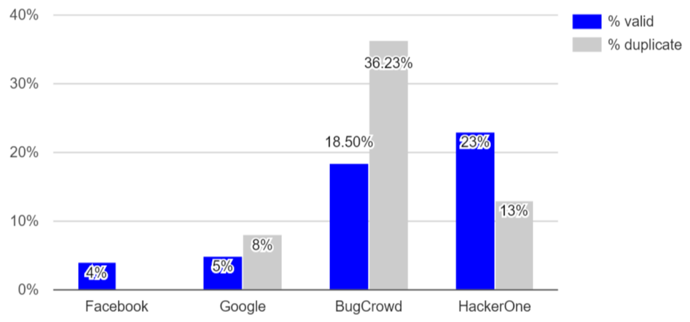


Figure 1: Comparing the percentages of valid reports and duplicate reports across different bug-bounty programs and platforms. The percentage of duplicate reports for Facebook is unknown. The percentage of duplicates obtained from HackerOne might be underestimated, because program administrators are not required to mark a report as duplicate.

We also worked with HackerOne to obtain additional data of its public bug-bounty programs and private programs. Figure 2 shows that private programs have a much higher percentage of valid reports (around 45%). However, private programs are not designed to efficiently utilize the wider crowd of white-hats, which has also been shown to be a valuable contributor of vulnerability discoveries with higher severity levels.[14] For that reason, even though the public bug-bounty programs of Facebook and Google have a low signal percentage (i.e., percentage of valid reports), the absolute number of valid reports submitted to their programs is high. Therefore, these companies have been strongly supporting bug-bounty in the past several years. Figure 2 also shows that since April 2015, there is a continuous decrease of noise (from 52% to 30% in public programs, and from 23% to 10% in private programs), and an increase of valid reports (from 13% to 20% in public programs, and from 35% to 52% in private programs)

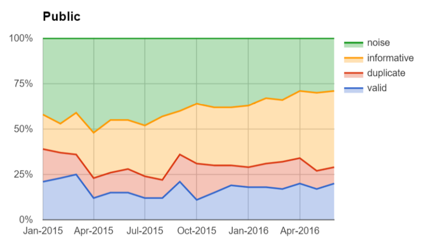
11. Facebook, *2015 Highlights: Less Low-Hanging Fruit*, `https://www.facebook.com/notes/facebook-bug-bounty/2015-highlights-less-low-hanging-fruit/1225168744164016`, 2016.

12. Google, *Statistics and Charts of Google VRP*, `https://sites.google.com/site/bughunteruniversity/behind-the-scenes/charts/2014`, 2014.

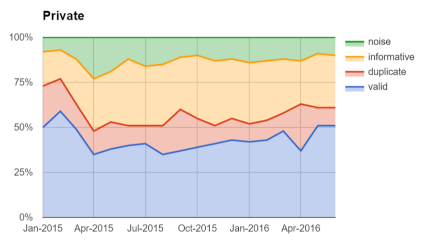
13. Bugcrowd, *The State of Bug Bounty*, June 2016.

14. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."

on HackerOne. A main reason behind this trend is HackerOne's constant effort in improving the signal-to-noise ratio.[15] Nevertheless, we see that there is still ample space for improvements, particularly for public programs.



(a) Public



(b) Private

Figure 2: Trend of report types for public programs and private programs on HackerOne. In our paper, we consider noise and so-called informative reports 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 high false-positive rates.[16] Since filtering out false positives is costly, some hackers may prefer to send the outputs of an automated scanner to the bug-bounty program, hoping that some of them may be recognized with a reward. Further, some discoveries may initially appear to be valid, but after further inspection they may fail to prove a genuine security vulnerability. Another important facet is that the hacker should clearly explain in the report what the discovered flaw is, and how it can lead to a security problem. Failure of articulating the issue could result in the report being marked as invalid, or the report being accepted only after many rounds of communication between the hacker and an organization's security team. However, writing a clear report takes time and effort from the hackers, who might not be willing to spend the effort.

As a direct response, bug-bounty platforms have started to offer different incentive 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.[17] 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,[18] these measures together have decreased the percentage of noise to around 25%.

15. HackerOne, *Improving Signal Over 10,000 Bugs*, *HackerOne Blog*, `https://hackerone.com/blog/improving-signal-over-10000-bugs`, July 2015; HackerOne, *Expanding Reputation: Introducing Signal and Impact*, *HackerOne Blog*, `https://hackerone.com/blog/introducing-signal-and-impact`, December 2015; HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*.

16. Adam Doupé, Marco Cova, and Giovanni Vigna, "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners," in *Detection of Intrusions and Malware, and Vulnerability Assessment* (2010).

17. HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*.

18. Ibid.

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.

2.3 Allocation Policies and Duplicate Reports

Another type of engagement policy is allocation (or invitation) policy, which determines which white-hats can participate in a given bug-bounty program. Some leading organizations in adopting bug bounty, such as Yahoo, Facebook and Google, allow everyone to participate. We will refer to such programs as *public programs*. However, for many other organizations, running a public program might provide an overwhelming number of reports that they cannot handle. For some other organizations, such as the U.S. Department of Defense,[19] only certain types of white-hats (e.g., U.S. citizens), are allowed. Therefore, these organizations will run *private programs* that are only open to a subset of all white-hats. Currently, HackerOne randomly selects white-hats into private programs, with selection probability determined by white-hats' previous performance.[20]

Our observation is that existing allocation policies have significant inefficiencies in utilizing the manpower of white-hats. Such inefficiencies are demonstrated in one key challenge for bug-bounty programs: *duplicate reports*. After a bug is first reported to an organization, it usually takes 1 - 2 months for the organization to fix the issue. Therefore, before the fix is completed, other white-hats might spend effort to discover and report the same issue. However, according to the predominantly used program rules, only the first discoverer will be rewarded, while other reporters' efforts are not properly compensated (and may only receive some small form of recognition). In addition, the organization also needs to spend effort triaging these duplicated reports, and interacting with the reporting (frustrated) white-hats.

The percentage of duplicates is quite high for bug-bounty programs, as we can seen in Figures 1 and 2. For Google and BugCrowd, the percentage of duplicates is higher than the percentage of valid reports. The duplicate rate is lower on HackerOne. However, since marking a report as duplication is not mandatory, it is possible that the reported number underestimates the true magnitude of the problem. Unfortunately, we do not know the duplicate rate of Facebook's bug-bounty program. But Facebook reported one interesting case in an annual report:[21] when `messenger.com` was launched, within minutes, 15 hackers filed a report for the same Cross-site Request Forgery issue. While this case shows how powerful bug-bounty programs are in finding vulnerabilities

19. Shannon Collins, *DoD Announces 'Hack the Pentagon' Follow-Up Initiative*, *DoD News, Defense Media Activity*, `https://www.defense.gov/News/Article/Article/981160/dod-announces-hack-the-pentagon-follow-up-initiative`, October 2016.

20. HackerOne, *Fair and Transparent Hacker Invitations*, *HackerOne Blog*, `https://hackerone.com/blog/fair-and-transparent-hacker-invitations`, March 2016.

21. Facebook, *2015 Highlights: Less Low-Hanging Fruit*.

	25th	50th	75th
First Response	0.1	0.8	4.1
Triage	0.4	1.8	7.1
Bounty	4.6	16.7	52.7
Fix	4.7	20.9	66.6

Table 1: Percentiles of organizations' response efficiency on HackerOne (in days).

quickly, it also demonstrates the inefficiency in utilizing hacker's valuable manpower.

One obvious way of reducing the number of duplicates is to shorten the time window between a bug being reported and the same bug being fixed. Such effort also reduces the length of time window for exploiting the vulnerability, and generally increases the speed of hackers receiving rewards.

Most bug bounty programs today are created after releasing the software product. However, developing and deploying a patch of the product usually requires multiple steps, and coordination between multiple departments of the affected organization. Considering these factors, it would be hard to expect an organization, particularly a larger one, to fix bugs very rapidly. HackerOne has reported several statistics about how long it takes for organizations to respond and fix a bug,[22] shown in Table 1. As we can see, the median time to address a vulnerability is 21 days, and the 75th percentile is more than 2 months. Within the time period, it is not unlikely for other hackers to find the same bug, particularly if the incentives are high, like in the case of Facebook, which paid more than $1700 per bug on average in 2015.[23]

We also want to point out that, despite the negative impact of duplicated reports, they also provide some value to bug-bounty. For example, duplicated reports can help bug-bounty programs to further understand and verify certain issues, and to get a sense of which fixes to prioritize based on discovery frequency. In addition, the organization and the bug-bounty platform can gain a better understanding of researcher expertise based on duplicated reports. How to utilize the duplicated vulnerability discovery effort more effectively is left as a future work.

2.4 Summary

In this section, we discussed three types of engagement policies for bug-bounty: bug-bounty rules, incentive policies and allocation policies. We also discussed two major challenges, invalid reports and duplicate reports, that are associated with engagement policies. In Section 3 and 4, we will analyze existing policies, and then propose and evaluate several new policy approaches, using the methodology of economic modeling.

22. HackerOne, *The HackerOne Success Index - Response Efficiency*, *HackerOne Blog*, `https://hackerone.com/blog/response-efficiency`, February 2016.

23. Facebook, *2015 Highlights: Less Low-Hanging Fruit*.

3 Policies for Incentivizing Validation Effort

In this section, we study policies for incentivizing hackers to validate their discoveries before submitting them, thereby reducing the ratio of invalid reports. We first consider two canonical policies, called *accuracy threshold* and *rate threshold*. These policies model real-world mechanisms available on HackerOne, called “Signal Requirements” and “Rate Limiter.” Then, we propose a novel policy, called *validation reward*, which aims to align the incentives of hackers with the organization. To analyze these policies, we introduce a mathematical model of discovery and validation, which we use to characterize the hackers’ behavior. Finally, we present numerical results comparing the policies. The key findings of our analysis are summarized in Section 3.4.

This section is organized as follows. In Section 3.1, we introduce our economic model of valid and invalid reports in bug-bounty programs. In Section 3.2, we study a set of canonical policies for decreasing the number of invalid reports. In Section 3.3, we present numerical results on these policies. Finally, in Section 3.4, we offer concluding remarks on this problem area and outline future work.

3.1 Model

In this section, we introduce the economic model that we use to study incentivizing validation effort in bug-bounty programs. Note that we will focus exclusively on features that are relevant to invalid reports and policies for limiting them. A list of symbols used throughout Section 3 can be found in Table 2.

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

Table 2: List of Symbols in Section 3

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

supported by experimental results and existing models (e.g.,[24]).

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., verified 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.[25] 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

24. Mingyi Zhao and Peng Liu, "Empirical Analysis and Modeling of Black-Box Mutational Fuzzing," in *International Symposium on Engineering Secure Software and Systems (ESSoS)* (2016); Omar Alhazmi and Yashwant Malaiya, "Modeling the vulnerability discovery process," in *16th IEEE International Symposium on Software Reliability Engineering (ISSRE)* (2005); Robert Brady, Ross Anderson, and Robin Ball, *Murphy's law, the fitness of evolving species, and the limits of software reliability*, technical report 471 (University of Cambridge, Computer Laboratory, 1999).

25. Validation is actually a multiple-step process, which includes verifying that the discovery is an actual vulnerability, that it is within the scope of the program, etc. We let $I_i(v_i)$ denote the number of vulnerabilities that are completely verified by the hacker. We consider the other $D_i(t_i) - I_i(v_i)$ discoveries to be non-validated.

$D_i(t_i) - I_i(v_i)$ non-validated discoveries, of which $\Phi_i \cdot (D_i(t_i) - I_i(v_i))$ are valid and $(1-\Phi_i)\,(D_i(t_i) - I_i(v_i))$ are invalid. Hence, the number of valid reports made by hackers of type i is

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

while the number of invalid reports is

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

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. In practice, the constant W_i can be estimated using the average hourly wage of a hacker of type i (e.g., hourly wage taking into account typical expertise and qualifications).

The organization's utility is

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

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.

3.2 Policies

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.

3.2.1 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 ([26]). *Suppose that the organization institutes no policy for limiting invalid reports. Then, if the marginal utility of vulnerability discovery at $t_i = 0$ is less than or equal to zero (i.e., $b \cdot \Phi_i \cdot D_i'(0) - W_i \leq 0$), hackers of type i will not spend any time on vulnerability discovery. Otherwise, they will spend $\left(D_i'\right)^{-1}\left(\frac{W_i}{b \cdot \Phi_i}\right)$ time on vulnerability discovery, but they will not spend any time on validating their discoveries.*

In other words, if the organization chooses a bounty value lower than $\frac{W_i}{\Phi_i \cdot D_i'(0)}$, then it will not receive any reports from hackers of type i. If it chooses a higher bounty value, it will receive some reports, but the ratio of valid reports will be only Φ_i.

3.2.2 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[27]). We model these programs using a policy that imposes a restriction on the participating hackers' accuracy. We define accuracy formally as the following ratio:

$$\frac{\text{number of valid reports}}{\text{number of valid reports} + \text{number of invalid reports}}. \tag{8}$$

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

$$\frac{\Phi_i \cdot D_i(t_i)}{D_i(t_i) - (1 - \Phi_i) I_i(v_i)} \geq \alpha. \tag{9}$$

The following proposition characterizes the hackers' responses to the accuracy-threshold policy.[28]

Proposition 2 ([29]). *Suppose that the organization institutes an accuracy-threshold policy with threshold $\alpha > \Phi_i$. Then, if $b \cdot \Phi_i \cdot D_i'(0) \leq W_i \left(1 + D_i'(0) \frac{1}{I_i'(0)} \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)}\right)$, hackers of type i will not spend any time on vulnerability discovery. Otherwise, they will spend $\tilde{t}_i$ time on vulnerability discovery and $I_i^{-1}\left(D_i(\tilde{t}_i) \frac{\alpha - \Phi_i}{\alpha \cdot (1 - \Phi_i)}\right)$ time on validation,*

26. Aron Laszka, Mingyi Zhao, and Jens Grossklags, "Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms," in *21st European Symposium on Research in Computer Security (ESORICS)* (2016), 161–178.

27. HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*.

28. Note that we only consider the case when the accuracy threshold α is higher than the ratio Φ_i of discoveries that are valid. When $\alpha \leq \Phi_i$, the hackers' responses are obviously the same as without a policy.

29. Laszka, Zhao, and Grossklags, "Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms."

where $\tilde{t}_i$ is the unique solution to[30]

$$D_i'(\tilde{t}_i)\left(b \cdot \Phi_i - W_i \frac{1}{I_i'\left(I_i^{-1}\left(D_i(\tilde{t}_i)\frac{\alpha-\Phi_i}{\alpha\cdot(1-\Phi_i)}\right)\right)} \frac{\alpha - \Phi_i}{\alpha \cdot (1-\Phi_i)}\right) = W_i. \tag{10}$$

In sum, using an accuracy-threshold policy, an organization can ensure a higher ratio of valid reports. However, in exchange, the organization must also choose a higher bounty value to incentivize participation, as the minimum bounty can be much higher for an accuracy-threshold policy $\left(b \geq \frac{W_i\left(1+D_i'(0)\ldots\right)}{\Phi_i \cdot D_i'(0)}\right)$ than without a policy $\left(b \geq \frac{W_i}{\Phi_i \cdot D_i'(0)}\right)$.

3.2.3 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[31]). 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

$$D_i(t_i) - (1-\Phi_i)I_i(v_i) \leq \rho. \tag{11}$$

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

Proposition 3 ([32])**.** *Suppose that the organization institutes a rate-threshold policy with threshold ρ. Then, if the marginal utility of vulnerability discovery at $t_i = 0$ is less than or equal to zero (i.e., $b \cdot \Phi_i \cdot D_i'(0) - W_i \leq 0$), hackers of type i will not spend any time on vulnerability discovery. Otherwise, they will spend $\tilde{t}_i$ time on vulnerability discovery, where $\tilde{t}_i$ is the unique solution to $\frac{d}{dt_i}\mathcal{U}_{H_i} = 0$ subject to the rate-threshold policy. In this case, they will also spend $I_i^{-1}\left(\frac{D_i(t_i^*)-\rho}{1-\Phi_i}\right)$ time on validation if $D_i(\tilde{t}_i) \leq \rho$; otherwise, they will not spend any time on validation.*

With respect to bounty values, rate thresholds can be viewed as a cheaper alternative to accuracy thresholds, as the minimum bounty value for participation with rate

30. Note that even though we cannot express the solution of Equation (10) in closed form, it can be found easily numerically since the left-hand side is strictly decreasing or negative. Furthermore, this also holds for the remaining propositions (Propositions 3 and 4).

31. HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*.

32. Laszka, Zhao, and Grossklags, "Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms."

thresholds is the same as without a policy. On the other hand, rate thresholds can also be less effective than accuracy thresholds, since some hackers may participate without spending any time on validating their discoveries.

3.2.4 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

$$\mathcal{U}_{H_i}(b, \delta, t_i, v_i) = b \cdot \Phi_i \cdot D_i(t_i) - W_i \cdot (t_i + v_i) - \delta \cdot (1 - \Phi_i)(D_i(t_i) - I_i(v_i)), \quad (12)$$

and the organization's utility is

$$\mathcal{U}_O(b, \delta, \boldsymbol{t}, \boldsymbol{v}) = \sum_i (V - C - b)\Phi_i \cdot D_i(t_i) - (C - \delta)(1 - \Phi_i)\left(D_i(t_i) - I_i(v_i)\right). \quad (13)$$

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

Proposition 4 ([33])**.** *Suppose that the organization institutes a validation-reward policy with incentive δ, and let the* desired validation effort $\hat{v}_i$ *of type i hackers be*

$$\hat{v}_i = \begin{cases} 0 & \text{if } I_i'(0) \leq \frac{W_i}{\delta \cdot (1 - \Phi_i)} \\ \left(I_i'\right)^{-1}\left(\frac{W_i}{\delta \cdot (1 - \Phi_i)}\right) & \text{otherwise.} \end{cases} \quad (14)$$

33. Laszka, Zhao, and Grossklags, "Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms."

Then, hackers of type i will not spend any time on vulnerability discovery if $\hat{v}_i = 0$ and $b \cdot \Phi_i \cdot D_i'(0) \leq W_i + \delta \cdot (1 - \Phi_i) \cdot D_i'(0)$ or if $\hat{v}_i > 0$ and $b \cdot \Phi_i \cdot D_i'(0) \leq W_i \left(1 + D_i'(0) I_i'(0)\right)$. Otherwise, they will spend $\tilde{t}_i$ time on vulnerability discovery, where $\tilde{t}_i$ is the unique solution to $\frac{d}{dt_i}\mathcal{U}_{H_i} = 0$. In addition, they will spend $\hat{v}_i$ or $I_i^{-1}(D_i(t_i^*))$ time on validation, whichever one is lower.

3.3 Numerical Results

In this section, we present numerical results on our bug-bounty model in order to evaluate and compare the policies introduced in Section 3.2. First, in Section 3.3.1, we consider homogeneous hackers by instantiating our model with a single hacker type, and we study the hackers' responses. Second, in Section 3.3.2, we consider heterogeneous hackers and evaluate policies based on the organization's utility.

3.3.1 Homogeneous Hackers

For the vulnerability-discovery function $D(t)$, we use an instance of *Anderson's thermodynamic model*:[34] $D(t) = \ln(10 \cdot t + 1)$. 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 3 shows the hackers' responses to various policies and the resulting utilities for the organization and the hackers. First, Figure 3(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 Figures 3(b), 3(c), and 3(d), we set the bounty value to $b = 2.07$ and study the effects of varying the policy parameters.

Figure 3(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 (Figure 3(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, Figure 3(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.

34. Anderson, "Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore."

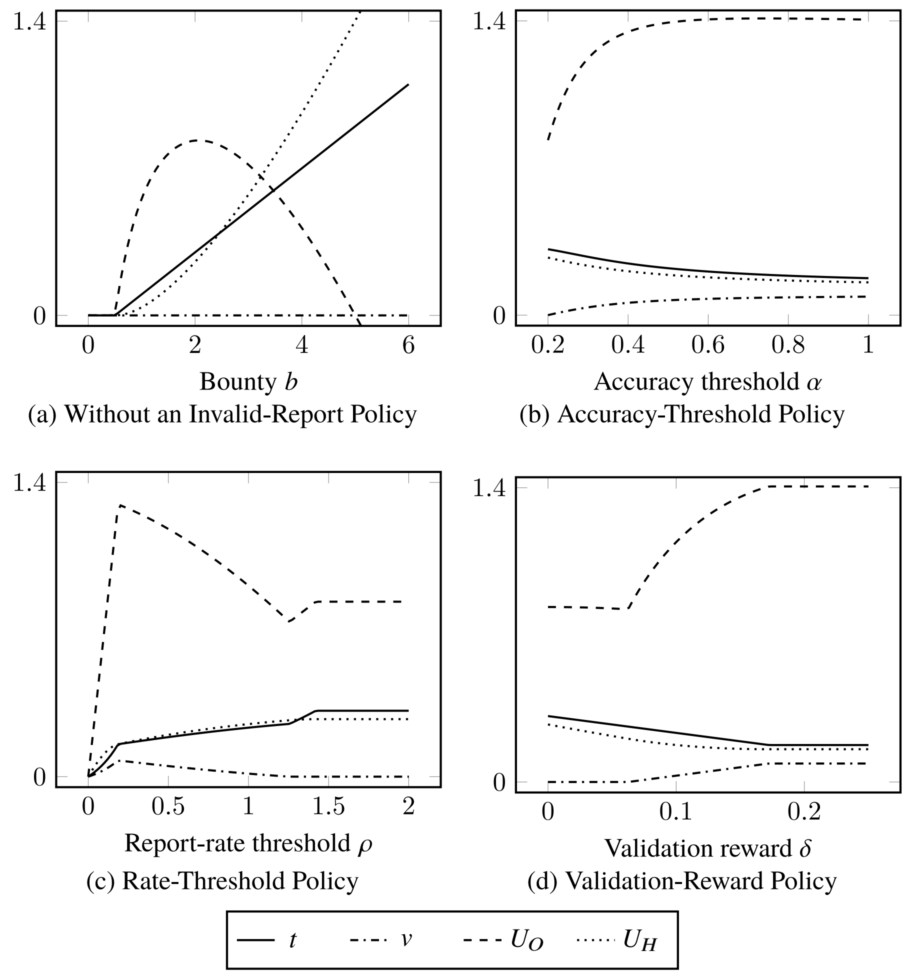
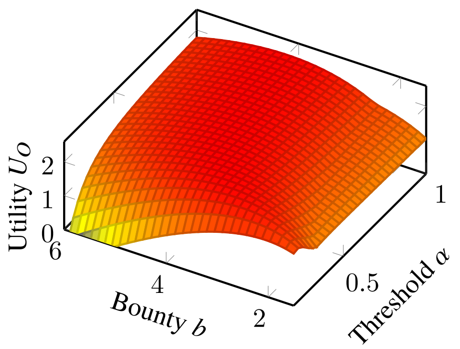


Figure 3: The organization's and the hackers' utilities (dashed and dotted lines) and the times spent on vulnerability discovery and validation (solid and dash-dotted lines) under various policies as functions of the bounty value.

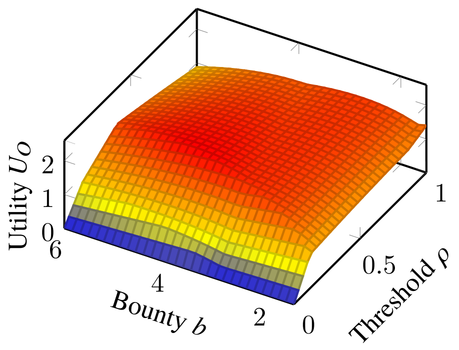
3.3.2 Heterogeneous Hackers

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.

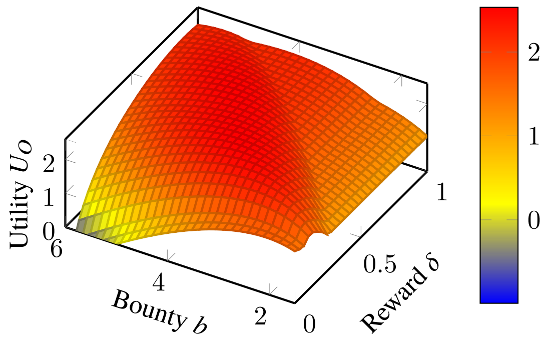
Figure 4 shows the organization's utility under various policies with two types of hackers. Similar to Figure 3(c), Figure 4(b) shows that the rate-threshold policy must be implemented carefully since overzealous limiting may significantly decrease



(a) Accuracy-Threshold Policy



(b) Rate-Threshold Policy



(c) Validation-Reward Policy

Figure 4: The organization's utility under various policies as a function of the bounty value and policy parameter.

the organization's utility, while lenient limiting is ineffective. On the other hand, the accuracy-threshold and validation-reward policies (Figures 4(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 5 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 Figure 5 are proportional to the values shown by Figure 3(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 significantly, since it is able to incentivize heterogeneous hackers to

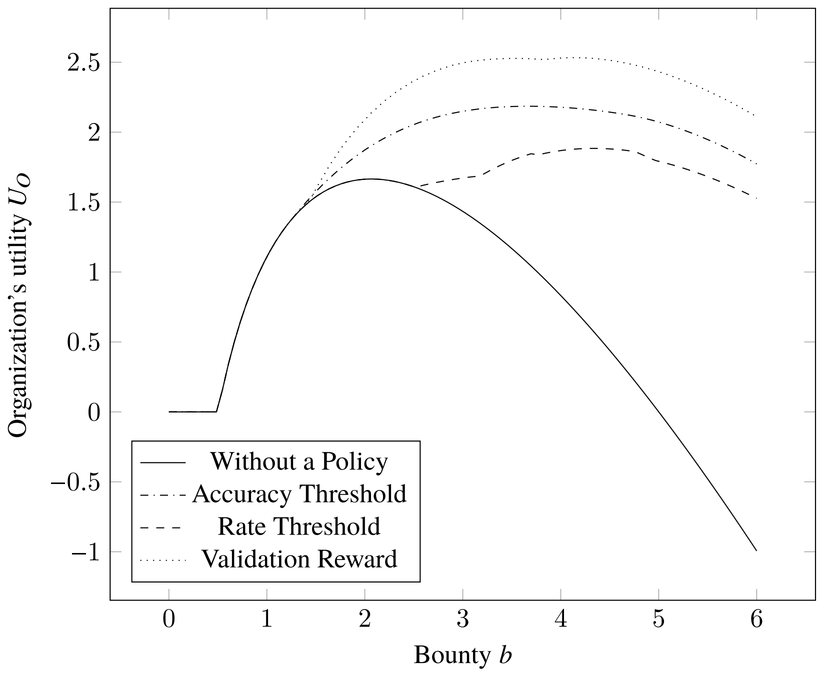


Figure 5: The organization's maximum attainable utility under various policies as a function of the bounty value

.

operate at their individual maxima instead of forcing them towards a uniform strategy.

3.4 Summary

In this section, 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.[35] 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.

35. HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*.

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 this 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.

4 Policies for Allocating Discovery Effort

In this section, we study policies for allocating hackers to bug-bounty programs, focusing on maximizing the number of unique reports and minimizing the number of duplicate reports. We first introduce a formal model of bug-bounty programs, which captures the non-deterministic vulnerability-discovery process and the diversity of the hackers' expertise. Based on our framework, we formalize two canonical policies, which model public and private bug-bounty programs in practice. Then, we study optimal allocations, which maximize the organization's utility while ensuring that hackers' have incentives to participate. Finally, we present a numerical illustration based on our framework, which shows how utilities are affected by the number of participating hackers.

This section is organized as follows. In Section 4.1, we introduce our economic model of unique and duplicate vulnerability reports in bug-bounty programs. In Section 4.2, we discuss a set of canonical policies for allocating hackers to a bug-bounty program. In Section 4.3, we present a numerical illustration and discuss our findings. Finally, in Section 4.4, we offer concluding remarks on this problem area and outline future work.

4.1 Model

Now, we introduce our economic model that we use to study the allocation of hacker in bug-bounty programs. A list of symbols used throughout Section 4 can be found in Table 3.

We assume that time is discrete, and we number the time steps of our model $1, 2, \ldots$. At the beginning (i.e., at the start of time step 1), there are a large number of unknown bugs[36] waiting to be discovered in a software product (e.g., a website). The organization responsible for this product creates a bug bounty program. It then invites participants from a set of hackers H at the start of each time step $t = 1, 2, \ldots$. We assume that the set of hackers H whom may be invited to participate is constant. We define an *allocation plan* A as a vector $A = \{H_1, H_2, \ldots\}$, where $H_t \subseteq H$ is the set of hackers invited for time step t.

At the end of each time step t, invited hackers submit r_t bug reports in total to the program. Among these reports, some are duplicates because multiple hackers could

36. Usually, bug bounty programs only focus on security bugs, or vulnerabilities. For brevity, we will use the more general word "bug" interchangeably with "vulnerability."

Table 3: List of Symbols in Section 4

Symbol	Description
Constants and Functions	
V	average value of a valid report for the organization
C_O	average cost of processing a report for the organization
C_H	average cost of finding and reporting a bug for a hacker
B	average bounty paid for the first report of vulnerability
H	set of hackers
S	set of vulnerability types
P_i	discovery probability of vulnerability i
Q_s	probability that a vulnerability is of type $s \in S$
S_h	set of vulnerability types that hacker $h \in H$ can discover
Variables	
H_t	set of hackers working on the program in time step t
r_t	total number of reports in time step t
u_t	number of unique discoveries in time step t

find the same issue. However, the organization is interested only in unique discoveries, whose number is denoted by u_t, and obviously $u_t \leq r_t$. The numbers r_t and u_t are generated using the vulnerability-discovery model, which we will discuss shortly. The organization rewards each unique bug discovery with bounty B (e.g., average around \$424 in 2015[37]), and fixes all discovered bugs at the end of each time step. The organization also incurs cost C_O for processing each submitted report. Finally, we assume that the organization gains value V from fixing each unique bug.[38] However, the value of fixing a bug decreases over time since the later the bug is fixed, the more likely it is that malicious parties will find and exploit the bug before it is fixed. We use a time-discounting factor $\delta \in (0, 1)$ to model this temporal preference.

We can write the utility function of an organization from bug bounty as

$$\mathcal{U}_O = \sum_{t=1}^{\infty} \left((\delta^{t-1} V - B) u_t - C_o r_t \right). \tag{15}$$

Next, we assume that it costs C_H for a hacker to find and report a vulnerability. Note that C_H can be determined by dividing the total amount of effort spent by hackers with the number of vulnerabilities reported. In other words, the constant C_H is the unit cost of reporting a vulnerability, which includes the effort spent on finding and validating the vulnerability as well as the effort spent on writing and submitting the report. Then, the utility function of all invited hackers is

$$\mathcal{U}_H = \sum_{t=1}^{\infty} (B u_t - C_H r_t). \tag{16}$$

37. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."
38. We assume that all bugs have equal impact for now. This assumption will be relaxed in future work.

4.1.1 Vulnerability-Discovery Model

Now, we will introduce our bug discovery model, which generates the numbers r_t and u_t. Each bug can be represented as a single input (or a group of inputs) that triggers a specific error in the software or hardware system. Since the input space of any non-trivial system is prohibitively large, a hacker usually discovers bugs based on tools with randomization (e.g., fuzzing), heuristics, experiences, and luck. Based on these observations, we propose the following bug discovery model. We assume that each bug i is discovered by an invited hacker in a single time step with probability P_i, independently of the other hackers and other bugs, as long as bug i has not been discovered in an earlier time step. Probability P_i not only models the randomness of bug discovery, but also captures the difficulty of discovering a bug.

Previous research has shown that bugs have different discovery difficulty in practice.[39] More specifically, it was observed that if bugs are numbered $i = 1, 2, \ldots$, then their probabilities P_i follow a discrete power law distribution:[40]

$$P_i = \frac{i^{-\alpha}}{\zeta(\alpha)}, \tag{17}$$

where α is a scaling factor and ζ is the Riemann Zeta function. The factor α reflects the size of the system's attack surface and the security quality of the system. In practice, this factor can potentially be estimated from characteristics like codebase size, maturity of the security development life cycle, etc..[41]

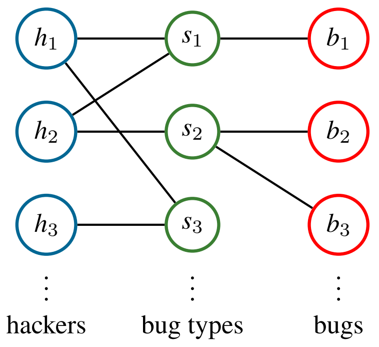


Figure 6: Illustration of the diversity model.

39. Brady, Anderson, and Ball, *Murphy's law, the fitness of evolving species, and the limits of software reliability*; Zhao and Liu, "Empirical Analysis and Modeling of Black-Box Mutational Fuzzing."

40. Zhao and Liu, "Empirical Analysis and Modeling of Black-Box Mutational Fuzzing."

41. Katie Moussouris, *A Maturity Model for Vulnerability Coordination*, HackerOne Blog, 2015; OWASP, *Software Assurance Maturity Model v1.1*, `https://www.owasp.org/index.php/OWASP_SAMM_Project`, 2016.

4.1.2 Hacker-Diversity Model

Existing literature has revealed that hackers have diverse expertise, use different tools, etc., so they are good at discovering different types of bugs.[42] Now, we introduce a model that captures this diversity.

First, each bug belongs to one type from a set of vulnerability types S.[43] We assume that the probability of a bug belonging to type s is Q_s, where Q_s is exogenous and known, and obviously $\sum_{s \in S} Q_s = 1$. The probability Q_s can be estimated from earlier bug discovery data, obtained through internal security testing or from similar organizations. Second, we let S_h be the set of vulnerability types that hacker $h \in H$ can discover. S_h can be obtained from data accumulated on bug bounty platforms. Figure 6 illustrates the relationships between the elements of our model.

By combining this with our discovery model, we have that the probability that hacker h discovers bug i is P_i if $s \in S_h$, where s is the type of bug i, and the probability is 0 if $s \notin S_h$.

4.2 Policies and Computational Complexity

Now, we study various policies for choosing an allocation plan. First, in Section 4.2.1, we formulate two canonical policies, which model public and private bug-bounty programs in practice, as well as formulate a policy that finds an optimal allocation. Then, in Section 4.2.2, we prove that finding an optimal allocation plan is a computationally hard problem.

4.2.1 Policies

We consider three policies for choosing the allocation plan $A = \{H_1, H_2, \ldots\}$:

- public program
- private program
- optimal time-variant allocation.

Public Program Policy A public program allows any hacker to participate and submit reports. Since the set of all hackers in our model is H, we can formulate this policy as

$$A = \{H, H, \ldots\},$$

that is, letting $H_t = H$ for every time step t. The advantage of this policy is that it minimizes the expected time until a bug is discovered and, hence, maximizes security. Unfortunately, it also maximizes the number of duplicate reports, which may result in overwhelming processing costs for the organization.

42. Anne Edmundson et al., "An empirical study on the effectiveness of security code review," in *Engineering Secure Software and Systems* (2013); Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems"; Munawar Hafiz and Ming Fang, "Game of detections: How are security vulnerabilities discovered in the wild?," *Empirical Software Engineering* 21, no. 5 (2016): 9021–1959; Keman Huang et al., "Poster: Diversity or Concentration? Hackers' Strategy for Working Across Multiple Bug Bounty Programs," in *37th IEEE Symposium on Security and Privacy (S&P)* (2016).

43. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."

Private Program Policy On the other hand, private programs allow only a selected group of hackers to participate and submit reports. We formulate a simple policy that models private programs as follows: first select a set of hackers H^*, and then invite them by letting

$$A = \{H^*, H^*, \ldots\},$$

that is, letting $H_t = H^*$ for every time step t. The set of invited hackers H^* is typically chosen to include the most skillful individuals. For example, we can select hackers who are capable of discovering at least σ different types of bugs:

$$H^* = \{h \in H \mid |S_h| \geq \sigma\}\,. \tag{18}$$

Compared to public programs, the advantage of this policy is that it decreases the number of duplicate reports by limiting the set of participating hackers. However, it also increases the expected time until a bug is discovered and, hence, increases the risk of a malicious party discovering and exploiting a bug before it could be fixed.

Optimal Time-Variant Allocation Finally, we consider the policy that always selects an optimal allocation plan, which maximizes the organization's expected utility. In Section 4.2.2, we will show that finding an optimal allocation is computationally hard. Consequently, implementing this policy in practice is quite difficulty, and we may have to use heuristic approaches instead.

An optimal allocation can plan can be expressed simply as

$$\max_{A} \mathrm{E}[\mathcal{U}_O].$$

However, in order to make the bug-bounty program feasible, we also need to ensure that participating hackers actually have an incentive to spend time and effort on discovery. That is, we need to ensure that the hackers' expected utility is non-negative despite the fact that duplicates, which are due to competition between hackers, cannot be rewarded and that discovering bugs becomes harder over time. Assuming that the bounty B paid by the organization is fixed, we can formulate this problem as

$$\max_{A:\, \mathrm{E}[\mathcal{U}_H] \geq 0} \mathrm{E}[\mathcal{U}_O].$$

More generally, if we allow the organization to adjust the bounty B, we can formulate this problem as

$$\max_{A, B:\, \mathrm{E}[\mathcal{U}_H] \geq 0} \mathrm{E}[\mathcal{U}_O].$$

4.2.2 Computational Complexity of Finding an Optimal Allocation

Next, we study the computational aspects of optimal time-variant allocation. Finding an optimal allocation is a challenging problem because the number of possible allocations to choose from is an exponential function of the number of hackers, which means that considering all possible allocations is computationally infeasible in practice. Indeed, we prove that this problem is NP-hard, which implies that there exist no efficient (i.e., polynomial-time) algorithm for finding an optimal allocation as long as the conjecture $P \neq NP$ holds. We remark for the benefit of those unacquainted with algorithmic complexity theory that $P \neq NP$ is a very widely accepted conjecture; if this were not true, then there would be a polynomial-time algorithm for every NP-hard problem. Consequently, practical policies must be based on heuristic approaches for finding allocations.

For ease of presentation, we consider the problem of finding an optimal set of hackers for a private program. The hardness of finding an optimal allocation that may use different sets of hackers for each time step follows readily from the hardness of this problem. To prove computational complexity, we first formulate the problem as a decision problem.

Definition 4 (Hacker Allocation Problem)**.** Given an instance of our model and a threshold utility $\mathcal{U}_O^*$, determine if there exists a set of hackers H^* such that the expected utility $\mathcal{U}_O$ for allocation $A^* = \{H^*, H^*, \ldots\}$ is greater than or equal to $\mathcal{U}_O^*$.

Theorem 1. *The Hacker Allocation Problem is NP-hard.*

We prove computational complexity by reducing a well-known NP-hard problem, the Exact Cover Problem, to the Hacker Allocation Problem. The Exact Cover Problem is defined as follows.

Definition 5 (Exact Cover Problem)**.** Given a base set X and a family $\mathcal{F}$ of subsets of X (i.e., each $F \in \mathcal{F}$ is a subset of X), determine if there exists a subfamily C of $\mathcal{F}$ such that each element of X is contained by exactly one subset in C.

Proof sketch. (Theorem 1). Given an instance of the Exact Cover Problem (i.e., a base set X and a family $\mathcal{F}$ of subsets), we create an instance of the Hacker Allocation Problem as follows:

- Let the set of vulnerability types be $S = X$ and the set of hackers be $H = \mathcal{F}$.
- For each hacker $h \in H = \mathcal{F}$, let the set of vulnerability types that the hacker can discover be $S_h = h$ (i.e., the vulnerability types corresponding to the elements of X that are in the subset corresponding to h).
- For each vulnerability type $s \in S$, let $Q_s = \frac{1}{|S|}$.
- Let $V = 3$, $C_O = 1$, $C_H = 0$, $B = 1$, $\delta = 0.1$, and $P_i = 1$ for every i.

It is clear that the reduction can be performed in polynomial time. Further, it is also clear that the organization's expected utility attains its maximum when the set hackers H^* forms an exact cover of X. Hence, by setting the threshold utility to this maximum, we can show that the Hacker Allocation Problem has a solution *iff* the Exact Cover Problem does. □

4.3 Numerical Illustration

The practical goal of our work is to help organizations optimize their bug bounty programs. A basic, but often voiced idea is to attract as many hackers as possible,

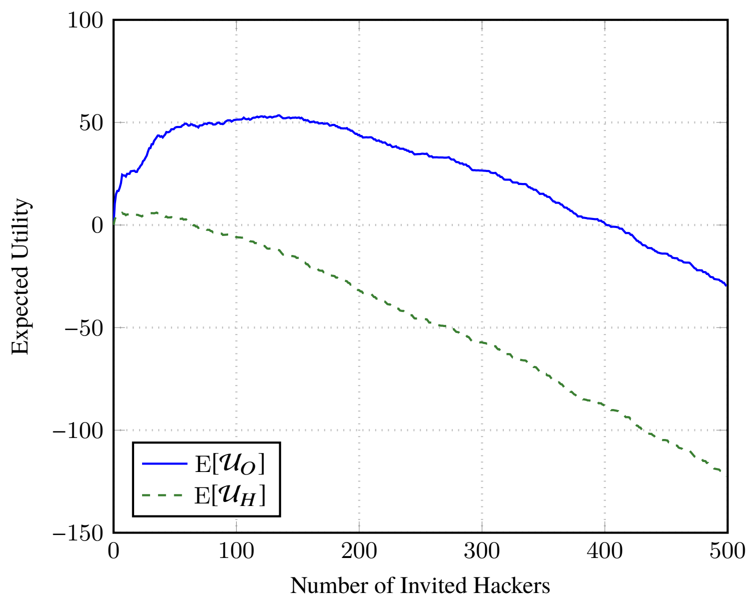


Figure 7: Expected utilities with different numbers of hackers invited. Parameters used: $V = 20$, $B = 5$, $\delta = 0.99$, $C_o = C_h = 1$, $\alpha = 2$.

which corresponds to a public program policy. We evaluate this notion using our model and data collected from a bug bounty platform, Wooyun.[44] Figure 7 shows that the expected utilities of the inviting organization and the invited hackers exhibit inverted U-shapes, and do not scale linearly with the number of hackers. Rather, they start to decrease after a certain number of hackers have joined. The reason is that as more hackers are invited, the number of duplicates increases, which raises the cost of processing reports by the organization, and also decreases the expected bounty received by hackers. This result suggests that for bug bounty programs, *more participation is not always better*. Instead, the bug bounty program should carefully design its allocation plan to control the competition among participants and to diversify its workforce.

4.4 Summary

In this section, we introduced the first theoretical framework for modeling the allocation of hackers to bug-bounty programs, focusing on the number of unique and duplicate reports. To capture how hackers discover vulnerabilities with their diverse set of expertise, we built our framework on a detailed vulnerability-discovery model and a hacker-diversity model. Our vulnerability discovery model is based on our prior experimental work,[45] which showed that discovery can be modeled as a relatively basic stochastic process.

44. Zhao, Grossklags, and Liu, “An empirical study of web vulnerability discovery ecosystems.”
45. Zhao and Liu, “Empirical Analysis and Modeling of Black-Box Mutational Fuzzing.”

Based on our framework, we first formalized two canonical policies, which modeled public and private bug-bounty programs. Then, we studied optimal time-variant hacker allocations, and showed that finding an optimal allocation is a computationally hard problem. The significance of this result is that practical allocation policies must be based on efficient heuristics instead of optimal allocations, since finding optimal allocations is computationally infeasible if the number of hackers is not very small. Finally, we presented numerical results on the organization's and the hackers' utilities in a typical bug-bounty program. Our results showed that the optimal number of invited hackers must be carefully chosen as increasing participation may decrease both the organization's and the hackers' utilities, which explains the growing popularity of private programs.

In future work, we plan to study more elaborate hacker-allocation policies, and devise effective heuristics algorithms for finding near-optimal allocations. In addition, we will carry out a more extensive numerical study comparing public and private programs, based on real-world data on unique and duplicate vulnerability reports.

5 Regulatory Policy Challenges

In the previous sections, we have discussed challenges for creating effective engagement policies between organizations and white-hats. In this section, we expand the scope of our discussion to regulatory policy challenges for bug-bounty approaches. We discuss how governments can create regulatory policies to better utilize bug-bounty to improve information security of the private sector and the public sector. We first focus on challenges related to creating regulatory policies for reducing risk and conflict associated with vulnerability research and disclosure. We then discuss whether governments should make bug-bounty mandatory for organizations. Finally, we discuss regulatory challenges that arise from potential conflict between bug-bounty and other policies related to cyber-security.

5.1 Protect Legitimate Vulnerability Research

To find security vulnerabilities that are eligible for bounties, white-hats have to conduct various security assessments against the target software system. However, there is a lack of clear regulatory policies on what vulnerability research activities are acceptable. The bug-bounty rules discussed in Section 2.1 can be considered as contracts between organizations and white-hats that permit limited vulnerability research. For example, Facebook's bug-bounty program explicitly states that "*If you comply with the policies below when reporting a security issue to Facebook, we will not initiate a lawsuit or law enforcement investigation against you in response to your report.*".[46] However, there will be times when white-hats and organizations disagree on the interpretation of those rules.[47]

46. Facebook, *Responsible Disclosure Policy*, `https://www.facebook.com/whitehat`, 2016.

47. Michael Mimoso, *Facebook, Researcher Spar over Instagram Vulnerabilities*, Threatpost; `https://threatpost.com/facebook-researcher-spar-over-instagram-vulnerabilities/115658/`, December 2015.

Having clear regulatory policies on legal vulnerability research can reduce risks and uncertainties to both organizations and white-hats, and encourages wider collaboration between them. Organizations would want to make sure that vulnerability research will not damage its systems' integrity and availability, nor harm its data confidentiality. While white-hats also want to be protected from (unreasonable) legal threats and penalties. However, the first challenge of creating such regulatory policies is that it is hard to separate vulnerability research from malicious hacking. After all, both of them typically include collecting information about the system, reverse-engineering a piece of software, sending exploit payload through different input vectors, attempting to escalate privilege to access critical data, etc. Loose regulations, therefore, might reduce the legal risk of malicious hacking and might encourage more cybercrimes.

In addition, it is also common that security researchers discover vulnerabilities in software products and systems without associated bug-bounty programs by organizations. Such efforts can lead to fixes of severe security flaws that impact a large user base. However, they could be considered as illegal cyber-attacks. In a recent case, a Chinese online dating company reported a white-hat to the police, who then arrested the white-hat based on the charge of illegal intrusion and acquisition of user data.[48] In addition to organization-level legal actions, governments might also take actions against bug-bounty organizers and white-hats. Wooyun, the oldest and largest bug-bounty platform in China, was abruptly shut down in Summer 2016, followed by the arrest of its key members.[49]

Moreover, a recent report published by the U.S. National Communications and Information Administration (NTIA) shows that less than one in five surveyed companies currently make use of bug-bounties.[50] The rest of the organizations could argue that not participating in bug-bounty approaches avoids risks associated with vulnerability research. But, at the same time, these organizations, who may be part of different industry sectors such as government, finance, education, could harbor a significant amount of vulnerabilities, which pose serious threats to the privacy, property or even safety of consumers and these organizations' interests. In fact, previous research has shown that white-hats are able to discover a wide range of severe vulnerabilities at organizations who have no formal bug-bounty programs.[51] One may postulate that most of these organizations, compared with organizations that have bug-bounty programs, are actually less secure, and are in more need of help from white-hats.

Therefore, we argue that future regulatory policy effort should aim to protect legitimate vulnerability research (see, for example, the recent temporary DMCA security research exemption for consumer devices[52]). Such exemptions shall specify what kind of vulnerability research is allowed (e.g., no user data will be exported), even if there is

48. Lin Zizhen and Saga McFarland, *Are China's 'Ethical Hackers' Cyber Heroes or Criminals?*, *CaixinOnline*, `http://english.caixin.com/2016-10-17/100997728.html`, October 2016.

49. Samuel Wade, *Internet Security Platform Closed; Founder Arrested*, China Digital Times; `http://chinadigitaltimes.net/2016/08/internet-security-platform-closed-founder-arrested/`, August 2016.

50. NTIA Awareness and Adoption Group, *Vulnerability Disclosure Attitudes and Actions*, `https://www.ntia.doc.gov/files/ntia/publications/2016_ntia_a_a_vulnerability_disclosure_insights_report.pdf`, 2016.

51. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."

52. Aaron Alva, *DMCA security research exemption for consumer devices*, Tech@FTC, 2016.

an absence of organization-wide bug-bounty rules.

5.2 Guidance for Vulnerability Disclosure

Another important question regarding bug-bounty and vulnerability research in general is how to manage disclosure. There are two main types of vulnerability disclosure approaches,[53] full disclosure and coordinated disclosure. *Full disclosure* is the policy of publishing the discovery as early as possible, so that the security community and affected users can immediately scrutinize the issue and take mitigation actions. However, it also enables malicious attackers to take advantage of the vulnerability information before mitigation measures are fully deployed. *Coordinated disclosure* (also known as responsible disclosure) requires that the white-hat shall first disclose the vulnerability only to the affected organization, and give the organization enough time to fix the flaw. Under coordinated disclosure, the risk of the vulnerability information being misused is reduced. However, if the vulnerability is also discovered by attackers, then all users of the product or system are still exposed to security risk, before the patch is available.

There has been a long debate in the security community regarding which disclosure approach is more favorable. In practice, such disagreements often lead to severe conflicts between white-hats and organizations. In a recent example, German firm ERNW discovered several serious vulnerabilities in the products of FireEye. However, FireEye and ERNW failed to reach an agreement on what can be shared with the public, and FireEye filed an injunction in German court to prevent ERNW from disclosing certain information for intellectual property protection reason.[54]

Regulatory policies for vulnerability disclosure that specify the best disclosure decision under different scenarios not only reduce conflicts, but also increase the positive impact of bug-bounty on improving cyber-security. The key challenge here is to balance the trade-off between the benefit and risk of vulnerability disclosure. Although, there has been plenty research on vulnerability disclosure in the past,[55] we think that more research is required to understand this trade-off, and to create actionable guidelines for vulnerability disclosure decisions. Future analyses could consider indirect benefits of vulnerability disclosure as well, such as the facilitation of more effective legitimate vulnerability research, and an increase of the security awareness of consumers.

5.3 Make Bug-bounty Mandatory

The benefits of bug-bounty lead to proposals to further widen the application of the approach. As a first step, regulatory agencies can include bug-bounty as one solution to

53. Hasan Cavusoglu, Huseyin Cavusoglu, and Srinivasan Raghunathan, "Emerging Issues in Responsible Vulnerability Disclosure," in *Workshop on the Economics of Information Security (WEIS)* (2005).

54. FireEye, *BUG BOUNTIES, (NON) LAWSUITS AND WORKING WITH THE RESEARCH COMMUNITY*, `https://security.googleblog.com/2013/05/disclosure-timeline-for-vulnerabilities.html`.

55. Hasan Cavusoglu and S Raghunathan, "Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge," *IEEE Transactions on Software Engineering* 33, no. 3 (2007); Ashish Arora, Rahul Telang, and Hao Xu, "Optimal policy for software vulnerability disclosure," *Management Science* 54, no. 4 (2008): 642–656; Ashish Arora et al., "An empirical analysis of software vendors' patch release behavior: impact of vulnerability disclosure," *Information Systems Research* 21, no. 1 (2010): 115–132.

satisfy certain security compliance requirements. For example, bug-bounty could serve as an alternative to penetration testing in the PCI DSS Requirement 11.3.[56] Academics and security researchers have also proposed to make bug-bounty programs mandatory. For example, Barnes advocates for this "low-cost" approach.[57] Likewise, Hahn and Layne-Farrar see mandatory bounty programs as a solution to software security problems since they "cost little to establish, only pay out when a benefit is achieved, and can be set at a level that guarantees that benefits exceed any costs".[58] Other researchers go one step further to discuss the feasibility of a national or international vulnerability purchase program to absorb zero-day vulnerabilities from white-hat and black hat researchers worldwide.[59] Regulatory agencies also appear to move in the direction of requiring bug-bounty programs. For example, the Federal Trade Commission issued a penalty in the HTC America case also in part since the company did not have a process for receiving and addressing security vulnerabilities reports.[60] In addition to receiving vulnerability reports, bug-bounty programs could also indirectly and positively influence cyber-security. For example, bug-bounty data such as bounty levels, vulnerability statistics, and participation trends could potentially alleviate the market for lemons problem for security,[61] and enable consumers to select more secure information services.

However, there are several challenges for creating such regulatory policies. First, due to the invalid report issue discussed in Section 2, the cost of establishing and managing a bug-bounty program might be higher than it appears to be initially. Second, previous research has shown that many organizations, particularly smaller ones, are not well-prepared to handle vulnerability reports.[62] Under delayed disclosure policy, their vulnerability reports could be exposed to the public, and thus significantly increase the risk of security attacks. Therefore, we believe that it is important to establish regulatory policies for vulnerability research and disclosure before pushing for a wider application of bug-bounty. Also, such policies can consider utilizing third-parties, such as HackerOne and CERT, to handle reports for the majority of organizations with small and inexperienced security teams.

Another challenge for fulfilling these visionary ideas is the limited manpower of white-hats. Please note that these factors are interrelated since high-quality white-hat contributions and work efforts are a scarce and valuable resource which needs to be efficiently allocated and distributed to a growing number of participating companies. To illustrate the current limits to growing white-hat contributions consider the recently

56. PCI Security Standards Council, *Information Supplement: Penetration Testing*, March 2008.

57. Douglas Barnes, "Deworming the internet," *Texas Law Review* 83, no. 1 (2004).

58. Hahn and Layne-Farrar, "The Law and Economics of Software Security."

59. Stefan Frei and Francisco Artes, *International Vulnerability Purchase Program: Why buying all vulnerabilities above black market prices is economically sound*, NSS Labs, Analyst Brief, 2013; Stephen M. Maurer, *A market-based approach to cyber defense: Buying zero-day vulnerabilities*, Bulletin of the Atomic Scientists, 2017.

60. Federal Trade Commission, *Start with Security: A Guide for Business*, FTC Website; `https://www.ftc.gov/tips-advice/business-center/guidance/start-security-guide-business#current`, June 2015.

61. Ross Anderson, "Why information security is hard-an economic perspective," in *Computer security applications conference, 2001. acsac 2001. proceedings 17th annual* (IEEE, 2001), 358–365.

62. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."

released results from HackerOne's Bug Bounty Hacker Report.[63] The report indicates that researchers come from all walks of life (e.g., 39% work for a security company), and 70% of the surveyed population are either full-time employees or full-time students which limits the time which can be expended for vulnerability discovery. Further, only 3.4% reported to have learned the necessary skills from classes or certificate programs, while all others were either self-taught hackers or benefited from informal learning within their own social networks of colleagues or friends. These findings suggest a need to find new approaches to scale the available workforce to accommodate a much larger number of programs. For example, rather than merely waiting for more security professionals to join, bug-bounty platforms or other organizations can create online education and crowdsourcing programs to teach less experienced or new participants the knowledge and skills to do ethical security research. Such efforts will lower the bar of participation and enlarge the workforce.

Conceivably, *substantially* higher bounty rewards as, for example, advocated by Frei and Artes may increase the supply of labor to bounty platforms and potentially redirect some labor (and trading of vulnerabilities) from cybercriminal marketplaces;[64] otherwise, it is likely that we would merely observe a redistributive effect of a relatively constant overall work effort to higher paying programs. Our allocation model in Section 4 can be viewed as a first step to solve this challenge.

5.4 Bug-bounty and Other Cyber-security Policies

There are also challenges to reconcile bug-bounty with other cyber-security policies. At present, the white-hat community is very international. For example, HackerOne, the leading bug-bounty platform in U.S., reported that only 19% of the hackers are self-identified as U.S.-based, and hackers come from more than 70 countries[65] (please also note data from Facebook and Bugcrowd[66]). Therefore, for bug-bounty platforms to run smoothly, a large share of vulnerability reports have to flow across international borders. In contrast, a broad coalition of countries (including Western nations and former Warsaw Pact countries) are regulating the export of intrusion software through the Wassenaar Arrangement. The relevant terms have been introduced in the text of the arrangement in 2013, but are undergoing incremental changes with each revision of the arrangement. While the intended goal of the provisions is to prevent repressive regimes from exploiting such software, the security research community considers the definition of intrusion software as overly broad[67] and remains concerned about negative (chilling) effects on the vulnerability discovery ecosystem, which are difficult to observe. Further, the Wassenaar Arrangement may expand to include other nations known for contributions to bug-bounty platforms such as India, which would further sharpen these

63. HackerOne, *The 2016 Bug Bounty Hacker Report*, *HackerOne Blog*, `https://hackerone.com/blog/bug-bounty-hacker-report-2016`, 2016.

64. Frei and Artes, *International Vulnerability Purchase Program: Why buying all vulnerabilities above black market prices is economically sound*.

65. HackerOne, *The 2016 Bug Bounty Hacker Report*.

66. Facebook, *Bug Bounty Highlights and Updates*, 2014; Bugcrowd, *The State of Bug Bounty*, July 2015.

67. Eric Chabrow, *US-Backed Effort to Ease Software Export Limits Fails*, `http://www.govinfosecurity.com/us-backed-effort-to-ease-software-export-limits-fails-a-9598`; Katie Moussouris, *You Need to Speak Up For Internet Security. Right Now.*, Wired, 2015.

concerns. In addition, such arrangements may have further unintended consequences including the misuse of vulnerability information. For example, if white-hats are (or feel) subjected to export control procedures and request an export license from their government, then a self-interested government might reject the request, and secretly put the report in its cyberweapons arsenal.

Another angle is related to government purchasing and stockpiling of zero-day vulnerabilities for offensive purposes. Government agencies or their brokers often pay much higher rewards compared to bug-bounty programs. For example, Apple's recently launched bug-bounty program is expected to pay up to $200,000,[68] while Zerodium, a vulnerability broker company whose clients are said to include government agencies, offers up to $1.5M for top iOS vulnerabilities.[69] Such programs, however, are in conflict with the goal of bug-bounty because they not only prevent issues from being disclosed and fixed, but might also drain manpower away from bug-bounty programs.

5.5 Summary

At present, bug-bounty, or vulnerability research and disclosure in general, is not regulated by clear policies. Such uncertainty hinders a wider collaboration between organizations and white-hats to enhance system security and protect user data. We argue that it is necessary to spend more effort on researching and constructing regulatory policies for vulnerability research and disclosure. We also think there are benefits to enforce certain forms of mandatory bug-bounty, yet more research effort is needed to understand their impact, and to build more efficient mechanisms to engage white-hats and organizations. Finally, we think that policy makers of new cyber-security initiatives shall consider the potential impact on bug-bounty.

6 Related Work

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.[70] 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.[71] Ozment further extended Schechter's testing competition into a vulnerability auction to improve its efficiency and better defend against attacks.[72] In

68. Kate Conger, *Apple announces long-awaited bug bounty program*, *Tech Crunch*, `https://techcrunch.com/2016/08/04/apple-announces-long-awaited-bug-bounty-program/`, August 2016.

69. LILY HAY NEWMAN, *A Top-Shelf iPhone Hack Now Goes for $1.5 Million*, *Wired*, `https://www.wired.com/2016/09/top-shelf-iphone-hack-now-goes-1-5-million/`, September 2016.

70. Rainer Böhme, "A Comparison of Market Approaches to Software Vulnerability Disclosure," in *Emerging Trends in Information and Communication Security*, ed. Günter Müller (Springer Berlin Heidelberg, 2006), 298–311.

71. Stuart Schechter, "How to Buy Better Testing Using Competition to Get the Most Security and Robustness for Your Dollar," in *Infrastructure Security* (Springer, 2002), 73–87.

72. Andy Ozment, "Bug auctions: Vulnerability markets reconsidered," in *Workshop on the Economics of Information Security (WEIS)* (2004).

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 account for 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.[73] 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,[74] and suggested that VRPs are more cost-effective compared to hiring full-time security researchers in terms of finding security flaws. Kuehn and Mueller applied institutional economics theory and document analysis to explain the formation of bug bounty programs.[75]

Zhao et al. conducted a comprehensive study of two bug bounty ecosystems, Wooyun and HackerOne, to understand their characteristics, trajectories and impact.[76] They quantitatively discussed the low signal-to-noise ratio problem which is one focus of this paper. In follow-up research, they 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.[77] Both papers 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.

Compared with these empirical studies, this paper applies theoretical modeling to study the dynamics of bug bounty, and proposes new mechanisms to improve its efficiency and effectiveness.

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.[78] 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

73. Schechter, "How to Buy Better Testing Using Competition to Get the Most Security and Robustness for Your Dollar."

74. Finifter, Akhawe, and Wagner, "An empirical study of vulnerability rewards programs."

75. Andreas Kuehn and Milton Mueller, *Analyzing bug bounty programs: An institutional perspective on the economics of software vulnerabilities*, SSRN, 2014.

76. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems."

77. Maillart et al., "Given Enough Eyeballs, All Bugs are Shallow? Revisiting Eric Raymond with Bug Bounty Markets."

78. Karthik Kannan and Rahul Telang, "Market for software vulnerabilities? Think again," *Management Science* 51, no. 5 (2005): 726–740.

publicly disclosed vulnerabilities to examine the risk and impact of such an ecosystem.[79] 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.[80] 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.[81] Bacon et al. have proposed a more general market design that contains bug hunters, developers, and users.[82] Finally, Libicki et al. conducted a comprehensive study of vulnerability markets and their relevance to cyber security and public policy.[83]

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.[84] 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[85] and uncovered the diversity of white hat behaviors regarding productivity, vulnerability type specialization, and discovery transitions. Likewise, Huang et al. reported that hackers at various levels of experience exist in the vulnerability disclosure ecosystem.[86] They found that hackers with different levels of accuracy have diverse strategies in selecting to which programs to contribute.

Our discussion of duplicated reports in Sections 2.3 and 4 also relates to existing studies of vulnerability rediscovery. Ozment investigated vulnerability data of OpenBSD 2.2, and found that vulnerabilities are often independently rediscovered within a relatively short time span.[87] Herr and Schneier, based on data of different vendors and software types, estimated that the rediscovery rate is roughly 20%[88]. Ablon

79. Stefan Frei et al., "Modeling the security ecosystem - The dynamics of (in)security," in *Economics of Information Security and Privacy* (2009).

80. Sam Ransbotham, Sabyaschi Mitra, and Jon Ramsey, "Are markets for vulnerabilities effective?," *MIS Quarterly* 36, no. 1 (2012): 43–64.

81. Abdullah Algarni and Yashwant Malaiya, "Software Vulnerability Markets: Discoverers and Buyers," *International Journal of Computer, Information Science and Engineering* 8, no. 3 (2014): 71–81.

82. David Bacon et al., "A market-based approach to software evolution," in *24th ACM SIGPLAN Conference Companion on Object Oriented Programming, Systems, Languages, and Applications* (2009).

83. Martin Libicki, Lillian Ablon, and Tim Webb, *The defender's dilemma: Charting a course toward cybersecurity* (Rand Corporation, 2015).

84. Edmundson et al., "An empirical study on the effectiveness of security code review."

85. Zhao, Grossklags, and Chen, "An Exploratory Study of White Hat Behaviors in a Web Vulnerability Disclosure Program."

86. Cheng Huang et al., "A study on Web security incidents in China by analyzing vulnerability disclosure platforms," *Computers & Security* 58 (2016): 47–62.

87. Andy Ozment, "The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting," in *Workshop on the Economics of Information Security (WEIS)* (2005).

88. Trey Herr and Bruce Schneier, *Taking Stock: Estimating Vulnerability Rediscovery*, SSRN, 2017.

and Bogart further looked at zero-day vulnerabilities, and found that approximately 5.7% of a stockpile of zero-day vulnerabilities will be discovered by outside parties in one year.[89] The main goals of these existing work are either to understand whether vulnerability hunting is beneficial to security (e.g., by depleting zero-days in the hands of attackers), or to help government decide when a zero-day vulnerability shall be disclosed, rather than keeping it in the stockpile for potential offensive usage. Our research focuses on a different bug hunting scenario, with the goal of making the crowdsourcing effort more efficient.

While there is a growing literature on the empirical, theoretical and policy/legal study of software security and vulnerability discovery, we are unaware of any literature that targets improvements to the effectiveness of bug-bounty platforms by devising economic policies to reduce the occurrence of duplicate submissions and reports with various forms of invalid submissions.

Our work is also related to recent research effort on allocating workers to crowdsourcing tasks. However, their results cannot be directly applied to bug bounty, for mainly three reasons. First, even though people consider bug bounty as a kind of crowdsourcing, the vulnerability discovery task is actually much more difficult, and requires more expertise and diversity compared to simple tasks, such as image labeling discussed in the crowdsourcing literature.[90] Therefore, existing crowdsourcing allocation mechanisms cannot be applied directly to bug bounty. Second, crowdsourcing mechanisms typically try to address the issue of unreliable workers by allocating multiple workers to the same task and infer the correct result.[91] This is because the employer cannot directly verify workers' output. However, bug bounty operators can validate submissions. Third, literature of allocation in crowdsourcing typically considers the utility of employer only, while our model in Section 4 needs to consider the utility of both organizations and hackers.

7 Concluding Remarks

In our study, we consider software developers and businesses who manage their own bug bounty programs (e.g., Google or Facebook) or voluntarily join one of several commercial bug-bounty platforms who run programs for them (e.g., HackerOne, BugCrowd, Cobalt). Across these different programs and platforms, we can observe various positive success metrics. For example, on HackerOne, almost 30,000 security vulnerabilities have been reported and fixed for hundreds of organizations. The other platforms report similar data indicating their overall growth and attraction (see, for example, Bugcrowd's The State of Bug Bounty reports[92]). Previous research also provided empirical evidence

89. Lillian Ablon and Andy Bogart, *Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits* (Rand Corporation, 2017).

90. Weici Hu and Peter Frazier, "Bayes-Optimal Effort Allocation in Crowdsourcing: Bounds and Index Policies," in *Artificial Intelligence and Statistics* (2016), 324–332.

91. David R Karger, Sewoong Oh, and Devavrat Shah, "Budget-optimal crowdsourcing using low-rank matrix approximations," in *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on* (IEEE, 2011), 284–291; Xi Chen, Qihang Lin, and Dengyong Zhou, "Optimistic Knowledge Gradient Policy for Optimal Budget Allocation in Crowdsourcing.," in *ICML (3)* (2013), 64–72.

92. Bugcrowd, *The State of Bug Bounty*; Bugcrowd, *The State of Bug Bounty*.

showing that white hats' scrutiny makes the finding of new vulnerabilities increasingly difficult.[93]

In light of these observations, it is imperative to more effectively organize current bug bounty programs. We contribute to finding solutions for this problem space by suggesting and evaluating engagement policies that would address two prevalent problems. First, we develop a model for evaluating approaches for reducing the number of invalid reports. We anticipate that the deployment of the evaluated policies will have a significant impact on the viability of a broad range of bug bounty programs and platforms. Initial positive results achieved by similar policies such as the "Signal Requirements" and "Rate Limiter" mechanisms on HackerOne, give confidence of their practical applicability.[94] A second contribution of this paper is the introduction of an economic model to study the allocation of hackers in bug-bounty programs.

It is also necessary to have regulatory policies for bug-bounty regarding vulnerability research and disclosure. We have discussed multiple challenges for creating such regulatory policies, and pointed out future research directions.

Acknowledgements

We thank the anonymous reviewers and editorial staff for their comments, which greatly improved the article. We also thanks Thomas Maillart for his comments on an early version of the paper. We are grateful for feedback received at talks at the Technical University of Munich, the CROSSING conference at the Technical University of Darmstadt, the 2016 PrivacyCon Conference facilitated by the Federal Trade Commission, the 2016 Legal and Policy Dimensions of Cybersecurity workshop, and the 2017 CRESSE Conference on the Advances in the Analysis of Competition Policy and Regulation. The research activities of Jens Grossklags are supported by the German Institute for Trust and Safety on the Internet (DIVSI). Aron Laszka's work was supported in part by the National Science Foundation (CNS-1238959) and the Air Force Research Laboratory (FA 8750-14-2-0180).

References

Ablon, Lillian, and Andy Bogart. *Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits*. Rand Corporation, 2017.

Algarni, Abdullah, and Yashwant Malaiya. "Software Vulnerability Markets: Discoverers and Buyers." *International Journal of Computer, Information Science and Engineering* 8, no. 3 (2014): 71–81.

Alhazmi, Omar, and Yashwant Malaiya. "Modeling the vulnerability discovery process." In *16th IEEE International Symposium on Software Reliability Engineering (ISSRE)*. 2005.

93. Zhao, Grossklags, and Liu, "An empirical study of web vulnerability discovery ecosystems"; Maillart et al., "Given Enough Eyeballs, All Bugs are Shallow? Revisiting Eric Raymond with Bug Bounty Markets."
94. HackerOne, *Improving Public Bug Bounty Programs with Signal Requirements*.

Alva, Aaron. *DMCA security research exemption for consumer devices*. Tech@FTC, 2016.

Anderson, Ross. “Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore.” In *Proceedings of Open Source Software: Economics, Law and Policy*. 2002.

———. “Why information security is hard-an economic perspective.” In *Computer security applications conference, 2001. acsac 2001. proceedings 17th annual*, 358–365. IEEE, 2001.

Arora, Ashish, Ramayya Krishnan, Rahul Telang, and Yubao Yang. “An empirical analysis of software vendors’ patch release behavior: impact of vulnerability disclosure.” *Information Systems Research* 21, no. 1 (2010): 115–132.

Arora, Ashish, Rahul Telang, and Hao Xu. “Optimal policy for software vulnerability disclosure.” *Management Science* 54, no. 4 (2008): 642–656.

Awareness, NTIA, and Adoption Group. *Vulnerability Disclosure Attitudes and Actions*. `https://www.ntia.doc.gov/files/ntia/publications/2016_ntia_a_a_vulnerability_disclosure_insights_report.pdf`, 2016.

Bacon, David, Yiling Chen, David Parkes, and Malvika Rao. “A market-based approach to software evolution.” In *24th ACM SIGPLAN Conference Companion on Object Oriented Programming, Systems, Languages, and Applications*. 2009.

Barnes, Douglas. “Deworming the internet.” *Texas Law Review* 83, no. 1 (2004).

Böhme, Rainer. “A Comparison of Market Approaches to Software Vulnerability Disclosure.” In *Emerging Trends in Information and Communication Security*, edited by Günter Müller, 298–311. Springer Berlin Heidelberg, 2006.

Brady, Robert, Ross Anderson, and Robin Ball. *Murphy’s law, the fitness of evolving species, and the limits of software reliability*. Technical report 471. University of Cambridge, Computer Laboratory, 1999.

Bugcrowd. *The State of Bug Bounty*, July 2015.

———. *The State of Bug Bounty*, June 2016.

Cavusoglu, Hasan, Huseyin Cavusoglu, and Srinivasan Raghunathan. “Emerging Issues in Responsible Vulnerability Disclosure.” In *Workshop on the Economics of Information Security (WEIS)*. 2005.

Cavusoglu, Hasan, and S Raghunathan. “Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge.” *IEEE Transactions on Software Engineering* 33, no. 3 (2007).

Chabrow, Eric. *US-Backed Effort to Ease Software Export Limits Fails*. `http://www.govinfosecurity.com/us-backed-effort-to-ease-software-export-limits-fails-a-9598`.

Chen, Xi, Qihang Lin, and Dengyong Zhou. "Optimistic Knowledge Gradient Policy for Optimal Budget Allocation in Crowdsourcing." In *ICML (3)*, 64–72. 2013.

Collins, Shannon. *DoD Announces 'Hack the Pentagon' Follow-Up Initiative. DoD News, Defense Media Activity*, `https://www.defense.gov/News/Article/Article/981160/dod-announces-hack-the-pentagon-follow-up-initiative`, October 2016.

Conger, Kate. *Apple announces long-awaited bug bounty program. Tech Crunch*, `https://techcrunch.com/2016/08/04/apple-announces-long-awaited-bug-bounty-program/`, August 2016.

Council, PCI Security Standards. *Information Supplement: Penetration Testing*, March 2008.

Doupé, Adam, Marco Cova, and Giovanni Vigna. "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners." In *Detection of Intrusions and Malware, and Vulnerability Assessment*. 2010.

Edmundson, Anne, Brian Holtkamp, Emanuel Rivera, Matthew Finifter, Adrian Mettler, and David Wagner. "An empirical study on the effectiveness of security code review." In *Engineering Secure Software and Systems*. 2013.

Facebook. *2015 Highlights: Less Low-Hanging Fruit.* `https://www.facebook.com/notes/facebook-bug-bounty/2015-highlights-less-low-hanging-fruit/1225168744164016`, 2016.

———. *Bug Bounty Highlights and Updates*, 2014.

———. *Responsible Disclosure Policy*. `https://www.facebook.com/whitehat`, 2016.

Federal Trade Commission. *Start with Security: A Guide for Business*. FTC Website; `https://www.ftc.gov/tips-advice/business-center/guidance/start-security-guide-business#current`, June 2015.

Finifter, Matthew, Devdatta Akhawe, and David Wagner. "An empirical study of vulnerability rewards programs." In *USENIX Security Symposium*. 2013.

FireEye. *BUG BOUNTIES, (NON) LAWSUITS AND WORKING WITH THE RESEARCH COMMUNITY*. `https://security.googleblog.com/2013/05/disclosure-timeline-for-vulnerabilities.html`.

Frei, Stefan, and Francisco Artes. *International Vulnerability Purchase Program: Why buying all vulnerabilities above black market prices is economically sound*. NSS Labs, Analyst Brief, 2013.

Frei, Stefan, Dominik Schatzmann, Bernhard Plattner, and Brian Trammell. "Modeling the security ecosystem - The dynamics of (in)security." In *Economics of Information Security and Privacy*. 2009.

Google. *Statistics and Charts of Google VRP*. `https://sites.google.com/site/bughunteruniversity/behind-the-scenes/charts/2014`, 2014.

HackerOne. *Expanding Reputation: Introducing Signal and Impact*. *HackerOne Blog*, `https://hackerone.com/blog/introducing-signal-and-impact`, December 2015.

———. *Fair and Transparent Hacker Invitations*. *HackerOne Blog*, `https://hackerone.com/blog/fair-and-transparent-hacker-invitations`, March 2016.

———. *Improving Public Bug Bounty Programs with Signal Requirements*. *HackerOne Blog*, `https://hackerone.com/blog/signal-requirements`, March 2016.

———. *Improving Signal Over 10,000 Bugs*. *HackerOne Blog*, `https://hackerone.com/blog/improving-signal-over-10000-bugs`, July 2015.

———. *The 2016 Bug Bounty Hacker Report*. *HackerOne Blog*, `https://hackerone.com/blog/bug-bounty-hacker-report-2016`, 2016.

———. *The HackerOne Success Index - Response Efficiency*. *HackerOne Blog*, `https://hackerone.com/blog/response-efficiency`, February 2016.

Hafiz, Munawar, and Ming Fang. "Game of detections: How are security vulnerabilities discovered in the wild?" *Empirical Software Engineering* 21, no. 5 (2016): 9021–1959.

Hahn, Robert, and Anne Layne-Farrar. "The Law and Economics of Software Security." *Harvard Journal of Law & Public Policy* 30, no. 1 (2006): 283–353.

Herr, Trey, and Bruce Schneier. *Taking Stock: Estimating Vulnerability Rediscovery*. SSRN, 2017.

Hu, Weici, and Peter Frazier. "Bayes-Optimal Effort Allocation in Crowdsourcing: Bounds and Index Policies." In *Artificial Intelligence and Statistics*, 324–332. 2016.

Huang, Cheng, JiaYong Liu, Yong Fang, and Zheng Zuo. "A study on Web security incidents in China by analyzing vulnerability disclosure platforms." *Computers & Security* 58 (2016): 47–62.

Huang, Keman, Michael Siegel, Stuart Madnick, Xiaohong Li, and Zhiyong Feng. "Poster: Diversity or Concentration? Hackers' Strategy for Working Across Multiple Bug Bounty Programs." In *37th IEEE Symposium on Security and Privacy (S&P)*. 2016.

Kannan, Karthik, and Rahul Telang. "Market for software vulnerabilities? Think again." *Management Science* 51, no. 5 (2005): 726–740.

Karger, David R, Sewoong Oh, and Devavrat Shah. "Budget-optimal crowdsourcing using low-rank matrix approximations." In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, 284–291. IEEE, 2011.

Kuehn, Andreas, and Milton Mueller. *Analyzing bug bounty programs: An institutional perspective on the economics of software vulnerabilities*. SSRN, 2014.

Laszka, Aron, Mingyi Zhao, and Jens Grossklags. “Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms.” In *21st European Symposium on Research in Computer Security (ESORICS)*, 161–178. 2016.

Libicki, Martin, Lillian Ablon, and Tim Webb. *The defender’s dilemma: Charting a course toward cybersecurity*. Rand Corporation, 2015.

Maillart, Thomas, Mingyi Zhao, Jens Grossklags, and John Chuang. “Given Enough Eyeballs, All Bugs are Shallow? Revisiting Eric Raymond with Bug Bounty Markets.” In *Workshop on the Economics of Information Security (WEIS)*. 2016.

Maurer, Stephen M. *A market-based approach to cyber defense: Buying zero-day vulnerabilities*. Bulletin of the Atomic Scientists, 2017.

Mimoso, Michael. *Facebook, Researcher Spar over Instagram Vulnerabilities*. Threatpost; `https://threatpost.com/facebook-researcher-spar-over-instagram-vulnerabilities/115658/`, December 2015.

Moussouris, Katie. *A Maturity Model for Vulnerability Coordination*. HackerOne Blog, 2015.

———. *You Need to Speak Up For Internet Security. Right Now.* Wired, 2015.

NEWMAN, LILY HAY. *A Top-Shelf iPhone Hack Now Goes for $1.5 Million*. *Wired*, `https://www.wired.com/2016/09/top-shelf-iphone-hack-now-goes-1-5-million/`, September 2016.

OWASP. *Software Assurance Maturity Model v1.1*. `https://www.owasp.org/index.php/OWASP_SAMM_Project`, 2016.

Ozment, Andy. “Bug auctions: Vulnerability markets reconsidered.” In *Workshop on the Economics of Information Security (WEIS)*. 2004.

———. “The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting.” In *Workshop on the Economics of Information Security (WEIS)*. 2005.

Ransbotham, Sam, Sabyaschi Mitra, and Jon Ramsey. “Are markets for vulnerabilities effective?” *MIS Quarterly* 36, no. 1 (2012): 43–64.

Schechter, Stuart. “How to Buy Better Testing Using Competition to Get the Most Security and Robustness for Your Dollar.” In *Infrastructure Security*, 73–87. Springer, 2002.

Wade, Samuel. *Internet Security Platform Closed; Founder Arrested*. China Digital Times; `http://chinadigitaltimes.net/2016/08/internet-security-platform-closed-founder-arrested/`, August 2016.

Zhao, Mingyi, Jens Grossklags, and Kai Chen. “An Exploratory Study of White Hat Behaviors in a Web Vulnerability Disclosure Program.” In *2014 ACM CCS Workshop on Security Information Workers*. 2014.

Zhao, Mingyi, Jens Grossklags, and Peng Liu. “An empirical study of web vulnerability discovery ecosystems.” In *22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2015.

Zhao, Mingyi, and Peng Liu. “Empirical Analysis and Modeling of Black-Box Mutational Fuzzing.” In *International Symposium on Engineering Secure Software and Systems (ESSoS)*. 2016.

Zizhen, Lin, and Saga McFarland. *Are China’s ’Ethical Hackers’ Cyber Heroes or Criminals? CaixinOnline*, `http://english.caixin.com/2016-10-17/100997728.html`, October 2016.