Crowdsourcing for Security Vulnerabilities

This paper will examine the role of crowdsourcing as innovation in security vulnerability discovery. Crowdsourcing has been applied to a multitude of causes, from new chip flavours[6], [7], through to charitable funding[8]. The niche discussed here is its application to security vulnerability discovery. In this model, the discovery of security vulnerabilities in products (software or websites) are crowdsourced out under strict conditions, with found bugs attracting a reward.[9] Two examples are discussed, one an internally managed crowdsource (Google), one a company that manages crowdsourcing for other companies (HackerOne). Lastly, it compares the two within the crowdsourcing literature and how they fit into the wider innovation system.

What is Crowdsourcing?

To the unfamiliar, crowdsourcing can be introduced conceptually as: “Outsourcing innovation problem solving (including scientific problems) via an open call to external organizations and individuals to submit ideas” (Chesbrough and Brunswicker [1])

Estellés-Arolas and González-Ladrón-de-Guevara[2] developed a detailed eight point definition involving:

- a clearly defined crowd(a) and a clearly identified crowdsourcer(d),
- an ‘open call’ inviting participation(g)
- a clear task(b) that can be assigned online(f), and conducted over the internet(h)
- providing mutual benefit between the crowd member(c) and the crowd sourcer(e)

Brabham[3] expressly calls out the sponsorship of an organisation, (the crowdsourcer who does the top-down management of the process) as the factor distinguishing crowdsourcing from other collaborative online endeavours. For example, 'commons based peer production'[4] such as editing Wikipedia, and open source software development is peer organised (bottom-up) and is not crowdsourcing.

Why Crowdsource?

Brabham[3] notes that crowdsourcing is a type of distributed intelligence that leverages the scale of the participation on the internet – ie using more brains should produce more solutions.

Afuah & Tucci[5] explain further using behavioural economics: the benefit is one of local searching (cognitively familiar problems where the agent has expertise) vs distant searching (cognitively unfamiliar and intensive problems). When seeking ‘distant’ solutions agents do not know if the solution they find is optimal, and would need to expend increasing effort to research further solutions. By casting the question widely via crowdsourcing, certain crowd members will consider the request ‘local’ because they have expertise, and submit their solution. With lower net effort the best solution can be found by assessing the group’s responses.

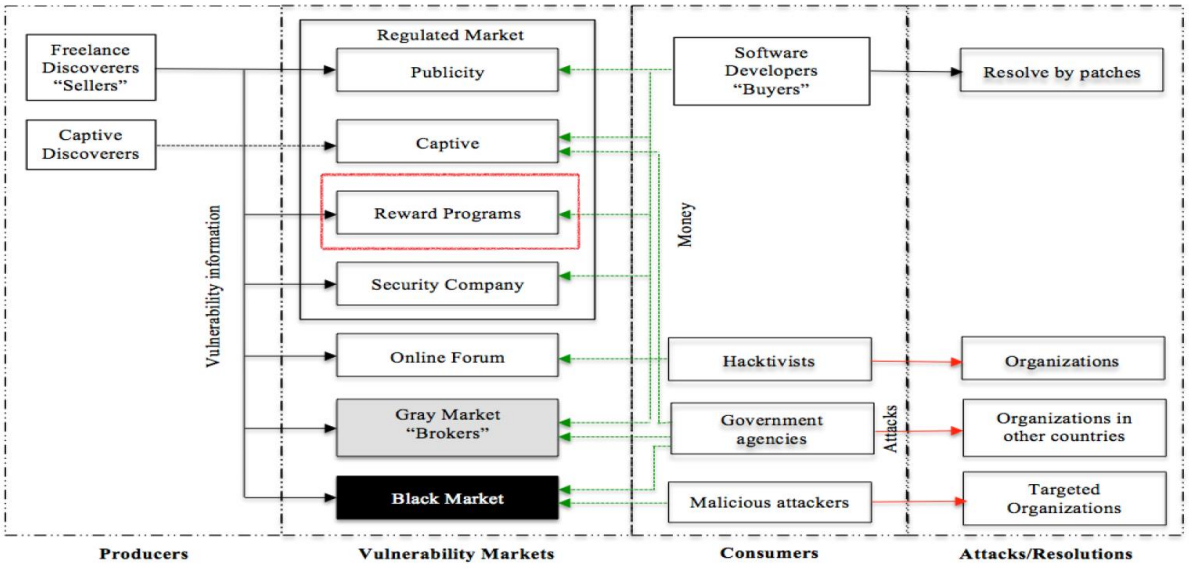
Vulnerability Discovery Options

Organisations are expected to conduct security testing on their own systems and products to find vulnerabilities or bugs for repair[10]. There are a few options which can be arranged on a matrix by internal/external, and crowdsourced/designated supplier (see table). When crowdsourcing the bug finding, the program is often called a 'bug bounty program' or a 'vulnerability reward program'[11] [12].

	Designated supply	**Crowdsourced**
Internal processing	Internal security team	An internally managed bug bounty – eg Chromium
External processing	An outsourced dedicated team - eg a 'pentest'	An externally managed bug bounty supplier – eg HackerOne

Crowdsourced bug bounties are crowdsourcing of the type 'broadcast search' as defined by Brabham[3]. Because finding software bugs is 'recognized as a challenging computational problem that often requires human intelligence'[11], they are looking for the expertise of a group that exist *out there* but in small numbers. By providing a well-defined set of challenges (and building on the local/distant paradigm) these people will respond to the call if it is sufficiently local (ie easy, given their expertise), and if they are sufficiently motivated.

Regarding motivation, the crowd is comprised of the type of security researcher called ethical hackers or 'white hats' who agree to follow the bug bounty guidelines in return for a reward[10], [13]. Rewards include recognition (such as the EFF Hall of Fame) [14], or financial (such as USD$250,000 from Microsoft) [15], 'swag' (eg t-shirts, gift cards), or some combinations[16]. This reward is critical, as explained by Algarni[13], as there is a software vulnerability market in which bug bounties ('reward programs') make up a large part of the 'regulated' part of that market, and the reward and recognition retain the vulnerability disclosures for the vendor as opposed to the unregulated market.



Emphasis around 'Reward Programs' done by author. From Figure 1 in Algarni[13 pg 281]

Most importantly, crowdsourcing to find bugs works. Edmundson[17] showed that no one code reviewer can find all bugs, but the probability of any 15 reviewers finding all the vulnerabilities is 95%.

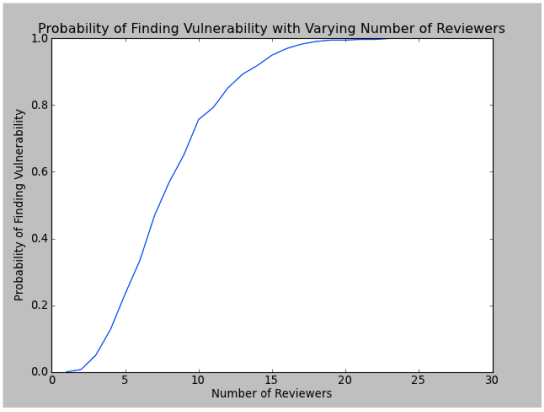


Figure 4 in Edmundson[17 pg 207]

Crowdsourced yet Internally Processed – Google

Since 2010 Google runs a few different bug bounties on its software and services [18]–[20]. Google runs these programs themselves, which means they review and fix the bugs internally, after the crowd finds and submits them.

Muniah[10] analyses the bug bounty run on the Chromium Project (the open source browser and operating system which diverged into the Chrome browser and operating system [21]). If the vulnerability was reported by a bug bounty participant they are due a reward of up to USD$100,000[18] whereas Google employees are not extrinsically rewarded beyond their standard remuneration.

The crowdsourcing program has had a positive result on finding and fixing bugs, though the percentage varies over time between internal and external finders. Muniah[10] and Finifter[22] both found that most bugs were fixed internally (~70%). However, Algarni & Malaiya[13] and Mussa[23] showed ~60% outside found. Another cause of discrepancy could be taxonomic, as each researcher cleaned the dataset with different parameters.

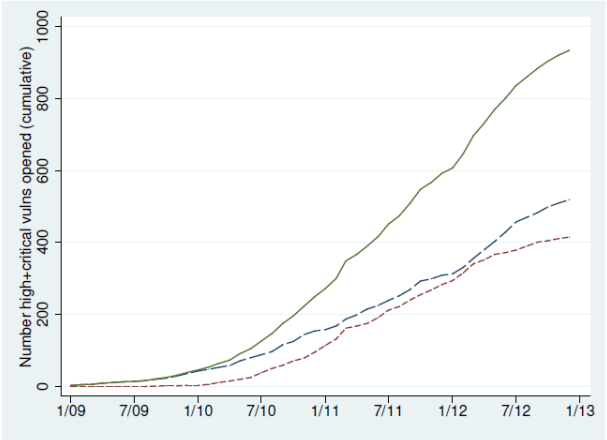
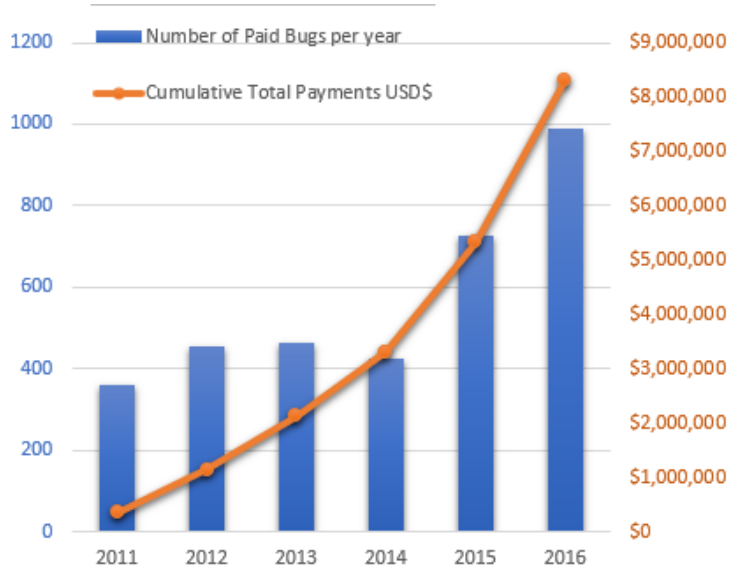


Figure 2: Number of high- plus critical-severity vulnerabilities reported over time, discovered internally (blue long-dashed line), externally (red short-dashed line), and total (green solid line). Finifter[22]

However, there is still an overhead issue for Google. Zhao[24] finds that the invalidity rate for Google VRP is 95%, whilst the duplicate rate is 8%. Zhao notes that one of the ways to keep duplicate reports low is to fix the bug quickly, of which Google originally aimed for 60 days[25], but now aims for 30 days[26] to as low as 7 days for a bug with evidence of current exploitation[27]. Finifter[22] also notes that Google paid 32% of its bug bounty total for bugs that were not in a stable release, and may never have been widely used.

Regarding the cost-benefit, Finifter[22] concludes that the bug bounty programs for Google are cost effective because the program found more bugs than even the top internal researcher for the cost of one researcher. In 2016 the program paid out for 988 bugs, and over the whole program it has paid USD$8.3M in bounties[28], multiples more than Finifter's analysis. However, this still represents a small cost to the institution given the potential benefits.



Comparison of bug submission paid out each year, and total cost of program cumulative year on year.
Produced by author from Google dataset [28]

Crowdsourced and Externally Managed – HackerOne

HackerOne is a third party bug bounty company which means it manages bug bounties for other companies. Different to the Google bounties, the HackerOne crowd submits the bug to HackerOne, then HackerOne deals with reviewing the bug, before turning over the validated bugs to the client, and paying the crowd. Clients include Twitter, AirBnB, and the US Department of Defense[29].

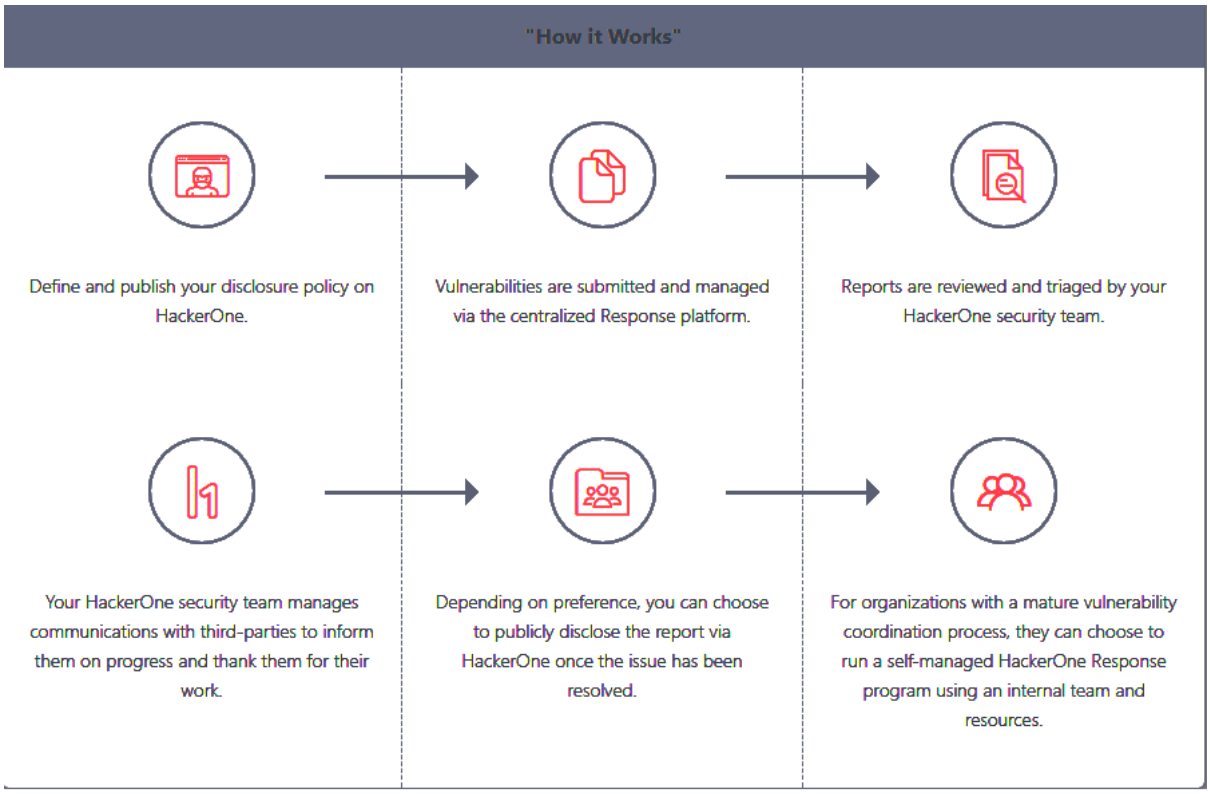
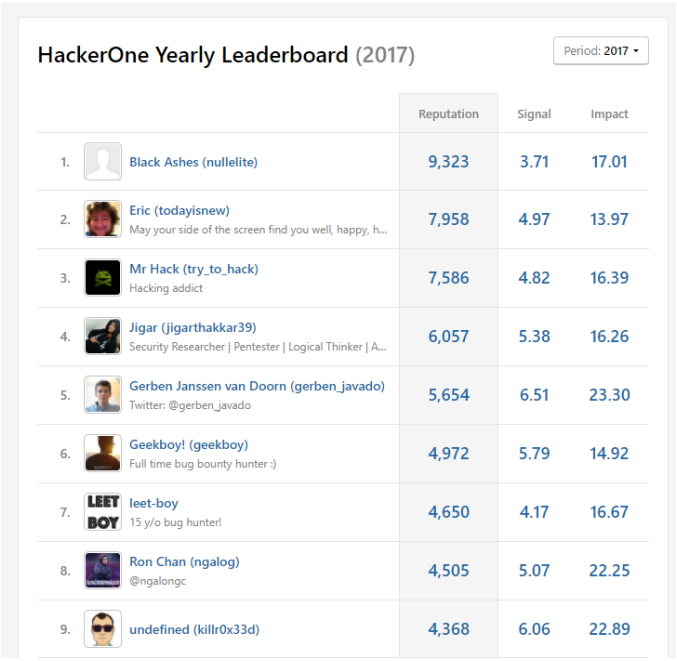


Illustration of how HackerOne works. Image contrast increased by author. From HackerOne[29]

At the time of analysis by Zhao[30], HackerOne had 1653 white-hat hackers in their crowd that have made at least one valid bug submission, 99 publicly known bug bounties run, and USD$3.6M in bug bounty payouts. From this, nearly 11,000 vulnerabilities were reported.

Zhao[24], [30] assesses the responsiveness of HackerOne to its crowd noting a median response time of only a few hours, and a triage time of less than a day. 75% of all reported bugs are fixed within 25 days, with a median time of under 7 days. However, this data could only be assessed on the approximately 7% of HackerOne bugs that are publicly disclosed, and he warns that they could be biased in favour of those organisations that are efficient and experienced.

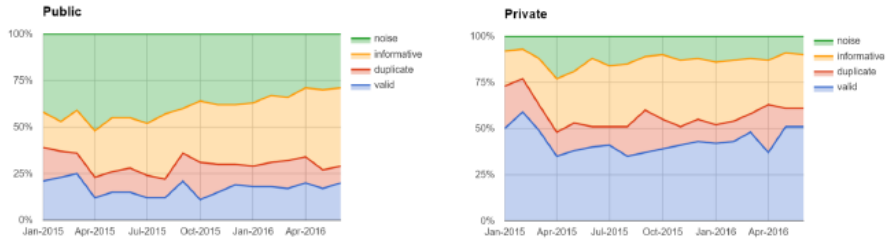


An example of a HackerOne leaderboard - this one for the running totals in 2017. From HackerOne[31]

HackerOne also taps into the recognition motivation[11], with leaderboards, profiles and badges that the crowd can use to promote themselves and their skills eg [31], [32]. Many of the clients of HackerOne do not permit disclosure of the details bug publicly by the finder[30], leaving the leaderboards as the proof of their skill in vulnerability discovery. Publicity and promotion is one of the mechanisms of switching from freelance bug finding into a career[11], [22], so this is an important tool for HackerOne to implement to attract and motivate the crowd.

Maillart[11] reviews the HackerOne programs as part of an examination of the economic motivations and diminishing returns. They note that there is a peak of activity within weeks of the initial launch, then the bounty rewards decay, meaning the majority of HackerOne bugs are found in the first few weeks. Their analysis of the crowdsourced vulnerability market showed that bug bounty crowd members have the following incentives: to search for bugs on a wide variety of programs, to switch to new programs when they are launched (to get the easier bugs first), and especially to switch if the monetary reward in the new program is higher.

Zhao[24] elaborates further that, like the Google program, HackerOne has challenges with invalid reports (77%) and duplicates (13%+). However, because the crowd is registered and assessed this is less of a problem. Further, some bounties are private - only opened to crowd members that are a certain quality or higher - creating a valid bug report rating of 50%[24]. However, this has the disadvantage of diminishing the collective intelligence effect of crowdsourcing – it is cutting out a proportion of the crowd that, though inefficient, do still find bugs[30], [33].



(a) Public (b) Private

Figure 6.2: Trend of report types for public programs and private programs on HackerOne. In our paper, we consider noise and informative reports as invalid reports.

From Zhao[24 pg 104]

Analysis

Despite one being self-managed and the other being a managed service, there is no significant difference between the use of crowdsourcing between Google and HackerOne except for the latter's sometime use of a restricted crowd. However this has emerged as a very serious point of discussion: Laszka[9] cautions that restrictive policies can be a deterrent to participation, and less researchers means less bugs found[11]. Hypothetically, should the 'crowd' get too restrictive it is arguably not even 'crowdsourcing' by definition.

Case Studies Meet Definition?

Bug bounties meet all the definitions of crowdsourcing[12]: they are online, they have clearly defined tasks set out by a sponsoring entity, for a clearly defined crowd, who receive 'reward' or 'bounty' (typically either recognition or monetary), in return for verified bugs found.

Because of the relative maturity of the concept of crowdsourcing, further refinement of the concept has occurred for specific applications such as software. LaToza's[34] work on software can be even more finely applied to vulnerability discovery. He takes some of the characteristics of previous definitions for granted, perhaps indicating the process has achieved a type of design dominance, enabling granular differences to rise to the surface.

The dimensions of crowdsourcing.

Dimension	Explanation	Scale
Crowd size	The size of the crowd necessary to effectively tackle the problem	Small to large
Task length	The amount of time a worker spends completing a task	Minutes to weeks
Expertise demands	The level of domain familiarity required for a worker to contribute	Minimal to extensive
Locus of control	Ownership of the creation of tasks or subtasks	Client to workers
Incentives	The factors motivating workers to engage with the task	Intrinsic to extrinsic
Task interdependence	The degree to which tasks in the overall workflow build on each other	Low to high
Task context	The amount of system information a worker must know to contribute	None to extensive
Replication	The number of times the same task might be redundantly completed	None to many

Dimensions of crowdsourcing for software purposes. From Table 1 in LaToza & van der Hoek [31 pg 76]

Combining both the broad and specific parameters for crowdsourcing from Brabham[3] and LaToza[34]:

Parameter	Definer	Google	HackerOne
Defined crowd	Estellés-Arolas	*Yes – security researchers*	*Yes – registered security researchers*
Crowd size	LaToza	1000s+	
Expertise demands	LaToza	Experience in computer security research – low to high level	
Identified crowdsourcer	Estellés-Arolas	Google	HackerOne's clients via HackerOne
Locus of control	LaToza	Client	
'open call'	Estellés-Arolas	*Yes – anyone could participate*	*Sometimes – some calls are restricted*
Clear task	Estellés-Arolas	Yes	
Task length	LaToza	Hours to weeks	
Task interdependence	LaToza	Low	
Task context	LaToza	Low to medium	
Replication	LaToza	High	
Assigned online/ conducted over the internet	Estellés-Arolas	Yes	
Mutual benefit	Estellés-Arolas	*Financial and recognition*	*Financial, and/or recognition*
Incentives	LaToza		

Intersection with Other Innovation Concepts

Product Platforms and Platform Ecosystem

The HackerOne model is also a type of platform. As the conduit(platform owner) between their client(consumer) and the crowd(producer), it connects multiple 3rd party companies to the security researchers with the same format for each bounty and provides overarching community engagement[35]. The managed bounty product puts HackerOne as an intermediary (which is perhaps not a standard platform[36]), however, HackerOne also provides a mechanism for selling only their infrastructure, enabling companies to use the platform as a base for their own bug bounty maintenance.

Open-source

HackerOne began as a company providing bounties on the open source software that is the foundation of the internet[37] and still offers these bounties sponsored by large technology companies[38]. Further vulnerability discovery is enhanced by the target software being open source (such as Chromium[21]).

Web APIS

As well as crowdsourced vulnerability research into APIs improving the sense of security and reliability of these services, HackerOne has an API for customers to speed up their reporting and feedback[39].

Releasing Data Sets

Much of the analysis in the literature would have been deeply hindered by the data being restricted. Google's provision of data through the open source Chromium project provided useful data on issues such as time to fix, invalids and duplicates[10], [22], [24]. Though only a portion of HackerOne's data is available, researchers felt confident enough to use it for insights like crowd utility and motivation[11], [30].

Conclusion

Crowdsourcing for vulnerability discovery is widespread and accepted in the industry. The use of crowdsourcing for this purpose was an important step to take in innovation in the software industry, as it has supported many of the intersecting innovations relied upon by the internet.

Questions regarding overall efficacy and utility for crowdsourcing for vulnerabilities have been answered in the affirmative. However, issues such as those raised by Laszka[9], Maillart[11] and Zhao[40] involving the optimal methods, maximising utility and other intersections of economics and vulnerability research are still being examined.

Bibliography

[1] H. Chesbrough and S. Brunswicker, “A Fad or a Phenomenon? The Adoption of Open Innovation Practices in Large Firms,” *Research Technology Management*, vol. 57, no. 2, pp. 16–25, Mar. 2014.

[2] E. Estellés-Arolas and F. González-Ladrón-de-Guevara, “Towards an integrated crowdsourcing definition,” *Journal of Information Science*, vol. 38, no. 2, pp. 189–200, Apr. 2012.

[3] Daren Brabham, “Crowdsourcing: A Model for Leveraging Online Communities,” in *The Participatory Cultures Handbook*, edited by Aaron Delwiche & Jennifer Henderson, 2012, pp. 120–129.

[4] Y. Benkler, “Coase’s Penguin, or, Linux and‘ The Nature of the Firm,’” *Yale Law Journal*, pp. 369–446, 2002.

[5] A. Afuah and C. L. Tucci, “Crowdsourcing As a Solution to Distant Search,” *Academy of Management Review*, vol. 37, no. 3, pp. 355–375, Jul. 2012.

[6] S. Djelassi and I. Decoopman, “Customers’ participation in product development through crowdsourcing: Issues and implications,” *Industrial Marketing Management*, vol. 42, no. 5, pp. 683–692, Jul. 2013.

[7] C. Reading, “Lay’s ‘Do Us A Flavor’ Crowdsourcing Has Backfired Right On Cue,” *Uproxx*, 20-Feb-2017.

[8] M. J. Young and E. Scheinberg, “The Rise of Crowdfunding for Medical Care: Promises and Perils,” *JAMA*, vol. 317, no. 16, p. 1623, Apr. 2017.

[9] A. Laszka, M. Zhao, and J. Grossklags, “Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms,” in *Computer Security – ESORICS 2016*, vol. 9879, I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, Eds. Cham: Springer International Publishing, 2016, pp. 161–178.

[10] N. Munaiah, F. Camilo, W. Wigham, A. Meneely, and M. Nagappan, “Do bugs foreshadow vulnerabilities? An in-depth study of the chromium project,” *Empirical Software Engineering*, vol. 22, no. 3, pp. 1305–1347, 2017.

[11] T. Maillart, M. Zhao, J. Grossklags, and J. Chuang, “Given enough eyeballs, all bugs are shallow - Revisiting Eric Raymond with bug bounty markets,” *Revisiting Eric Raymond with bug bounty markets*, Aug. 2017.

[12] H. Fryer and E. Simperl, “Web Science Challenges in Researching Bug Bounties,” presented at the WebSci ’17, June 25-28, 2017, Troy, NY, USA, 2017, pp. 273–277.

[13] A. Algarni and Y. Malaiya, “Software vulnerability markets: Discoverers and buyers,” *International Journal of Computer, Information Science and Engineering*, vol. 8, no. 3, pp. 71–81, 2014.

[14] EFF, “Security Vulnerability Disclosure Program Hall of Fame,” n.d. [Online]. Available: https://www.eff.org/security/hall-of-fame. [Accessed: 28-Oct-2017].

[15] Microsoft, “Microsoft Hyper-V Bounty Program Terms,” *Microsoft TechNet*, 31-May-2017. [Online]. Available: https://technet.microsoft.com/en-us/mt784431.aspx. [Accessed: 29-Oct-2017].

[16] bugcrowd, “Bug Bounty List,” n.d. [Online]. Available: https://www.bugcrowd.com/bug-bounty-list/. [Accessed: 29-Oct-2017].

[17] A. Edmundson, B. Holtkamp, E. Rivera, M. Finifter, A. Mettler, and D. Wagner, “An Empirical Study on the Effectiveness of Security Code Review,” in *Engineering Secure Software and Systems: 5th International Symposium, ESSoS 2013, Paris, France, February 27 - March 1, 2013. Proceedings*, J. Jürjens, B. Livshits, and R. Scandariato, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 197–212.

[18] Google, “Chrome Reward Program Rules,” *Google Application Security*, n.d. [Online]. Available: https://www.google.com/about/appsecurity/chrome-rewards/. [Accessed: 28-Oct-2017].
[19] Google, “Google Vulnerability Reward Program (VRP) Rules,” *Google Application Security*, n.d. [Online]. Available: https://www.google.com/about/appsecurity/reward-program/. [Accessed: 28-Oct-2017].
[20] Google, “Android Security Rewards Program Rules,” *Google Application Security*, n.d. [Online]. Available: https://www.google.com/about/appsecurity/android-rewards/. [Accessed: 28-Oct-2017].
[21] Google, “The Chromium Projects,” n.d. [Online]. Available: https://www.chromium.org/. [Accessed: 28-Oct-2017].
[22] M. Finifter, D. Akhawe, and D. Wagner, “An Empirical Study of Vulnerability Rewards Programs.,” in *USENIX Security Symposium*, 2013, pp. 273–288.
[23] A. A. Y. Mussa, “Quantifying The Security Risk Of Discovering And Exploiting Software Vulnerabilities,” Colorado State University, Fort Collins, Colorado, 2016.
[24] M. Zhao, “Discovering And Mitigating Software Vulnerabilities Through Large-Scale Collaboration,” Pennsylvania State University, 2016.
[25] Google Security Team, “Rebooting Responsible Disclosure: a focus on protecting end users,” 20-Jul-2010. [Online]. Available: https://security.googleblog.com/2010/07/rebooting-responsible-disclosure-focus.html. [Accessed: 28-Oct-2017].
[26] Google, “Severity Guidelines for Security Issues,” n.d. [Online]. Available: https://chromium.googlesource.com/chromium/src/+/master/docs/security/severity-guidelines.md. [Accessed: 29-Oct-2017].
[27] Google Security Team, “Disclosure timeline for vulnerabilities under active attack,” 29-May-2013. [Online]. Available: https://security.googleblog.com/2013/05/disclosure-timeline-for-vulnerabilities.html. [Accessed: 29-Oct-2017].
[28] Google, “VRP Rewards Over Time,” *DATASET*, 02-Mar-2017. [Online]. Available: https://docs.google.com/spreadsheets/d/tF49QiAmJh9FebPhoeFtLtg/edit#gid=0. [Accessed: 29-Oct-2017].
[29] HackerOne, “HackerOne Response,” *HackerOne*, n.d. [Online]. Available: https://www.hackerone.com/product/response. [Accessed: 29-Oct-2017].
[30] M. Zhao, J. Grossklags, and P. Liu, “An Empirical Study of Web Vulnerability Discovery Ecosystems,” in *CCS’15, October 12–16, 2015, Denver, Colorado, USA.*, 2015, pp. 1105–1117.
[31] HackerOne, “HackerOne Yearly Leaderboard (2017),” *HackerOne*, n.d. [Online]. Available: https://hackerone.com/leaderboard/2017. [Accessed: 29-Oct-2017].
[32] HackerOne, “Mathias Karlsson (avlidienbrunn),” *HackerOne*, n.d. [Online]. Available: https://hackerone.com/avlidienbrunn?sort_type=latest_disclosable_activity_at&filter=type%3Aall%20from%3Aavlidienbrunn&page=1&range=forever. [Accessed: 29-Oct-2017].
[33] M. Zhao, J. Grossklags, and K. Chen, “An Exploratory Study of White Hat Behaviors in a Web Vulnerability Disclosure Program,” in *SIW’14, November 7, 2014, Scottsdale, Arizona, USA*, 2014, pp. 51–58.
[34] T. D. LaToza and A. van der Hoek, “Crowdsourcing in Software Engineering: Models, Motivations, and Challenges,” *IEEE Software*, vol. 33, no. 1, pp. 74–80, Jan. 2016.
[35] M. Alstyne, G. Parker, and S. Choudry, “Pipelines, Platforms, and the New Rules of Strategy.,” *Harvard Business Review*, vol. 94, no. 4, pp. 54–62, 2016.
[36] F. Zhu and N. Furr, “Products to Platforms: Making the Leap.,” *Harvard Business Review*, vol. 94, no. 4, pp. 72–78, Apr. 2016.
[37] Dan Goodin, “Now there’s a bug bounty program for the whole Internet,” *ARS Technica*, 07-Nov-2013.
[38] HackerOne, “The Internet Bug Bounty,” n.d. [Online]. Available: https://www.hackerone.com/internet-bug-bounty. [Accessed: 29-Oct-2017].

[39] HackerOne, “Announcing the HackerOne API,” 01-Jun-2016. [Online]. Available: https://www.hackerone.com/blog/launching-the-hackerone-api. [Accessed: 29-Oct-2017].
[40] M. Zhao, A. Laszka, T. Maillart, and J. Grossklags, “Crowdsourced Security Vulnerability Discovery: Modeling and Organizing Bug-Bounty Programs,” in *HCOMP Workshop on Mathematical Foundations of Human Computation*, 2016.