A Bug Bounty Perspective on the Disclosure of Web Vulnerabilities

Jukka Ruohonen
University of Turku
Email: juanruo@utu.fi

Luca Allodi
Eindhoven University of Technology
Email: l.allodi@tue.nl

***Abstract*—Bug bounties have become increasingly popular in recent years. This paper discusses bug bounties by framing these theoretically against so-called platform economy. Empirically the interest is on the disclosure of web vulnerabilities through the Open Bug Bounty (OBB) platform between 2015 and late 2017. According to the empirical results based on a dataset covering nearly 160 thousand web vulnerabilities, (i) OBB has been successful as a community-based platform for the dissemination of web vulnerabilities. The platform has also attracted many productive hackers, (ii) but there exists a large productivity gap, which likely relates to (iii) a knowledge gap and the use of automated tools for web vulnerability discovery. While the platform (iv) has been exceptionally fast to evaluate new vulnerability submissions, (v) the patching times of the web vulnerabilities disseminated have been long. With these empirical results and the accompanying theoretical discussion, the paper contributes to the small but rapidly growing amount of research on bug bounties. In addition, the paper makes a practical contribution by discussing the business models behind bug bounties from the viewpoints of platforms, ecosystems, and vulnerability markets.**

***Index Terms*—vulnerability disclosure, vulnerability reward program, bug hunting, bug challenge, open bug bounty, security patching, web vulnerability, cross-site scripting, XSS, CSRF**

I. Introduction

Bug bounties have become increasingly popular in recent years. As a testimony of the popularity, even the United States Department of Defense (DoD) recently piloted a bug bounty program, which further led to a partnership with a crowd-sourcing bug bounty platform [8, 10]. Despite of the popularity, bug bounties are surrounded by many unanswered and controversial questions. These questions range from monetary incentives and ethical practices to the fundamental question of whether bug bounties actually help at improving security.

Motivated by the many unanswered questions, this paper examines the vulnerability disclosure dynamics on the one-sided OBB platform that was launched in 2015 based on the older volunteer-driven XSSPosed platform [12].[1] The term vulnerability *disclosure* frames the scope of the paper: the primary focus is on the dissemination of vulnerabilities through the OBB platform to the vendors affected by the vulnerabilities. The term *one-sided* is used to emphasize that OBB is mostly a community-based platform that neither pays for the vulnerabilities disseminated nor explicitly engages with vendors through a subscription model. As will be shown, both terms are important for a theoretical framing of bug bounties.

The paper's main empirical findings, theoretical points, and contributions can be summarized and generalized as follows:

- Bug bounty platforms in general align well with the *theories about network effects and platform economy*.
- However, when *excluding the crowd-sourcing* elements, *innovation seems limited* from a business perspective; most current bug bounties *mimic the business models that have been used already in the older vulnerability markets*.
- *One-sided bug bounty platforms for web vulnerabilities represent an interesting case of comparison* to two-sided bug bounty platforms such as HackerOne and the older platforms such as the notorious Zero Day Initiative (ZDI).
- In terms of the mere volume of software vulnerabilities disseminated, bug bounties *can be successful without monetary compensations*, although the lack of compensations tends to intensify the focus on *quantity over quality*.
- Only a relatively *few participants disclose most* of the web vulnerabilities disseminated through bug bounties.
- *Automated tools for web vulnerability discovery are used* also in the bug bounty context, and this automation presumably *influences the websites targeted and affected*.
- In addition to the productivity gap between participants, the use of automated tools can create *knowledge gaps* even with respect to a single type of web vulnerabilities.
- The *evaluation of new submissions can be rapid* in the context of simple web vulnerabilities, although it remains an *open question of how well the vulnerabilities disseminated are coordinated and communicated to vendors*.
- *Patching of vulnerabilities by the vendors affected takes a relatively long time* also in terms of low-impact web vulnerabilities often disseminated through bug bounties.
- Patching times vary across both participants and vendors, but *learning from the past disseminated vulnerabilities seems limited*; the reputation of a bug bounty is likely a more important factor affecting the patching times.

[1] The paper covers a period from the platform's initial launch to late 2017. As is typical to current bug bounty platforms, the underlying business models are constantly changing [31, 60]. For instance, OBB recently started to gear itself toward managing vendors' bug bounty programs, while also permitting the submission of new types of web vulnerabilities. Despite of these changes, most of what is being discussed apply also to the situation in mid-2018.

Three additional remarks are required about the terms used. First and foremost: there exists no established terminology to describe the current crowd-sourcing patterns for vulnerability discovery and disclosure. Bug bounties, vulnerability reward programs, security challenges, vulnerability hunting campaigns, and related terms are used more or less interchangeably to describe the same phenomenon [39]. Throughout this paper, the term *bug bounty* is used in the same loose sense. Second, the term (white hat) *hacker* is used for referring to individual participants in bug bounties. Because particularly criminology research has started to equate hackers with computer crime [32], the term security researcher often used in the industry would be a slightly better choice. Nevertheless, both terminology choices are justifiable due to the concrete connections to the current security industry—the term bounty appears in the OBB abbreviation and the term hacker in the name of the likely most famous current bug bounty platform, HackerOne. Last but not least, the term *vendor* is often used to refer to "any producer of software, regardless of whether or not that software is sold commercially" [57]. In this paper, however, vendors are equated to domain names that have hosted the websites affected by the vulnerabilities observed.

The last point requires a further comment. The owner of a domain may not be the same party who is responsible for the website(s) hosted from the domain. In fact, the remediation of some web vulnerabilities may involve domain name registrants, webmasters, hosting providers, and even Internet service providers [70]. In theory, the same may apply to the disclosure of web vulnerabilities. As the bug bounty platform examined does not provide information about the actual vendor-side individuals who were contacted about the vulnerabilities, domain names provide a sensible simplification, however. As will be elaborated, questions related to contact persons differentiate also many bug bounty platforms.

Bug bounties are a challenging research topic. There is at the same time a limited but growing amount of existing research to build upon [22], and a large amount of loosely related research on vulnerabilities. In terms of scholarly disciplines, relevant contributions have been made in computer science and software engineering, information systems research, and what has been branded as (cyber) security economics [4]. In order to maintain this interdisciplinary focus, the remainder of this paper proceeds by first discussing the background in Section II from a socio-technical perspective. This theoretical discussion is used to also motivate the empirical case study presented in Section III. The empirical findings, theoretical points, and few practical insights are discussed in the final Section IV.

II. Background

The following discussion will outline the background by considering some similarities and differences between current bug bounties and the older vulnerability markets. Even though formal economic models have been proposed for approaching bug bounties [9], the theoretical tone adopted for the discussion is more informal, drawing from the platform literature.

A. Bug Bounties

The history of bug bounties traces to the early 2000s emergence of commercial vulnerability disclosure programs and different security alerting services. The perhaps most memorable example is the TippingPoint's Zero Day Initiative that was launched already in 2005. This still active program relies on a business model that compensates hackers for their vulnerability discoveries on one hand, and helps the opt-in customers to patch their products on the other [4]. When reflected against the history of vulnerability disclosure practices, the new element brought forward by ZDI and related programs was the monetary compensations paid for the vulnerabilities disseminated through the programs. These monetary compensations were also a key element in the emergence of crowd-sourced bug bounty programs in the early 2010s. There are currently two main variants of bug bounty programs [22]. These are illustrated in Fig. 1. As a preparation for the forthcoming theoretical discussion, the second variant (B) is further broken down into two subvariants, B.1 and B.2.

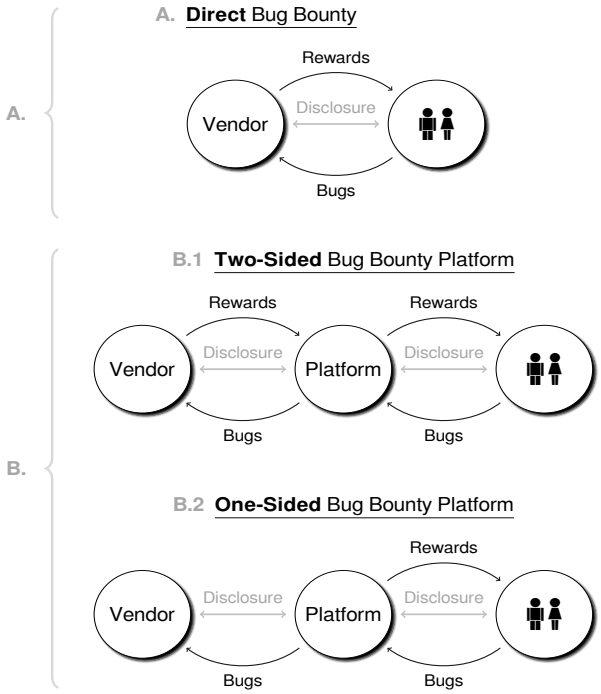

Fig. 1. Bug Bounty Variants

Direct bug bounty programs are nowadays orchestrated by many vendors themselves. In fact, the concept of a (direct) bug bounty is older than ZDI and related programs; Netscape introduced the first known bug bounty already in 1995. Although also the Mozilla Foundation later adopted the same approach, these early initiatives did not gain widespread traction in the software industry. It was much later in the early 2010s when direct bug bounties became commonplace through the initiation of programs by many technology giants, including Google, Microsoft, Facebook, and Yahoo, among others. These bug finding contests directly orchestrated by

technology companies themselves were also quickly adopted for initiating different platforms (see [17] for the concept of platform and related theory). HackerOne is currently the perhaps most famous case of these bug bounty platforms.

B. Bug Bounty Platforms

Many of the two-sided platforms (B.1) rely on modified versions of the business models that were used already in ZDI and related programs. For instance, vendors subscribe to HackerOne in order to improve the security of their software products via security assessments carried out by hackers who are compensated for their vulnerability discoveries. As with ZDI and some later online marketplace endeavors [64], the platform exploits a two-sided market; the platform enables two distinct groups to interact and transact (security) information according to their distinct needs [75]. Unlike the vulnerability and security data feeds provided in ZDI and related programs, however, HackerOne is a proactive rather than a reactive service; vendors are explicitly exposing their products for security assessments. For vendors, the service provided could be even labeled as crowd-sourced penetration testing.

It is important to further remark that there is no need for vendors to participate in order to avoid missing vulnerability information. Consequently, many but not all [80] of the business models lack the element of blackmailing that is implicitly present in the ZDI-style programs [6]. Because vendors are the paying customers, nevertheless, the revenue streams are still comparable to those used by the older vulnerability brokers.

To some extent, the demarcation between the two main theoretical types (A and B) has become slightly blurry as some vendors have started to modify their direct bug bounties toward the direction of platforms. For instance, Google extended its bug bounty program in 2013 to cover also a few security-critical open source software projects [77]. Analogously, a few years later in 2016 the Mozilla Foundation launched a fund to improve security in the open source domain [59]. This initiative can be also considered a platform because the fund is used to pay for security audits and vulnerability discoveries, which are both coordinated through Mozilla. Moreover, bounty systems have recently been expanded toward areas beyond security. Possibly inspired by bug bounties and vulnerability hunting, these two-sided crowd-sourcing platforms offer monetary bounties for implementing new features and fixing conventional (non-security) bugs [23, 36]. A further trend relates to the arrival of different one-sided bug bounty platforms, including the OBB platform studied later on in Section III.

Unlike the two-sided subvariants (B.1), these one-sided platforms do not seek to monetize vendor involvement. Although vendors are encouraged to provide voluntary compensations [56], the one-sided platforms (B.2) do not explicitly pay for the vulnerabilities reported. In this sense, these platforms are not pure vulnerability marketplaces on which each vulnerability is a unit of trade [47]. Instead, the one-sided subvariants provide community-based platforms for hackers to report and disclose vulnerabilities they have discovered. These characteristics imply that the ethos behind the one-sided platforms is closer to the classical topics in vulnerability disclosure. The absence of compelled monetary compensations also implies that the rewards from participation are intrinsic rather than extrinsic. Before proceeding to discuss these rewards in more detail, it should be briefly noted that the two subvariants share both similarities and dissimilarities in terms of the basic theoretical premises for platform success.

C. Network Effects

A fundamental challenge for any platform orchestrator has always related to the creation and maintenance of a critical mass [55, 66]. This question is also well-understood among the current bug bounty platform orchestrators. For instance, many bug bounty websites market themselves by visible announcements about the amount of vendors and hackers who have participated, the amount of vulnerabilities reported through the platform in question, the amount of compensations paid, and so forth. These marketing techniques relate to the concept of network effects [38], which can be further chopped analytically into cross-side (or indirect) network effects and direct (or one-side) network effects [30, 55, 66]. The former effects mean that the value of participating on one side of a platform depends on the amount of participants on the other side. Thus, the more there are hackers participating on a two-sided bug bounty platform, the more there are incentives for vendors to also participate, and the other way around.

Similarly to dating platforms [75], say, these cross-side network effects are also numerically asymmetric in two-sided bug bounty platforms. For a platform orchestrator, one high-profile vendor (such as Adobe or Intel—or DoD) is worth a hundred hackers, in a manner of speaking. As the one-sided platforms do not require explicit vendor participation on the platforms, these cross-side network effects pose a bigger challenge only for the two-sided bug bounty platforms.

The direct network effects refer to theoretical premises in which the amount of participants on one side influences the value for participation on this same side. Social media platforms would be the prime example of such direct network effects; there is only a small incentive to participate on a social media platform unless there are already plenty of other participants. For the two subvariants of bug bounty platforms, there exist analogous challenges related to direct network effects—a critical mass of hackers is required. Likewise, a large amount of well-known vendors participating on a two-sided bug bounty platform likely increases the likelihood that also other vendors will join. Reputation and trust offer one way to meet these requirements in the context of online markets for vulnerabilities and related security items [3, 50, 64]. Different rewards for the vulnerabilities reported provide another way to attain and maintain a critical mass of participants.

D. Rewards

The concept of reward is important for better understanding bug bounty platforms in particular. Five points are worth making about the concept of reward in the bug bounty context.

First, there are monetary rewards and non-monetary rewards [39]. Monetary rewards are the ones grabbing the attention in popular discourse—the outlying payments that can amount even up to a hundred thousand dollars make catchy headlines in media. Although empirical research about the money involved is regrettably limited, there are good reasons to suspect that most monetary compensations are quite moderate, however (see [2, 19, 58, 64, 80], though note also [29]). Despite of the popularity of bug bounties, it seems fair to also question whether a stable supply-demand equilibrium has been reached in terms of the monetary compensations. The history provides a rationale for this caution: the lack of clear reference prices has been a continuing problem in the vulnerability markets [50, 64]. Furthermore, recent industry surveys also indicate that monetary compensations are not commonly expected from reporting vulnerabilities [71]. Therefore, it is important to emphasize the role of non-monetary rewards.

The non-monetary rewards include everything from free-lunch parties to t-shirts, but the most important factor relates to the "acknowledgments in the security hall of fame of the respective bounty program" [39]. Given that career development is one intrinsic motivation for participating in open source software development [13], having a name in a security hall of fame may be a good abstract business card when seeking employment in security companies [60]. For this reason, it should be noted that the importance of acknowledgements is not limited only to bug bounty platforms; also many companies that do not engage with bug bounties maintain their own halls of fame. For instance, the Finnish company Nokia maintains its own hall of fame for those who have disclosed vulnerabilities in the company's products [53], although the company currently neither maintains a direct bug bounty nor participates in two-sided platforms as a customer.

Second, there are non-monetary rewards that originate from the collaboration between hackers participating on a platform. When diverse members of a cohesive social group interact, the shared interests typically facilitate knowledge sharing among the members of the group [49, 54]. As bug bounty platforms help at networking with colleagues [60], these may also increase the competency of a member either through collaboration with other members or by learning from others [79]. By implication, such rewards are also related to direct network effects; the larger the amount of participants, the more there are opportunities for collaboration and learning. As is typical to open source software development and online communities in general [44, 78], also different intrinsic incentives are present [60]. These include abstract rewards related to social approval, enjoyment and leisure time, intellectual stimulation, and other sociological and psychological aspects.

Third, there are rewards for hackers, and costs and rewards for vendors. Orchestrating and maintaining a direct bug bounty program is not free. Depending on a vendor's software portfolio and its size, already the maintenance costs can be noteworthy. These and other related costs are an important element in the business models behind many two-sided bug bounty platforms. For vendors seeking to outsource a portion of security assessments to a crowd, it is presumably cheaper and easier to organize the outsourcing via a third-party platform. Liability, communication, and related aspects likely further increase the lucrativeness of two-sided platforms.

Unfortunately, no empirical research has been done to examine the pricing of bug bounty platforms for vendors. It can be noted that measuring the costs may not be easy because there may be indirect costs in addition to the direct participation expenses. For instance, a vendor who participates in a bug bounty platform may be exposed to a deceptive incentive to underinvest in other secure software engineering practices [41], possibly due to the perceived cost-effectiveness of bug bounties [19, 69]. That is, paying for vulnerabilities may diminish resources from the prevention of vulnerabilities [15]. There is also lack of research on the rewards that two-sided bug bounty platforms offer for vendors. To pinpoint directions for further research in this regard, it can be noted that besides security assurance itself, marketing and public relations constitute a reward. When a vendor participates in a popular and widely known two-sided platform such as HackerOne, it also delivers a public statement that security is taken seriously—regardless whether this is actually true.

Fourth, most current bug bounty platforms harness the rewards for making their platforms profitable and lucrative for both hackers and vendors. Taking cuts from transactions was already a part of the older vulnerability brokerage models (currently, HackerOne takes 20% of the monetary rewards offered by vendors [28]). The harnessing extends also toward the extrinsic and intrinsic rewards for the hackers. For achieving and maintaining the critical mass, many bug bounty platforms rely on so-called gamification techniques (for the concept of gamification see, e.g., [24]). These techniques include metric-based rankings and constantly updated dashboards, badges for most productive hackers, and other commonly used social reputation elements. Moreover, having a name in a platform's security hall of fame is not enough; it is encouraged to also compete in a constantly updated dashboard [34]. As soon discussed, it should be emphasized that both the extrinsic and intrinsic rewards vary in terms of different vulnerability types.

Last, there exists an implicit societal reward. In theory, bug bounties may reduce exploitable vulnerabilities stockpiled by criminals and state-level actors, providing also pull-off incentives that may decrease the probability of participating on illegal underground platforms [2, 81]. The direct network effects involved imply that an already reached critical mass may be also educated and guided toward established security practices, ethical codes of conduct, and more sophisticated vulnerabilities. These points resonate with the classical but still ongoing debates about software vulnerability markets in general. Thus, depending on a viewpoint, the societal reward may also be a liability: instead of working toward the ultimate goal of improving software quality, bug bounties may increase the stockpiling tendency and the exploitation of vulnerabilities [8, 15, 39]. Given these fundamental problems, it is important to emphasize that most current crowd-sourcing bug bounty platforms target low-impact web vulnerabilities.

E. Disclosure on Bug Bounty Platforms

The historical genesis behind the ZDI-like programs was largely related to the altruistic motives in the 1990s full disclosure movement [47]. These motives can be also seen underneath the current bug bounty platforms. To recall, full disclosure refers to a vulnerability disclosure practice via which full technical details are released to the public, possibly regardless whether a vendor was even contacted about the vulnerabilities prior to the release. There were—and still are—good reasons for hackers to prefer this type of vulnerability disclosure. Among these is the reluctance of many vendors to acknowledge and fix the vulnerabilities reported. An intermediate actor is one way to address this problem and related issues affecting the disclosure of software vulnerabilities.

Direct bug bounty programs share a key similarity with the so-called direct disclosure practice through which hackers and vendors communicate privately about the publication and patching of vulnerabilities. This direct (or two-party) disclosure practice [33] has historically been the most common way to disclose vulnerabilities. Many of the alternative practices were also formulated to overcome limitations affecting the two-party direct disclosure type. These alternatives include the full disclosure ideology, disclosure through computer emergency response teams (CERTs), and the ZDI-like brokerage solutions [10, 58]. The similarity between direct disclosure and direct bug bounties relates to the absence of a middleman, whether a commercial vulnerability broker or a CERT. However, there exists also a fundamental difference: with direct bug bounties, vendors are well-prepared to handle vulnerabilities disclosed to them. Whereas a classical direct disclosure of a vulnerability may come out from the blue sky to a vendor, a vulnerability disclosed through a direct bounty program is—or at least should be—received by a dedicated contact team.

The current bug bounty platforms lean toward either the direct disclosure practice or the hybrid brokerage models. A key differentiating factor relates to a platform's role in coordinating the disclosure. If a platform takes a weak brokerage position, the platform does not explicitly coordinate the disclosure process between vendors and hackers; the process cannot be outsourced to the bug bounty platform. For instance, the primary way to handle disclosure on the HackerOne's platform is to directly disclosure information to vendors based on the contact details provided by the platform. While there is an additional service for helping hackers with the initial handshaking [27], the service does not mean that HackerOne would be the primary coordinator. Analogous point applies to the OBB platform. In contrast, some bug bounties take a much stronger brokerage position, handling all communication with vendors on behalf of the hackers [60]. It should be noted that some implicit mediation is still present even when explicit brokerage is not implemented. Any bug bounty platform is still implicitly present in the disclosure processes; already the name of a platform may carry some authority for influencing vendors' behavior. When a vendor knows that a bug bounty platform is involved, communication may be easier compared to classical pure direct disclosure. This theoretical reasoning is summarized with the cross-tabulation shown in Table I.

TABLE I
A TOPOLOGY OF DISCLOSURE BROKERAGE

		Brokerage	
		Weak	Strong
Platform	Two-sided	HackerOne	ZDI
	One-sided	OBB	Vulnerability Lab?

While different forms of brokerage may be implemented in both one-sided and two-sided platforms, there is a key difference between these two theoretical types with respect to vulnerability disclosure—vendors are explicitly participating on two-sided platforms as paying customers. Due to the lack of explicit vendor-side engagement, vulnerability disclosure is presumably more difficult to carry out through one-sided platforms. A further differentiating factor relates to the type of vulnerabilities disseminated through bug bounty platforms.

Bug bounties vary in terms of the types of vulnerabilities typically disseminated. The hierarchy is usually clear in terms of both extrinsic and intrinsic rewards: remote code execution vulnerabilities and related memory corruption issues are usually at the top and web vulnerabilities at the bottom. Like with vulnerability markets in general [1], this hierarchy also influences the types of hackers who are likely to successfully discover and disclose vulnerabilities. A bug bounty that targets memory corruption vulnerabilities is likely to attract high-skill professionals, whereas web vulnerabilities are easy to discover even with moderate computing knowledge. This skill gap likely contributes to the typical problems affecting vulnerability disclosure, including the frequent delays for vendors to release patches to the security issues disclosed. The absence of vendors' participation and the context of web vulnerabilities are important characteristics of the one-sided OBB platform.

III. A PRELIMINARY ANALYSIS OF A ONE-SIDED PLATFORM FOR WEB VULNERABILITIES

In what follows, a preliminary empirical analysis is presented about the one-sided OBB platform based on a dataset collected from the platform's online website in October 2017. The dataset analyzed contains 158794 web vulnerabilities. Before proceeding to tackle this vast amount with descriptive statistics, a brief discussion is necessary about the type of vulnerabilities present in the dataset and the vulnerability disclosure practices on the OBB platform. After this discussion, the empirical analysis proceeds by first considering the evolutionary and productivity aspects in relation to earlier work done by Zhao, Grossklags, and associates (see [80] in particular). The second part of the empirical analysis focuses on the evaluation of new submissions and the time vendors (or, rather, websites) take to patch the issues disclosed to them.

A. Web Vulnerabilities and the OBB Platform

The OBB platform only permits the submission of cross-site scripting (XSS) and cross-site request forgery (CSRF) vulnerabilities. Ever since the initial surge in the early 2000s, cross-site scripting bugs have continued to remain the likely most common software vulnerability type [40, 67]. Although there exists a myriad of different XSS vulnerabilities, the essence is that an attacker injects malicious code to a benign website, and this code is executed by the browser of another user visiting the website (see [26] for a concise recent review on XSS). Even though automated black box scanners are presumably used also in the bug bounty context [1, 5, 45], cross-site scripting vulnerabilities are also easy to discover simply by browsing a website and inserting small blocks of code to parts that require user input. In contrast to XSS, a CSRF vulnerability exposes a weakness that may trigger a client to make an unintended request (for the background see, e.g., [14]). For instance: if a client is currently authenticated to a website, a CSRF vulnerability may be exploited by luring the client to click a forged link that makes an unwanted state change, such as changing the client's password.

In general, cross-site scripting vulnerabilities are much more common than CSRF vulnerabilities. This point can be shown also with the OBB dataset: only as little as 38 of the vulnerabilities in the dataset dealt with CSRF. Thus, OBB is very much a platform specialized to cross-site scripting vulnerabilities, just like its predecessor, XSSPosed, was explicitly.

The focus on XSS and CSRF vulnerabilities affects the types of hackers who are likely to participate on the OBB platform, but there are a couple of additional reasons why the web context is important. First, the discovery of these web vulnerabilities requires no intrusive testing techniques [68]. This point is important for a one-sided platform; reporting vulnerabilities found by intrusive techniques may easily expose a platform to liability questions—after all, legal issues have not been unheard-of also in the web vulnerability context [48, 81]. Likewise, the difficult ethical questions that surround some bug bounties [15, 76] are a lesser concern in the OBB case. Second, the dynamics of finding vulnerabilities are different. The world wide web is an endless resource. By implication and in contrast to some vendor-specific bug bounties [46, 81], it cannot be assumed that finding new vulnerabilities would get more difficult over time in the OBB case. Given that the current size of the indexed world wide web is estimated to be around five billion pages or more [73], there will always be vulnerable websites to discover—and rediscover.

B. Disclosure on the OBB Platform

The OBB's disclosure model is illustrated in Fig. 2 with four abstract actor types and six events. Five of the events equate to timestamps marked with a symbol τ. Various axioms could be postulated for the possible ordering of these events, but, in general, it suffices to note that at least $\tau_a \leq \tau_b \leq \tau_d$ always holds, provided that all three events actually occur. While τ_a and τ_b are always defined in the analytical model, the set $\{\tau_c, \tau_d, \tau_e\}$ may be undefined. For instance, the timestamp τ_d remains undefined in case a vendor never patches its website.

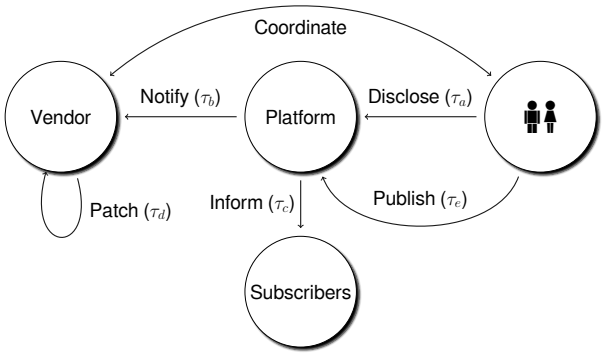

Fig. 2. Disclosure on the OBB Platform (analytical sketch)

The process starts when a hacker discloses a vulnerability to the platform (τ_a). As is typical in the vulnerability context [61, 64], the platform orchestrator then carries out an evaluation to rule out fake submissions, bugs that are not actually vulnerabilities, and other false positives that are typical menaces of bug bounties [42, 81]. After the evaluation has been completed, either a customized message or a bulk notification is send to the vendor (τ_b). This notification includes the contact details for reaching the hacker who made the discovery and disclosed the vulnerability on the platform.

Thus,

$$\mathrm{TTE} = \tau_b - \tau_a \geq 0 \tag{1}$$

defines a straightforward *time-to-evaluate* (TTE) metric. The actual coordination is left for the two parties. If a vendor acknowledges the notification, either explicitly or implicitly, and the possibly required further coordination is successful, the vendor likely also patches the vulnerability at time τ_d. Provided that τ_d is defined, such that patching occurred, a conventional metric [35], say *time-to-patch* (TTP), is given by

$$\mathrm{TTP} = \tau_d - \tau_b \geq 0. \tag{2}$$

The model also includes a set of subscribers for whom the OBB platform provides security information about the vulnerabilities disclosed through the platform. Moreover, the public disclosure on the platform occurs at τ_e based on the discretion of a hacker. This assumption is theoretically important. An intermediate actor may hold the decision to publish vulnerability information in strong brokerage models [37], but OBB leaves this decision to hackers. If a hacker decides not to publish details, τ_e remains undefined. Given that many of the intrinsic rewards depend on the availability of public information, these cases are supposedly rare, however.

Finally, it should be noted that OBB follows the so-called responsible disclosure [62] by providing a grace period before hackers are allowed to disclose public information on the platform. If a vendor patched a vulnerability, a 30 day grace period is provided; otherwise a 90 day restraint is applied [56]. These lengths are comparable to current industry practices.

C. Evolution

The initial evolution of a new platform provides a good bird's-eye view on the prospects for the future success of the platform. The volume of vulnerabilities disseminated through a bug bounty platform provides the most straightforward metric to observe the evolution [80]. In addition, it is important to consider the network effects; reaching a critical mass is important early on. For a new two-sided bug bounty platform, it is particularly important to promptly attract a group of well-known vendors. As it is a common mistake to overemphasize pricing aspects during the initial evolution of a platform [7], the names of the vendors likely carry more weight than the compensations paid by the vendors in the early stages. For both two-sided and one-sided bug bounty platforms, a key factor is also the initial amount of active hackers; when a platform is able to attract a group of productive hackers early on, the incentives may intensify for others to join. This direct network effect is fostered by social media. Analogous to other bug bounty platforms [34], also OBB relies heavily on social media—in fact, a Twitter account is required for reporting vulnerabilities on the platform. Given these basic premises for network effects, Fig. 3 shows three key metrics on the monthly evolution of the OBB platform during the period observed.

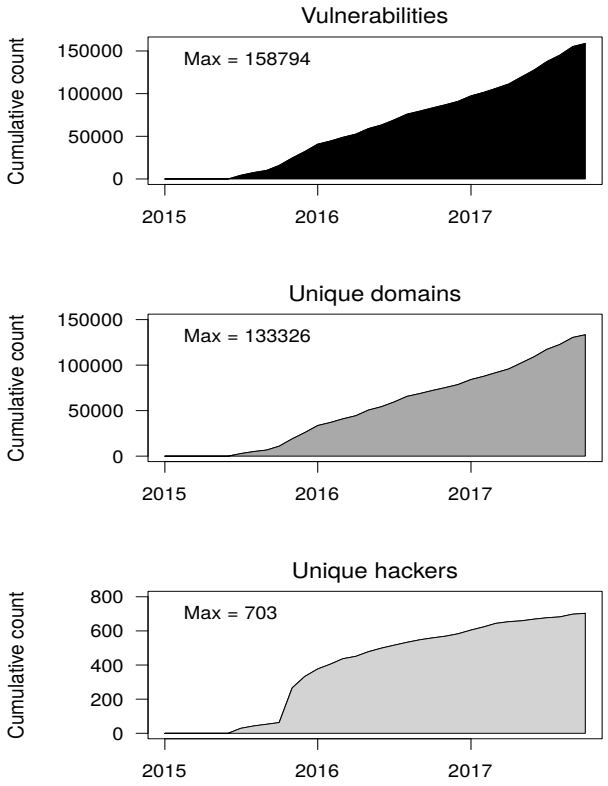

Fig. 3. Three Key Metrics on the Initial Evolution of the OBB Platform

The volume of vulnerabilities disseminated and the amount of unique domains that were affected by the vulnerabilities both follow almost a perfectly linear trend. A peek at the y-axes also reveals that only one XSS vulnerability was reported for most of the domains. In fact, only about 26% of the unique domains were affected by multiple vulnerabilities. This observation is interesting because it would seem sensible to hypothesize that finding one XSS vulnerability would increase the probability of finding more. Given the social aspects of bug bounties (see Subsection II-D), it could be also asserted that when one hacker finds web vulnerabilities from a domain, the probability would increase for others to also take a look at the domain. This line of reasoning does not seem plausible according to the results, however. As will be discussed later on, the explanation likely relates to the use of automated tools.

The amount of unique hackers participating on the OBB platform started to rapidly increase in the late 2015. After this initial surge, the growth rate in the amount of new participants has slowly started to decrease. Although the period observed is too short for making definite conclusions, this mild deceleration hints that there may be a saturation point in terms of the global amount of hackers who are engaging with bug bounties. Thus, a good hypothesis for further work would be the examination of a potential S-shaped growth curve that typically characterizes the diffusion dynamics of many online platforms in general [7, 55]. Another point is that the total amount of unique participants generally aligns rather well with previous empirical observations about bug bounties.

Although the total of 703 unique hackers is much lower than what has been observed for Chinese bug bounties [34, 79], the amount is still comparable in magnitude to platforms such as HackerOne [80]. There likely exists also a crossover effect; hackers tend to switch from one bug bounty to another [46]. Such crossover effects were typical already in the older vulnerability markets. For instance, a common speculation has been the possibility to sell a vulnerability in one market and an exploit for the vulnerability in another [64]. Likewise: direct, full, or some other type of vulnerability disclosure may be pursued only after a failed attempt to sell the vulnerability to a broker. An analogous pattern may be present with respect to bug bounties: if a hacker failed to obtain a compensation from a two-sided bug bounty platform, she may decide to publish the discovery on a community-based platform such as OBB. Duplicate reports between platforms are also a real possibility.

D. Productivity

The little over seven hundred hackers who participated on the OBB platform between 2015 and late 2017 disclosed nearly 160 thousand XSS vulnerabilities. This amount is substantial even when keeping in mind the low-profile of cross-site scripting bugs. The volume is also so large that the discoveries cannot have had happened through manual inspection of websites. In other words, human intelligence is generally important for finding security bugs [46], but human intelligence may be even more important in terms of engineering software solutions for automated (web) vulnerability discovery. Manual source code inspection often works well for finding new vulnerabilities [18, 20], but automatic scanning is also a necessity insofar as the whole world wide web is the

target. Although finding XSS bugs does not require extensive knowledge as such, engineering automated scanners is another thing. The resulting knowledge gap is one factor contributing to the typically uneven distribution of web vulnerability disclosures among hackers participating on a bug bounty platform.

For instance, the most productive hacker on the OBB platform has disclosed over 23 thousand XSS vulnerabilities. This amount is comparable in magnitude to what can be reached with large-scale Internet scanning of cross-site scripting vulnerabilities [43], especially since many websites continue to remain vulnerable even after the corresponding web vulnerabilities have been disclosed and reported [26, 68]. However, the overall consequences for a bug bounty platform presumably remain similar irrespective of whether the vulnerability discoveries result from manual labor or automated tools. Analogous to many online platforms in general, a long-tailed probability distribution likely follows in terms of the per-hacker amount of vulnerability disclosures made on a platform.

To examine this typical productivity gap further, Fig. 4 displays the cumulative distribution function (CDF) for the per-hacker amount of vulnerabilities disclosed on the OBB platform. The two distribution approximations visualized are based on the classical estimation setup for examining so-called power-laws [11, 25]. In addition to the apparent productivity gap, there are three points worth making from the illustration.

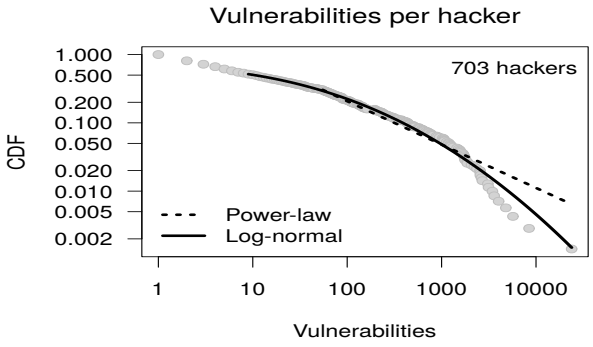

Fig. 4. Productivity of Hackers

First, the log-normal distribution seems to provide a slightly better fit than the power-law one. From a purely empirical point of view, this stylized fact is noteworthy because it differs from previous observations about bug bounties [46, 79]. The log-normal distribution is also more challenging to interpret theoretically than a power-law distribution that can be attached to theoretical constructs such as preferential attachment.

Second, the productivity gap causes a vulnerability for the OBB platform. The same applies to most bug bounties [80]. If the OBB platform would lose some of the most productive hackers, the volume of new vulnerabilities disseminated would presumably decrease substantially. This risk is fostered by the network effects that work also toward the reverse direction. In other words: when a sufficiently large group of participants abandon a platform, a second group of participants may follow. In addition to providing incentives for the most productive hackers to stay on board, it is important to consider means by which the productivity of other hackers could be improved. Given that about 19% of the unique hackers on the OBB platform have disclosed just one vulnerability, which is fairly typical for bug bounties [31], a further challenge relates to the common question of how to transform the apparent one-shot visitors into persistent users of the bug bounty platform.

Third, the uneven productivity has also other important theoretical consequences. In particular, the gap implies that merely increasing the volume of hackers is unlikely to substantially increase the volume of vulnerabilities disseminated. This assumption runs in counter to the basic hypotheses often made in the general platform literature [19, 75]. In the context of web vulnerabilities, an analogous diversity tenet can be also approached in terms of the websites and domains affected.

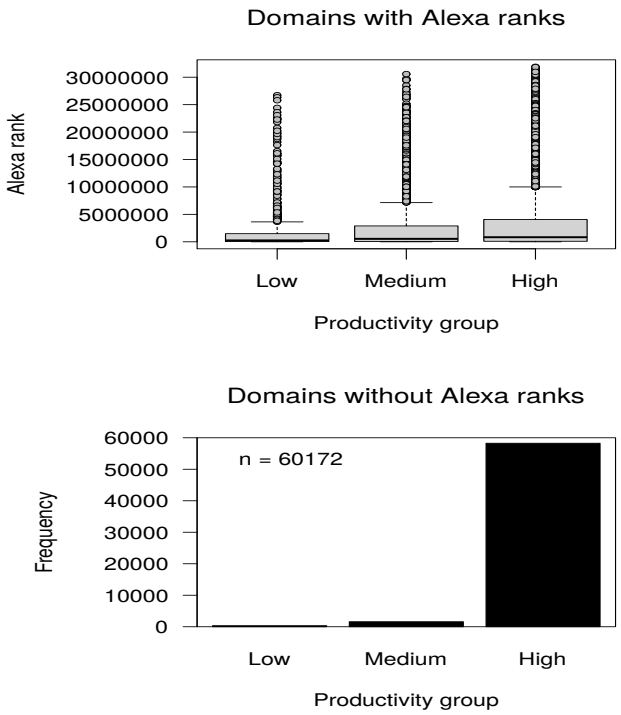

Fig. 5. Productivity and Popularity of Websites

The popularity of the domains offers a simple way to probe this kind of diversity. The basic hypothesis is that hackers who are particularly productive would target more popular domains and websites hosted from these domains, while the rank and file hackers would focus on the less popular domains [80]. The basic idea is sound. For instance, there is still some prestige involved in discovering XSS vulnerabilities from websites owned by Google or Netflix—or from online banking sites for which even XSS can pose a real threat. Following existing research [79, 80], the hypothesis can be examined by plotting Alexa's popularity ranks against different productivity groups. The three groups used in Fig. 5 are based on a simple classification: the "low" productivity group

refers to hackers who have disclosed less than or equal to the median of ten vulnerabilities per-hacker, the "high" productivity group contains those who have disclosed more than the 75th percentile of the per-hacker disclosures made, and the "medium" group sets in-between these two groups. The popularity ranks refer to those given by the OBB platform.

The results are more or less consistent with previous observations [80]. Although there is hardly a difference between the median popularity ranks of the three groups, there is a small tendency for the highly productivity hackers to disclose XSS vulnerabilities affecting less popular domains. This observation is reinforced by the lower plot in Fig. 5, which shows that almost all of the vulnerabilities affecting unpopular domains without Alexa's ranks were disclosed by the high-productivity group. Thus, all in all, the diversity-based hypotheses do not seem sensible for the OBB platform. The explanation may again relate to automation. In other words, running a large-scale XSS scanner is dependent on the empirical sampling and seeding characteristics. If a scanner uses hyperlinks and web crawling to find new targets, it is unlikely that the cross-site scripting findings would be consistent with website popularity.

E. Evaluation

The evaluation of new submissions is a generic problem in bug bounties and software bug tracking in general. The problems in triaging of vulnerability reports have also intensified in recent years, partially owing to the popularity of bug bounties and their monetary compensations that tend to incentivize poorly assembled reports and even fake submissions [42, 61]. Given the fundamental nature of the problem, there is also a long history in software engineering for automating at least some aspects of bug triaging [72]. The volume of web vulnerabilities disseminated through the OBB platform implies that automation is also a necessity. Fortunately, automatic evaluation of typical cross-site scripting vulnerabilities is easy.

The OBB platform uses a simple web form for reporting new vulnerabilities. For XSS issues, the form contains the typical `<script>alert('XSS')</script>` -style payload embedded to a uniform resource locator together with potentially required parameters. Given the information submitted via the form, automatic verification is easy. It should be also possible to use polling for automatically evaluating whether and when the issue is patched by the website affected [68]. Given this background, the results summarized in Fig. 6 are understandable and sensible. Before continuing further, it should be remarked that about eight thousand vulnerabilities had to be removed due to missing τ_a or τ_b used to define TTE. Furthermore, the sketch in Fig. 2 is not entirely accurate because some older vulnerabilities are rather accompanied with explicit timestamps denoting the dates and times on which the vulnerabilities were evaluated. When computing the TTE metric, these explicit timestamps are used when available. Furthermore, newer vulnerabilities are actually accompanied with two notification timestamps that record the dates and times on which custom and generic notifications were sent to the vendors. The smaller of the these is used to define τ_b, provided that an explicit evaluation timestamp is not available.

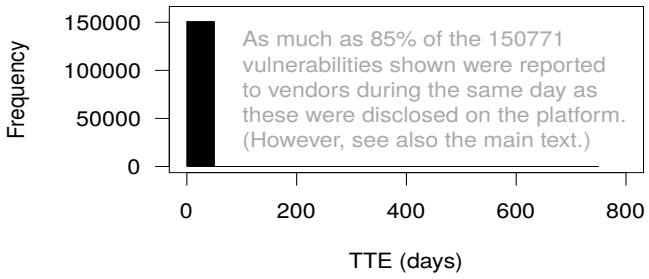

Fig. 6. Evaluation Times

Thus, the OBB platform is good at automatic evaluation of new submission due to the focus on XSS. However, there is another viewpoint to the time-to-evaluate metric. As was discussed in Subsection III-B, the timestamp τ_b refers to a vulnerability notification sent to a vendor. While the evaluation of a XSS vulnerability may be fast, establishing a contact to the website or domain affected is an entirely different thing. In fact, previous research indicates that many responsible vendor-side parties are not even reachable with notifications about web vulnerabilities [68, 74]. Analogous point can be made also with the disclosure help offered in some bug bounties [27]. Thus, either the OBB's orchestrators are exceptionally good at contacting vendors or the τ_b event in Fig. 2 is exposed to some validity concerns. In other words, it may well be that the XSS vulnerabilities disclosed on the platform are verified to be real, but the contacts made to vendors contain shortcomings. This potential deficiency contributes directly to the times vendors take to patch the cross-site scripting issues reported to them.

F. Patching

The time a vendor takes to patch its products is a classical research topic in the vulnerability disclosure literature. In addition to the noted reporting aspects, there are numerous factors that may influence the typically lengthy time delays. Among these are the type and severity of the vulnerabilities disclosed, the products affected and their age, the quality of a software source code base in general, shared code bases and third-party libraries, company policies and governmental regulations, the potential presence of a third-party coordinator and grace periods, trust, communication skills and the ego of a hacker, and numerous related factors [19, 33, 35, 37, 62, 65]. Like with bug fixing in general [16], also social media has recently brought a new element that may affect the delays.

While most of these factors may affect the patching times also in the bug bounty context, it seems reasonable to start from the premise that the type and reputation of a bug bounty platform play decisive roles. If a vendor participates in a two-sided platform as a paying customer, the patching times should be faster compared to one-sided and community-based platforms. Another decisive factor would be the vulnerabilities

disseminated through a bug bounty. Accordingly, XSS vulnerabilities should be patched relatively fast already because it is usually trivial to correct these in the software source code. This hypothesis does not hold for the OBB case, however.

After again removing a little over 32 thousand vulnerabilities due to missing timestamps, the time-to-patch values can be summarized in the form of a histogram shown in Fig. 7. As can be seen, the patching times have generally been relatively long. The mean and median of the TTP metric both indicate that vendors have taken over seven months to patch the reported XSS vulnerabilities on average. While the standard deviation is also large, the shape of the histogram does not resemble a long-tailed probability distribution seen for the TTE metric. About 14% of the vulnerabilities were fixed in less than three months, but rest of the vulnerabilities scatter rather evenly up to the maximum of a little over two years.

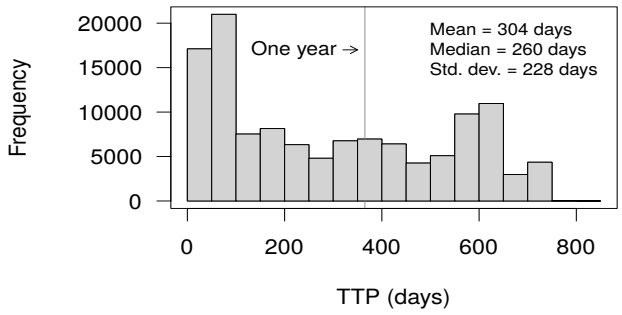

Fig. 7. Patching Times

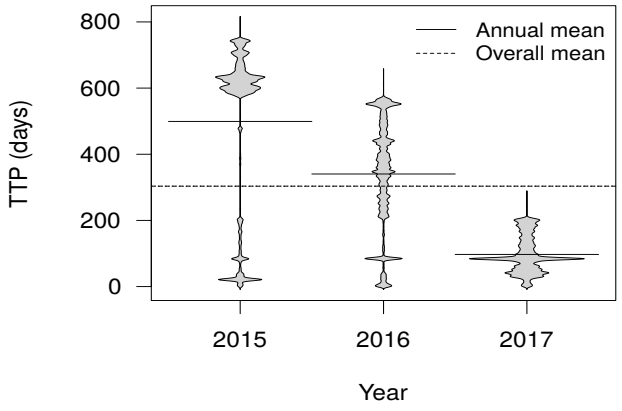

Fig. 8. Annual Patching Times

Despite of the stable linear growth trend in the volume of vulnerabilities disseminated (see Fig. 3), the patching times have also steadily decreased after the initial launch of the platform (see Fig. 8). A possible explanation may be simple: OBB has become more popular and better known already due to the volume of vulnerabilities disseminated. Another point is that the TTP values are not bad as such. In fact, the patching times are even surprisingly similar to those that have been observed for bug bounties targeting conventional software products such as web browsers [19]. Of course, this remark should not be used to hide the fact that many of the websites and domains affected remain vulnerable. The missing values for the τ_d timestamps hint that at least 19% of the cross-site scripting vulnerabilities are likely still exploitable today.

TABLE II
CORRELATIONS BETWEEN POPULARITY AND DELAY METRICS

	Time-to-evaluate	Time-to-patch
Subset n	94145	75834
Pearson r	0.002	-0.003

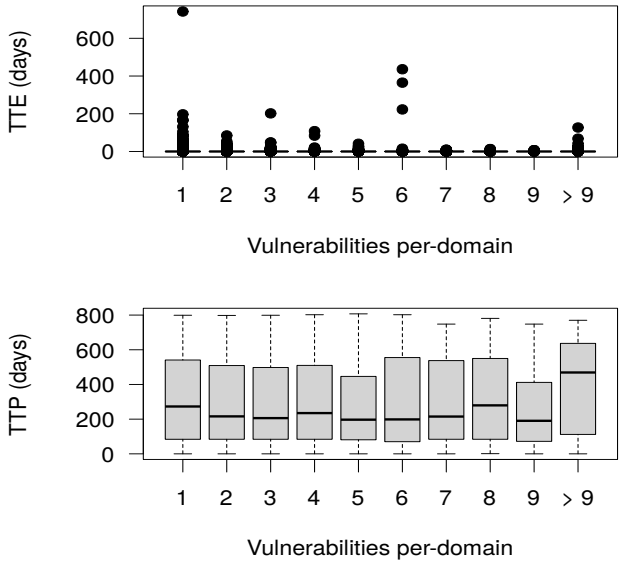

Fig. 9. TTE and TTP Across Domains Affected by Multiple Vulnerabilities

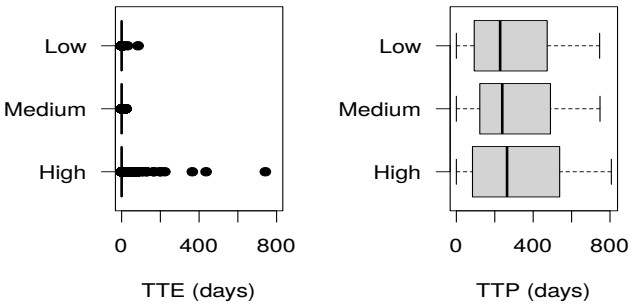

Fig. 10. TTE and TTP Across Hackers' Productivity Groups

There are a couple of additional hypotheses worth briefly visiting. The first is that the popularity of a domain would proxy the maintenance effort for the domain, and this effort would correlate with the patching times. As there are multiple

dimensions to the concept of effort in the web context [70], an alternative speculation could be that the evaluation and patching times would be shorter for the popular domains due to the availability of unambiguous contact details. Regardless of the interpretation, the numbers shown in Table II indicate no relation between the popularity ranks and the delay metrics.

The other hypothesis relates to learning effects. The basic theoretical rationale for such effects is simple: a hacker who discloses thousands of vulnerabilities may become better at communicating with vendors, while a domain affected by multiple vulnerabilities may patch faster due to a similar learning effect. According to Fig. 9, the medians for the TTE and TTP do not notably differ between domains affected by a single XSS and domains exposed to multiple cross-site scripting bugs. Nor is there a visible learning effect in terms of the three productivity groups for the hackers. In general, learning from the past seems limited on the OBB platform.

IV. Discussion

The results presented allow to start with an answer to the always good question about whether things might have changed regarding basic programming mistakes such as input validation [67]. Alas, the answer is clearly negative. Despite of over a decade worth of education and security awareness campaigns, basic issues such as cross-site scripting are still extremely common. In fact, the nearly 160 thousand web vulnerabilities disclosed through the OBB platform between 2015 and late 2017 is about twice the whole amount currently archived to the National Vulnerability Database, for instance.

The initial evolution of the OBB platform indicates that there is at least a good promise for bug bounty platforms that do not provide explicit monetary compensations. The amount of XSS vulnerabilities disclosed through the OBB platform is a laudable achievement from the idealistic viewpoint that the fundamental goal would be to improve security in the world wide web. The platform has also made its own contribution to security awareness through the communication with website maintainers and web developers, as well as through the presence in Twitter. There are also many challenges for both research and practice. To outline some of these challenges, the remainder of the paper enumerates a few directions for further research and catalogs some points that may be worthwhile to consider by the orchestrators of current bug bounty platforms.

The presented distinction between one-sided and two-sided bug bounty platforms offers a good way to proceed into more theoretically motivated comparisons. Given the continuing lack of business model research in the contexts of vulnerability markets and security industry in general [64], the platform (or ecosystem) literature provides also an extensive base for reference theories, hypotheses, and practical insights. The basic premise here is that there should be a difference between one-sided and two-sided platforms, money and non-money platforms, vendor-specific and community-based platforms, or targeted and untargeted bug bounties. While the contemporary platform economy is very much also a copycat economy, it is still surprising that the basic premise does not clearly manifest itself in the current bug bounties. When comparing the results presented to existing research—and when looking at the explicit comparisons already done in existing empirical research [19, 79, 80], the current bug bounty platforms appear extremely similar to each other in many respects. All typical traits of online platforms and their communities are present.

Among these traits is the productivity gap between hackers. A small group of people usually do most of the work on online platforms, and a small group of hackers disclose most of the vulnerabilities on bug bounty platforms. The commonplace network effects pose a double challenge for one-sided platforms: a platform should at the same time lure new productive participants and ensure that the existing productive participants continue to use the platform. While both challenges have been implicitly addressed in the existing bug bounty research, there is still room for further empirical inquiries also in this regard. For instance, (1) the so-called *diffusion* tradition [7, 55, 63] could be followed for examining not the volumes as such but the acceleration, deceleration, and potential saturation aspects related to the volumes. For both Chinese and Western bug bounties, there may be an upper limit in the potential amount of hackers who are able or willing to engage with bug bounties.

The productivity gap correlates with a knowledge gap. Given that some outlying participants on the OBB platform have disclosed tens of thousands of web vulnerabilities, it seems that the possession and use of automated tools contributes to the productivity gap significantly. By implication, there presumably exists also a knowledge gap in terms of the ability to engineer automated XSS scanners. Cross-site scripting vulnerabilities remain at the bottom of the pile in terms of prestige, but it seems that XSS alone can differentiate a hacker demography. Narrowing the productivity gap by narrowing the knowledge gap is an important point for bug bounty orchestrators to consider. Knowledge sharing is at the center of this question. But while learning from others is often noted as important for bug bounties [60, 80], following someone in social media hardly equates to actual learning.

Thus, (2) it may be important to encourage the *sharing* of technical details and software source code for finding new web vulnerabilities. Given the competitive aspects intentionally promoted in current bug bounties, voluntary sharing of technical details may be difficult to achieve, however. For balancing the competition, it might be reasonable to recommend that the bug bounty orchestrators should themselves provide blueprints and drafts for web vulnerability discovery à la Stack Overflow. After all, "smashing the stack for fun and profit" was a decisive historical learning moment for the older generation of hackers.

Now that the OBB platform has geared itself toward subscriptions and multiple types of web vulnerabilities, it becomes important for the orchestrators to consider the means by which quality could be incentivized over quantity. Given the lack of monetary compensations, (3) improving the *gamification* techniques may provide a lever, although it remains unclear how these techniques actually influence the intrinsic rewards. Even when monetary compensations cannot be relied upon, (4) the common *subsidization* techniques [7] may provide a

further option. For instance, it might be possible to consider (per hacker or per domain) varying grace periods, or to reward those hackers who are able to successfully coordinate with vendors. From a more contentious viewpoint, (5) even the platform's current *strategic goals* might be challenged. In other words, perhaps it is precisely XSS and the associated large-scale Internet scanning that sets OBB apart from the competitors. After all, platforms such as Shodan have shown that there is a general demand for Internet scanning services.

In terms of Internet scanning, the (open) data provided by the OBB platform leaves also some question marks. In particular, something does not match in terms of the evaluation times presented. While it is possible to automate the verification of XSS bugs, it is a different thing to automate successful notifications sent to the websites and domains affected. The platform's orchestrators have promoted new initiatives such as the so-called `security.txt` proposal [21], but the problem is that even the older reporting standards are frequently ignored by vendors and websites' owners [74]. In terms of further research, (6) a good question would be to evaluate how well the *reporting practices* perform in practice. The idea about submitting vulnerability reports based on research results has also been toyed in the past [68]. The XSS context would make it relatively easy to assemble a sufficient empirical sample that could be submitted to OBB or (and) some related bug bounty.

The empirical aspects can be approached also from a different angle; open data can be a success factor for bug bounty platforms [80]. More generally, (7) there would be an excellent opportunity particularly for one-sided community platforms to ride on the current artificial intelligence and *data mining* hype. For instance, the results presented indicate that website popularity, the productivity gap, and the learning effects do not affect the patching times. However, (8) the *maintenance effort* of websites would be a good hypothesis to examine in further research [70]. To avoid so-called post-hoc analysis, it would be preferable for data mining if a platform would record basic information about websites before and after a vulnerability is reported or patched. In terms of the potential data to record, a good start would be the hypertext transfer protocol headers and the hypertext markup language of the web pages affected.

In terms of empirical academic research, it can be also noted that the existing bug bounty research (including this paper) has focused on different aggregated metrics such as the volumes of vulnerabilities and hackers. However, (9) it may well be that contextual *outliers* are more interesting and relevant to examine. For instance, the character string `bank` appears in 464 domain names present in the sample used in this paper. Also `equifax.com` and `equifax.co.uk` are represented.

The points mentioned can be combined with other business considerations. The business models underneath the current bug bounty platforms have followed closely the historical heritage from the older vulnerability markets. However, it seems that some of the historical ideas and options have not yet been tried in the bug bounty context. For instance, (10) *auctions*, product *bundles*, and *insurance* contracts have been frequently discussed in the context of vulnerability markets [6, 50, 64], but concrete experimentation with such options seems limited in the bug bounty context. While such options are more relevant for the two-sided bug bounty platforms, new openings are available also for community-based one-sided platforms.

Given the popularity of different hacking contests [52] and security exercises [51], it might be possible to move away from the prevalent vendor-specific paradigm by considering the subscription concept from a different viewpoint. In terms of the OBB platform and cross-site scripting, (11) one option might be to offer a service for registering *domain name lists* to be scanned and evaluated by the hackers participating on the platform. Such a service would allow to also evaluate whether a platform can actually make a difference in terms of security.

Finally, the current bug bounty ecosystem as a whole can be viewed also as a rivalry between different platforms. The competitive aspects offer also new business opportunities. For instance, to maximize revenues, publicity, and user participation, (12) *alliances* between platforms are commonly used in some industry sectors [7, 55]. In addition to the maximization aspects, alliances may reduce the frequent hopping from one bounty to another [46], which likely contributes to the prevalence of invalid submissions, false positives, duplication of work, and related practical problems. Now that also governmental agencies and public sector organizations have started to participate in bug bounty platforms and implement bug bounties of their own, it becomes important to consider whether CERTs or related institutions should participate in the bug bounty ecosystem. New innovations are required also for improving the vulnerability disclosure practices and processes, which continue to be a bottleneck also in many bug bounties.

REFERENCES

[1] A. M. Algarni and Y. K. Malaiya. Software Vulnerability Markets: Discoverers and Buyers. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(3):480–490, 2014.

[2] L. Allodi. Economic Factors of Vulnerability Trade and Exploitation: Empirical Evidence from a Prominent Russian Cybercrime Market. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS 2017)*, pages 1483–1499, Dallas, 2017. ACM.

[3] L. Allodi, M. Corradin, and F. Massacci. Then and Now: On the Maturity of the Cybercrime Markets: The Lessons that Black-Hat Marketeers Learned. *IEEE Transactions on Emerging Topics in Computing*, 4(1):35–46, 2016.

[4] R. Anderson and T. Moore. Information Security: Where Computer Science, Economics and Psychology Meet. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367:2717–2727, 2009.

[5] E. Bazzoli, C. Criscione, F. Maggi, and S. Zanero. XSS PEEKER: Dissecting the XSS Exploitation Techniques and Fuzzing Mechanisms of Blackbox Web Application Scanners. In J.-H. Hoepman and S. Katzenbeisser, editors, *Proceedings of the 31st IFIP TC 11 International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2016)*, pages 243–258, Ghent, 2016. Springer.

[6] R. Böhme. A Comparison of Market Approaches to Software Vulnerability Disclosure. In G. Müller, editor, *Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006), Lecture Notes in Computer Science (Volume 3995)*, pages 298–311, Freiburg, 2006. Springer.

[7] T. R. Casey and J. Töyli. Dynamics of Two-Sided Platform Success and Failure: An Analysis of Public Wireless Local Area Access. *Technovation*, 32(12):703–716, 2012.

[8] A. T. Chatfield and C. G. Reddick. Cybersecurity Innovation in Government: A Case Study of U.S. Pentagon's Vulnerability Reward Program. In *Proceedings of the 18th Annual International Conference on Digital Government Research (g.o 2017)*, pages 64–73, New York, 2017. ACM.

[9] J. P. Choi, C. Fershtman, and N. Gandal. Network Security: Vulnerabilities and Disclosure Policy. *The Journal of Industrial Economics*, 58(4):868–894, 2010.

[10] U. Ķinis. From Responsible Disclosure Policy (RDP) towards State Regulated Responsible Vulnerability Disclosure Procedure (hereinafter – RVDP): The Latvian Approach. *Computer Law & Security Review*, 34(3):508–522, 2017.

[11] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Review*, 51(1):661–703, 2009.

[12] D. Correa. XSSPosed Launches Open Bug Bounty Programme for Web Flaws. SC Media, available online in December 2018: https://www.scmagazineuk.com/xssposed-launches-open-bug-bounty-programme-for-web-flaws/article/534341/, 2015.

[13] K. Crowston, K. Wei, J. Howison, and A. Wiggins. Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. *ACM Computing Surveys*, 44(2):7:1–7:35, 2012.

[14] A. Czeskis, A. Moshchuk, T. Kohno, and H. J. Wang. Lightweight Server Support for Browser-Based CSRF Protection. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, pages 273–284, Rio de Janeiro, 2013. ACM.

[15] S. Egelman, C. Herley, and P. C. van Oorschot. Markets for Zero-Day Exploits: Ethics and Implications. In *Proceedings of the 2013 New Security Paradigms Workshop (NSPW 2013)*, pages 41–46, Banff, 2013. ACM.

[16] M. El Mezouar, F. Zhang, and Y. Zou. Are Tweets Useful in the Bug Fixing Process? An Empirical Study on Firefox and Chrome. *Empirical Software Engineering*, (Published online in November):1–39, 2017.

[17] A. L. F. Facin, L. A. de Vasconcelos Gomes, M. de Mesquita Spinola, and M. S. Salerno. The Evolution of the Platform Concept: A Systematic Review. *IEEE Transactions on Engineering Management*, 63(4):475–488, 2016.

[18] M. Fang and M. Hafiz. Discovering Buffer Overflow Vulnerabilities in the Wild: An Empirical Study. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2014)*, pages 1–10, Torino, 2014. ACM.

[19] M. Finifter, D. Akhawe, and D. Wagner. An Empirical Study of Vulnerability Reward Programs. In *Proceedings of the 22nd USENIX Security Symposium*, pages 273–288, Washington, 2013. USENIX.

[20] M. Finifter and D. Wagner. Exploring the Relationship Between Web Application Development Tools and Security. In *Proceedings of the 2nd USENIX Conference on Web Application Development*, San Francisco, 2011. USENIX. Available online in May 2018: http://www.usenix.org/events/webapps11/tech/final_files/Finifter.pdf.

[21] E. Foudil and Y. Shafranovich. A Method for Web Security Policies. Internet-Draft, Internet Engineering Task Force (IETF). Available online in May: https://tools.ietf.org/html/draft-foudil-securitytxt-03, 2018.

[22] H. Fryer and E. Simperl. Web Science Challenges in Researching Bug Bounties. In *Proceedings of the 2017 ACM on Web Science Conference (WebSci 2017)*, pages 273–277, New York, 2017. ACM.

[23] R. Gao, Y. Wang, Y. Feng, Z. Chen, and W. E. Wong. Successes, Challenges, and Rethinking – An Industrial Investigation on Crowdsourced Mobile Application Testing. *Empirical Software Engineering*, (Published online in May 2018):1–25, 2018.

[24] F. Garćia, O. Pedreira, M. Piattini, A. Cerdeira-Pena, and M. Penabad. A Framework for Gamification in Software Engineering. *Journal of Systems and Software*, 132(21–40), 2017.

[25] C. S. Gillespie. Fitting Heavy Tailed Distributions: The poweRlaw Package. *Journal of Statistical Software*, 64(2):1–16, 2015.

[26] S. Gupta and B. B. Gupta. Cross-Site Scripting (XSS) Attacks and Defense Mechanisms: Classification and State-of-the-Art. *International Journal of System Assurance Engineering and Management*, 8(Suppl. 1):512–530, 2017.

[27] HackerOne. What Is Disclosure Assistance, and How Does It Work? Available online in September 2017: https://support.hackerone.com/hc/en-us/articles/115001936043-What-is-Disclosure-Assistance-and-how-does-it-work-.

[28] HackerOne. What Is HackerOne's Bug Bounty Service Fee? Available online in September 2017: https://support.hackerone.com/hc/en-us/articles/204951979-What-is-HackerOne-s-bug-bounty-service-fee-, 2017.

[29] HackerOne. The 2018 Hacker Report. Available online in January 2018: https://www.hackerone.com/blog/2018-Hacker-Report, 2018.

[30] N. Haile and J. Altmann. Value Creation in Software Service Platforms. *Future Generation Computer Systems*, 55:495–509, 2016.

[31] H. Hata, M. Guo, and M. A. Babar. Understanding the Heterogeneity of Contributors in Bug Bounty Programs. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2017)*, pages 223–228, Markham, 2017. ACM.

[32] T. J. Holt. Examining the Role of Technology in the Formation of Deviant Subcultures. *Social Science Computer Review*, 28(4):466–481, 2010.

[33] A. D. Householder, G. Wassermann, A. Manion, and C. King. The CERT® Guide to Coordinated Vulnerability Disclosure. Special Report, CMU/SEI-2017-SR-022, CERT Division, Carnegie Mellon University. Available online in August 2017: https://resources.sei.cmu.edu/asset_files/SpecialReport/2017_003_001_503340.pdf, 2017.

[34] C. Huang, J. Liu, Y. Fang, and Z. Zuo. A Study on Web Security Incidents in China by Analyzing Vulnerability Disclosure Platforms. *Computers & Security*, 58:47–62, 2016.

[35] A.-M. Jo. The Effect of Competition Intensity on Software Security – An Empirical Analysis of Security Patch Release on the Web Browser Market. In *Proceedings of the 16th Annual Workshop on the Economics of Information Security (WEIS 2017)*, San Diego, 2017. Available online in January 2018: http://weis2017.econinfosec.org/wp-content/uploads/sites/3/2017/05/WEIS_2017_paper_10.pdf.

[36] T. Kanda, M. Guo, H. Hata, and K. Matsumoto. Towards Understanding an Open-Source Bounty: Analysis of Bountysource. In *Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER 2017)*, pages 577–578, Klagenfurt, 2017. IEEE.

[37] K. Kannan and R. Telang. Market for Software Vulnerabilities? Think Again. *Management Science*, 51(5):726–740, 2005.

[38] M. L. Katz and C. Shapiro. Network Externalities, Competition, and Compatibility. *The American Economic Review*, 75(3):424–440, 1985.

[39] A. Kuehn and M. Mueller. Analyzing Bug Bounty Programs: An Institutional Perspective on the Economics of Software Vulnerabilities. In *Proceedings of the 42nd Research Conference on Communication, Information and Internet Policy (TRPC 2014)*, pages 1–16, Arlington Virginia, 2014. Available online in September 2017: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2418812.

[40] D. R. Kuhn, M. S. Raunak, and R. Kacker. An Analysis of Vulnerability Trends, 2008-2016. In *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C 2017)*, pages 587–588, Prague, 2017. IEEE.

[41] W. M. W. Lam. Attack-Prevention and Damage-Control Investments in Cybersecurity. *Information Economics and Policy*, 37:42–51, 2016.

[42] A. Laszka, M. Zhao, and J. Grossklags. Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms. In I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, editors, *Proceedings of the European Symposium on Research in Computer Security (ESORICS 2016), Lecture Notes in Computer Science (Volume 9879)*, pages 161–178, Heraklion, 2016. Springer.

[43] S. Lekies, B. Stock, and M. Johns. 25 Million Flows Later: Large-Scale Detection of DOM-Based XSS. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS 2013)*, pages 1193–1204, Berlin, 2013. ACM.

[44] Y. Li, C.-H. Tan, and H.-H. Teo. Leadership Characteristics and Developers' Motivation in Open Source Software Development. *Information & Management*, 49(5):257–267, 2012.

[45] Y.-F. Li, P. K. Das, and D. L. Dowe. Two Decades of Web Application Testing—A Survey of Recent Advances. *Information Systems*, 43:20–54, 2014.

[46] T. Maillart, M. Zhao, J. Grossklags, and J. Chuang. Given Enough Eyeballs, All Bugs Are Shallow? Revisiting Eric Raymond with Bug Bounty Programs. *Journal of Cybersecurity*, 3(2):81–90, 2017.

[47] D. McKinney. Vulnerability Bazaar. *IEEE Security & Privacy*, 5(6):69–73, 2007.

[48] D. McKinney. New Hurdles for Vulnerability Disclosure. *IEEE Security & Privacy*, 6(2):76–78, 2008.

[49] R. Méndez-Durón and C. E. García. Returns from Social Capital in Open Source Software Networks. *Journal of Evolutionary Economics*, 19(2):277–295, 2009.

[50] C. Miller. The Legitimate Vulnerability Market: Inside the Secretive World of 0-Day Exploit Sales. In *Proceedings of the 6th Workshop on the Economics of Information Security (WEIS 2007)*, Pittsburgh, 2007. Available online in January 2018: https://www.econinfosec.org/archive/weis2007/papers/29.pdf.

[51] B. E. Mullins, T. H. Lacey, R. F. Mills, J. E. Trechter, and S. D. Bass. How the Cyber Defense Exercise Shaped an Information-Assurance Curriculum. *IEEE Security & Privacy*, 5(5):40–49, 2007.

[52] M. Nakaya, M. Okawa, M. Nakajima, and H. Tominaga. A Support Environment and a Trial Practice of Hacking Contest with Attack and Defense Style on a Game Website. In *Proceedings of the 21st International Conference Information Visualisation (IV 2017)*, pages 360–365, London, 2017. IEEE.

[53] Nokia Corporation. Responsible Disclosure: Nokia Networks Position on Responsible Vulnerability Disclosure. Available online in September 2017: https://networks.nokia.com/responsible-disclosure, 2017.

[54] J. Olaisen and O. Revang. Working Smarter and Greener: Collaborative Knowledge Sharing in Virtual Global Project Teams. *International Journal of Information Management*, 37(1):1441–1448, 2017.

[55] J. Ondrus, A. Gannamanen, and K. Lyytinen. The Impact of Openness on the Market Potential of Multi-Sided Platforms: A Case Study of Mobile Payment Platforms. *Journal of Information Technology*, 30(3):260–275, 2015.

[56] Open Bug Bounty. https://www.openbugbounty.org/. 2017.

[57] A. Ozment. Improving Vulnerability Discovery Models: Problems with Definitions and Assumptions. In *Proceedings of the 2007 ACM Workshop on Quality of Protection (QoP 2007)*, pages 6–11, Alexandria, 2007. ACM.

[58] S. Ransbotham, S. Mitra, and J. Ramsey. Are Markets for Vulnerabilities Effective? *MIS Quarterly*, 36(1):43–64, 2012.

[59] C. Riley. Help Make Open Source Secure. The Mozilla Blog: Dispatches from the Internet Frontier. Available online in September 2017: https://blog.mozilla.org/blog/2016/06/09/help-make-open-source-secure/, 2016.

[60] T. Ring. Why Bug Hunters Are Coming in from the Wild. *Computer Fraud & Security*, (2):16–20, 2014.

[61] J. Ruohonen. Classifying Web Exploits with Topic Modeling. In *Proceedings of the 28th International Workshop on Database and Expert Systems Applications (DEXA 2017)*, pages 93–97, Lyon, 2017. IEEE.

[62] J. Ruohonen, J. Holvitie, S. Hyrynsalmi, and V. Leppänen. Exploring the Clustering of Software Vulnerability Disclosure Notifications Across Software Vendors. In *Proceedings of the 13th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2016)*, pages 1–8, Agadir, 2016. IEEE.

[63] J. Ruohonen and S. Hyrynsalmi. Evaluating the Use of Internet Search Volumes for Time Series Modeling of Sales in the Video Game Industry. *Electronic Markets*, 27(4):351–370, 2017.

[64] J. Ruohonen, S. Hyrynsalmi, and V. Leppänen. Trading Exploits Online: A Preliminary Case Study. In *Proceedings of the IEEE Tenth International Conference on Research Challenges in Information Science (RCIS 2016)*, pages 1–12, Grenoble, 2016. IEEE.

[65] J. Ruohonen, S. Hyrynsalmi, and V. Leppänen. Modeling the Delivery of Security Advisories and CVEs. *Computer Science and Information Systems*, 14(2):537–555, 2017.

[66] S. Ruutu, T. Casey, and V. Kotovirta. Development and Competition of Digital Service Platforms: A System Dynamics Approach. *Technological Forecasting & Social Change*, 117:119–130, 2017.

[67] T. Scholte, D. Balzarotti, and E. Kirda. Have Things Changed Now? An Empirical Study on Input Validation Vulnerabilities in Web Applications. *Computers & Security*, 31(3):344–356, 2012.

[68] B. Stock, G. Pellegrino, C. Rossow, M. Johns, and M. Backes. Hey, You Have a Problem: On the Feasibility of Large-Scale Web Vulnerability Notification. In *Proceedings of the 25th USENIX Security Symposium*, pages 1015–1032, Austin, 2016. USENIX.

[69] J. M. Such, A. Gouglidis, W. Knowles, G. Misra, and A. Rashid. Information Assurance Techniques: Perceived Cost Effectiveness. *Computers & Security*, 60:117–133, 2016.

[70] S. Tajalizadehkhoob, T. van Goethem, M. Korczyński, A. Noroozian, R. Böhme, T. Moore, W. Joosen, and M. van Eeten. Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, pages 553–567, Dallas, 2017. ACM.

[71] J. Uchil. Commerce Survey: Cyber Researchers Fear Legal Repercussions. The Hill. Available online in May 2018: http://thehill.com/policy/cybersecurity/310524-commerce-survey-cyber-researchers-fear-legal-repercussions, 2016.

[72] J. Uddin, R. Ghazali, M. M. Deris, R. Naseem, and H. Shah. A Survey on Bug Prioritization. *Artificial Intelligence Review*, 47(2):145–180, 2017.

[73] A. van den Bosch, T. Bogers, and M. de Kunder. Estimating Search Engine Index Size Variability: A 9-Year Longitudinal Study. *Scientometrics*, 107(2):839–856, 2016.

[74] M. van Eeten. Patching Security Governance: An Empirical View of Emergent Governance Mechanisms. *Digital Policy, Regulation and Governance*, 19(6):429–448, 2017.

[75] S. Voigt and O. Hinz. Network Effects in Two-Sided Markets: Why a 50/50 User Split is not Necessarily Revenue Optimal. *Business Research*, 8(1):139–170, 2015.

[76] M. J. Wolf and N. Fresco. Ethics of the Software Vulnerabilities and Exploits Market. *The Information Society: An International Journal*, 32(4):269–279, 2016.

[77] M. Zalewski. Going Beyond Vulnerability Rewards. Google Security Blog. Available online in September 2017: https://security.googleblog.com/2013/10/going-beyond-vulnerability-rewards.html, 2013.

[78] X. Zhang, S. Liu, Z. Deng, and X. Chen. Knowledge Sharing Motivations in Online Health Communities: A Comparative Study of Health Professionals and Normal Users. *Computers in Human Behavior*, 75:797–810, 2017.

[79] M. Zhao, J. Grossklags, and K. Chen. An Exploratory Study of White Hat Behaviors in a Web Vulnerability Disclosure Program. In *Proceedings of the 2014 ACM Workshop on Security Information Workers (SIW 2014)*, pages 51–58, Scottsdale, 2014. ACM.

[80] M. Zhao, J. Grossklags, and P. Liu. An Empirical Study of Web Vulnerability Discovery Ecosystems. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 1105–1117, Denver, 2015. ACM.

[81] M. Zhao, A. Laszka, and J. Grossklags. Devising Effective Policies for Bug-Bounty Platforms and Security Vulnerability Discovery. *Journal of Information Policy*, 7:372–418, 2017.