Productivity and Patterns of Activity in Bug Bounty Programs: Analysis of HackerOne and Google Vulnerability Research

Conference Paper · August 2019

DOI: 10.1145/3339252.3341495

Luca Allodi
Università degli Studi di Trento
40 PUBLICATIONS **344** CITATIONS

Marco Cremonini
University of Milan
88 PUBLICATIONS **1,459** CITATIONS

Productivity and Patterns of Activity in Bug Bounty Programs: Analysis of HackerOne and Google Vulnerability Research

Donatello Luna
Tribunale di Busto Arsizio
Busto Arsizio, Varese, Italy
donatello@luna.it

Luca Allodi
Eindhoven University of Technology
Eindhoven, Netherlands
l.allodi@tue.nl

Marco Cremonini
University of Milan
Milan, Italy
marco.cremonini@unimi.it

ABSTRACT

In this work, we considered two well-known bug bounty programs - HackerOne and Google Vulnerability Research - with the goal of investigating patterns of activity and comparing productivity of security researchers. HackerOne and Google's programs differ in many ways. HackerOne is one of the largest and most successful bug bounty programs, with heterogeneous membership of security researchers and software producers. Google Vulnerability Research, instead, is a closed program for selected Google employees working on a more homogeneous range of software. For the analysis, we introduced three productivity metrics, which let us study the performance of researchers under different perspectives and possible patterns of activity. A contribution of this work is to shed new light on the yet not well understood environment represented by bug bounties and software vulnerability discovery initiatives. The low-hanging fruits approach adopted by unexperienced researchers in open bug bounties has been often discussed, but less is known about the approach adopted by more experienced participants. Another result is to have shown that a generic comparison between different bug bounty programs may lead to wrong conclusions. Bug bounty programs could exhibits large variations in researcher profiles and software characteristics, which make them not comparable without a careful examination of homogeneous subsets of participants and incentive mechanisms.

CCS CONCEPTS

• **Security and privacy → Vulnerability management**; **Software and application security**; **Economics of security and privacy**; *Human and societal aspects of security and privacy.*

KEYWORDS

bug bounty programs, software vulnerability, researchers' productivity

ACM Reference Format:
Donatello Luna, Luca Allodi, and Marco Cremonini. 2019. Productivity and Patterns of Activity in Bug Bounty Programs: Analysis of HackerOne and Google Vulnerability Research. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3339252.3341495

1 INTRODUCTION

Bug bounty programs have become a popular initiative in cybersecurity, not just limited to tech companies willing to have a controlled vulnerability discovery program for their software products [15, 19, 24]. Independent bug bounty platforms have appeared, hosting and providing supporting tools for a number of companies that offer monetary rewards to participants, in exchange of software vulnerabilities. Finally, also public institutions have found convenient to organize bug bounties as a way to improve the security of their systems [2, 3].

In this work, we focused on two relevant case studies: the HackerOne Bug Bounty Platform [10] and the Google Vulnerability Research Program [8]. These two initiatives, which present relevant differences, discussed in the following sections, offer open data regarding part of their bug bounty activities [25]. We have analyzed the two datasets, as they appeared publicly until February 2018, and made quantitative and qualitative analyses of the vulnerability research and discovery process[1]. With respect to the goal of our work, first, we studied to which extent the two bug bounty programs are comparable, and from that what could be inferred for a better knowledge of bug bounties. Next, we were interested to know how security researchers work, in particular if they have patterns of activity and follow strategies to improve their productivity. Then, from researchers we moved to consider software and tools, representing the targets of bug bounties. Different software may lead to different chance to find vulnerabilities, and the different software availability in the two programs may motivate researchers to adopt specific strategies [14]. A better understanding of these aspects, we believe, is needed for a better evaluation of the claimed benefits for cybersecurity of bug bounty initiatives.

The paper is organized as follow: a Background section presents the scenario of vulnerability discovery. Section 3 introduces the two case studies of HackerOne and Google, followed by Section 4 where the two datasets are analyzed with respect to several parameters. In Section 5, we define our three productivity metrics and use them to discuss patterns of activity of homogeneous groups of researchers. Finally, some conclusions are drawn.

2 BACKGROUND

The quest of defining a policy for software vulnerability disclosure that could be universally adopted by all organizations is a cybersecurity topic from decades. Despite the many tentatives, however,

[1]The two datasets we have collected and used for the analyses are publicly available at: https://github.com/mc-unimi/bugbounty_ARES19

still there is not a real general agreement. The best approximation of a standard procedure is to consider that security researchers practice responsible disclosure. This procedure requires that the researcher privately notifies the software vendor, whose tool is affected by the vulnerability, and together reach an agreement on the conditions for the disclosure. Standard practices for responsible disclosure have been proposed, in particular with regard to the period of time left to a vendor for releasing a patch. After the patch has been released, the researcher has the opportunity to publicly reveal full details about his findings. In this way, the vendor can patch the vulnerability before details are public and users are not put at undue risk.

A different approach is known as full disclosure. With this approach, a researcher releases immediately full details of a vulnerability to everybody, without seeking any agreement with the vendor. The rationale of full disclosure is that it gives full and timely information to all potential victims that might adopt immediate countermeasures, and pushes vendors to not delay fixes [5]. The obvious drawback is that also potential attackers are fully and timely informed of a newly discovered vulnerability, and patches released by vendors under urgency might be of uncertain quality.

Each researcher with the knowledge of a zero-days vulnerability should determine if it should be kept private and for how long. As stated by Ablon and Bogart [1], this decision generally involves understanding, firstly, the longevity and, secondly, the collision rate of the zero-day vulnerability[2].

The discussion on vulnerability disclosure has also involved some government agencies, in particular by asking if it is admissible that they develop or trade zero-day vulnerabilities, assumingly for protecting national interests, or they should be forced by law to disclose them to the affected vendors, speeding up the patching process and reducing the number of vulnerable systems exposed to threats [1]. In 2017, an official document has been published, describing a Vulnerability Equities Policy and Process (VEP) [21], which lays out the procedure used by the U.S. Government to determine whether a known zero-days vulnerability should be retained for government use or revealed to the software producer. A discussion about government agencies' strategy and motivations is out of the scope of this work, but it is important to note how many conflicting interests still exist on the path to the definition of a standard vulnerability disclosure procedure.

Currently, many organizations worldwide, both in the public and private sectors, are expressing interest in the adoption of vulnerability disclosure procedures, in order to improve their ability to fix potential bugs and flaws in internal systems, networks and infrastructures. As a proof of the importance of this approach, the U.S. Department of Justice Criminal Division's Cybersecurity Unit has publicly released a framework to assist and guide organizations in defining an appropriate vulnerability disclosure program [22]. Also the International Organization for Standardization has defined the ISO/IEC 29147:2014 [20], aiming at establishing a fully standard procedure for vulnerability disclosure in products and online services. However, as a matter of fact, a truly recognized standard and commonly adopted vulnerability disclosure procedure is still far from being the reality for most organizations.

[2] *Longevity* refers to the survival probability and expected lifetime of zero-day vulnerabilities and their exploits, while *collision rate* is the likelihood that a zero-day found by one entity will also be found independently by another

3 METHODOLOGY

It has been shown [6] that for an organization, making use of the white hat security researchers' community could result in reduced costs, compared with hiring internal security researchers or using commercial services, like penetration tests offered by specialized companies [23]. In the last few years, there has been a growing interest in public vulnerability discovery programs where vendors openly reward independent researchers for a responsible disclosure of software vulnerabilities they could possibly find. These programs, known as Vulnerability Reward Programs (VRPs) or Bug Bounty Programs, are playing an increasingly important role in cybersecurity and cyber threat assessment. The underlying assumption is that these sort of initiatives might be effective to reduce the incentives of trading vulnerabilities in the black market or to disclose vulnerabilities in an uncontrolled manner. Currently, some important vendors have deployed, with much publicity, their own bug bounty program, signaling a changed approach to the issue of vulnerability disclosure, from the opacity and often hostile response to independent security researchers of the past, to a more open and collaborative stance of the present [15]. The best example of an influential vendor that has changed its approach is probably Microsoft, which is currently organizing a number of bug bounties with different monetary rewards depending on the threat target. In this way, Microsoft offers direct payments to bug hunters in exchange of reporting certain types of vulnerabilities and exploitation techniques [17]. These programs are sometimes open only for a limited time period and only apply to preview versions of the software, in order to address vulnerabilities before the final version is released. By this approach, Microsoft involves bug bounty programs and independent security researchers directly into its software development life cycle.

3.1 Case Study: HackerOne

We first focused on the HackerOne platform as our first case study. In recent years, the HackerOne platform has attracted many important software vendors who joined the platform to start their own bug bounty program. In mid October 2017, the HackerOne platform hosted more than 908 programs that permitted researchers to find more than 55000 vulnerabilities. According to its press release, HackerOne has awarded more than $21 millions to researchers [10]. The HackerOne platform is structured to support both bug hunters and companies. It offers resources to participants, including articles, reports, case studies, and training material aiming at helping users that wish to enter the bug bounty process. It also has a section, called Program Directory, where all active programs are listed. It also provides information about eligible vulnerabilities, program exclusions, or the disclosure and reward policy. Moreover, another important section is called Hacktivity [11], it records, for each program, the interactions between researchers and the platform, enumerating every report, bounty awarded, and vulnerabilities even if not rewarded. However, information available is partial: at the time of our study, between the end of 2017 and beginning of

2018, the HackerOne platform publicly disclosed only 1.787 activities, of those rewarded with a bounty, about 13% of the full dataset, while for the remaining 87% of the dataset only few information is made available. HackerOne claims to have created a community of more than 100000 security researchers registered on the platform. Those researchers comes from 90 countries worldwide, with the majority of them coming from India and the United States. A recent paper from HackerOne reported the result of an online survey submitted to active researchers [12]. It gives interesting statistics about researchers, like that 24% of them spend more than 40 hours per week searching for vulnerabilities and 17% spend between 20 to 40 hours per week. These two groups, for the hours spent regularly could be considered as professional vulnerability researchers. We also know [9] that more than 90% of successful researchers on HackerOne are under 34 years old, 43% are between 18 and 24 and 6% are 13-17. May be not unexpectedly, there is a small user group, less than 1%, reported to be 13 or younger.

3.2 Case Study: Google Vulnerability Reward Programs

As our second case study of bug bounty program, we investigated Google's Security Reward programs, first started in 2010. By now, Google reports to have a wide community of security researchers, awarding cash rewards for high-quality security research that identify software vulnerabilities [7]. Further information has been disclosed by Eduardo Vela Nava, Google VRP Technical Lead and Master of Disaster, in a post on the official Google's Security Blog [4], where he discussed the results of bug bounty programs for the year 2016. In addition, Google is running an internal research program, strictly reserved to Google engineers and researchers, inspecting and researching vulnerabilities in different software available on the market, not limited to Google products. Example of software categories are: Operating Systems, Libraries, Cryptography, and Services. The findings of this research program are publicly available at [8]. A distinctive feature of this program is that teams of researchers are also eligible to report security vulnerabilities, being awarded as a group. We found 43 distinct groups in the dataset. Being the membership of the program limited to Google engineers and researchers, it could be assumed that the discoveries are exclusively made by high-skilled professionals working according to professional quality standards in a well-organized fashion. After some data-cleaning operations, we obtained a dataset of 939 vulnerabilities found on 28 types of software by 80 distinct researchers, working individually or in group. To deal with researchers that for some vulnerabilities worked individually and for others in a group, we have produced two datasets that we have called *GaE (Groups as Entities)* and *GaP (Groups as Parts)*. In the GaE dataset, we considered a group as a distinct entity from individual researchers, which were listed individually only for the vulnerabilities discovered by working autonomously. In the GaP dataset, instead, we only listed researchers, not groups. Therefore, we assigned a contribution to researchers for vulnerabilities found as a group. No information was provided about group members' efforts, therefore we just attributed to each researcher of a group an equal fraction of the discovered vulnerability (e.g., in a group of three researchers, we attributed to each one 1/3 of discovered vulnerability).

4 ANALYSIS

4.1 Vulnerabilities by Software

We started our analysis by focusing on software and looking at some specific pattern in researchers' working behavior. The first consideration we made is that HackerOne and Google datasets are clearly different in nature and characteristics. Figure 1 shows the vulnerabilities by software category found by researchers in the HackerOne platform. Although the vulnerability count shows clear differences between many software, only Yahoo! represents an outlier. In Figure 2, we have plotted the vulnerabilities found in Google by considering the *GaE (Groups as Entities)* dataset (Figure 2 *Left*) and the *GaP (Groups as Parts)* dataset (Figure 2 *Right*). Most vulnerabilities are concentrated in few software only: Adobe, Apple, and Browsers vulnerabilities account for a significant majority of the overall population.

4.2 Vulnerability by Researcher

Our analysis continued with a focus on researchers and on their behavior in finding vulnerabilities. In order to characterize researchers, we plotted the number of software in which each researcher has found vulnerabilities and the total number of vulnerabilities that the researcher has found. Figure 3 shows that, despite the different characteristics of the HackerOne and Google cases - density of researchers is used to make the figures comparable despite the large difference in number -, there is a similar distribution of the number of discovered vulnerabilities with respect to the number of researchers. A large majority of researchers - and groups for Google's GaE dataset - has found few vulnerabilities, with some noticeable clusters of Google researchers/groups awarded for larger findings, up to hundreds of vulnerabilities.

For the HackerOne dataset, Figure 4 shows a different aspect of researchers' activity, namely how many vulnerabilities a researcher has found with respect to the number of different software analyzed, in order to distinguish if a researcher has worked on a limited set of software or (s)he has spread the activity on a larger set. That analysis aims at studying a first possible pattern of activity related to the specialization of researchers. We will see in the following that this perspective will help us to better characterize the two case studies. In Figure 4, 30 researchers (1.27% of the HackerOne population) found more than 50 vulnerabilities and 2,043 researchers (86.56%) found less than 10 vulnerabilities each. Moreover, 1,443 researchers (61.14%) have worked on just one software, while only 75 researchers (3.18%) have worked on more then 10 software.

With respect to these figures, researchers that found few vulnerabilities (<10) and worked on few software (<5) apparently represent those less engaged in the discovery activity and not particularly relevant to study patterns of activity. We then excluded them from the analysis and point our attention to the other types of researcher.

4.3 Gini Inequality Coefficient Analysis

Our analysis continued by inspecting the Gini inequality metric [18]. This coefficient measures the inequality among values and, in our work, it has been applied to the vulnerabilities found in each software by a researcher. It can vary in a range of [0,1], where 0 expresses perfect equality in a series - i.e. each entry has the same

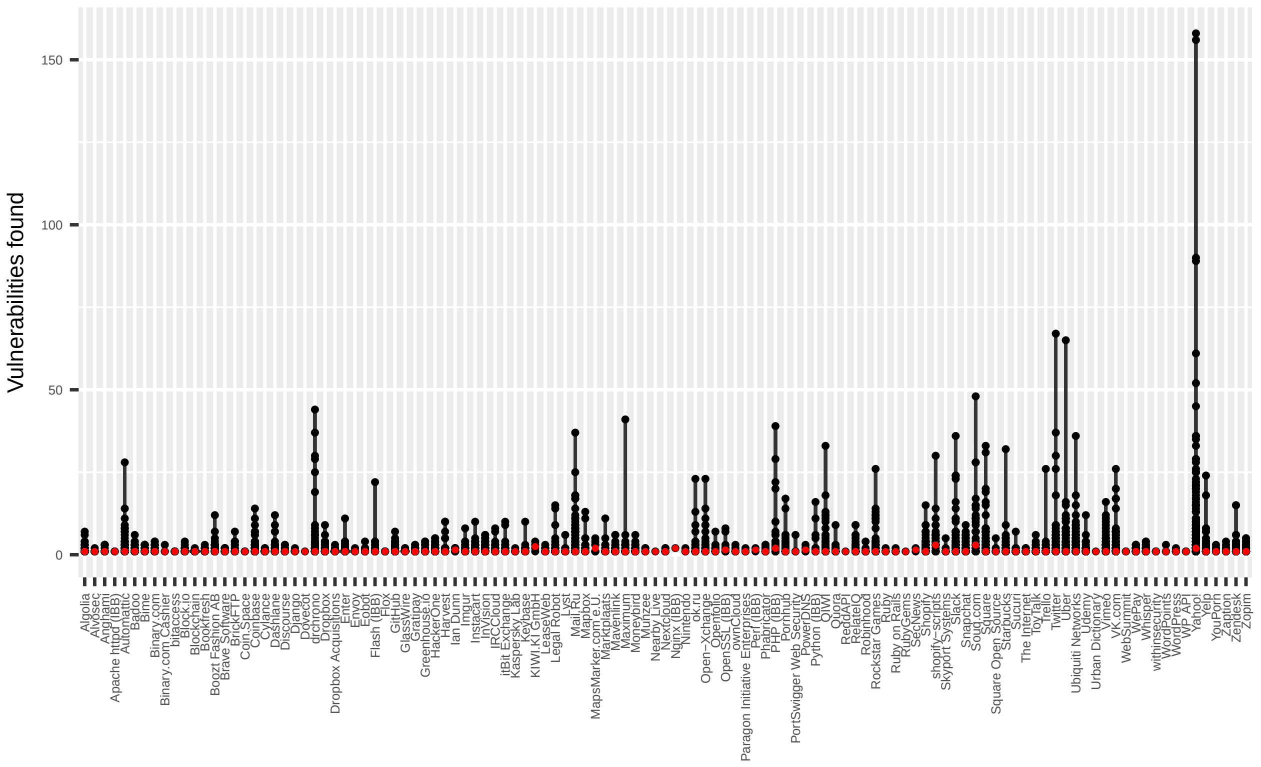

Figure 1: HackerOne: Vulnerabilities by software. The red dot are the average number of vulnerabilities found in the software.

value or there is only one entry in the series - and 1 expresses the maximal inequality among values. In our case, researchers with Gini coefficient equal to zero are those who have found *exactly the same number of vulnerabilities in any software they have worked on or who have found vulnerabilities in only one software.* On the other hand, researchers with a Gini coefficient close to one are those who have worked on many software and they had a large variance in the number of discovered vulnerabilities per software. Typically, they may have found most of their vulnerabilities in one or two software and very few vulnerabilities in a strike of other software. This observation reflects the concept of inequality among the distribution of vulnerabilities found by a researcher in all the software he has worked on. This metric has proved useful in our analysis because it permits to highlight those researchers that have a clearly different pattern of activity with respect to the whole population.

Figure 5 shows the distribution of researchers in the HackerOne dataset for Gini coefficient and number of vulnerabilities found. Analogously, Figure 6 show Google's GaE and GaP datasets. From these figures, one important thing we can observe is that there is always a noticeable group of researchers who has the Gini coefficient equal to zero.

A more detailed analysis, presented in Figure 7, has permitted to confirm that researchers with Gini coefficient equal to zero are for the most part those that have found just one vulnerability working on a single software.

4.4 The Hard-Working Researchers

To continue our study, we decided to focus on a smaller and more homogeneous set of researchers of the HackerOne dataset. In particular, we removed those researchers that have found only a few vulnerabilities on a small set of software, because we consider them as occasional researchers not comparable with Google ones. Instead, we focused on those researchers who happened to be particularly productive and eclectic, with respect to the number of vulnerabilities they found or to the number of software they worked on.

By looking at the distribution of researchers with respect to Gini coefficient values and to the number of vulnerabilities found by each researcher, we notice that there is a high dispersion rate in researchers with a high Gini coefficient or with a high number of vulnerabilities found. In Figure 8, we emphasizes a specific subset of researchers that we have called the*Hard-Working Researchers* group, characterized by high Gini coefficient (greater than 0.50), thus having an activity spread on many software, or high number of discovered vulnerability (more than 50). The color gradient in Figure 8 represents the number of software a researcher has worked on. Once defined the Hard-Working Researchers group, we further investigated researchers' behavior by looking at those particular researchers that present unusual patterns of activity in order to figure out which set of decision criteria was guiding their activity. A good example of these researchers is HackerOne researcher

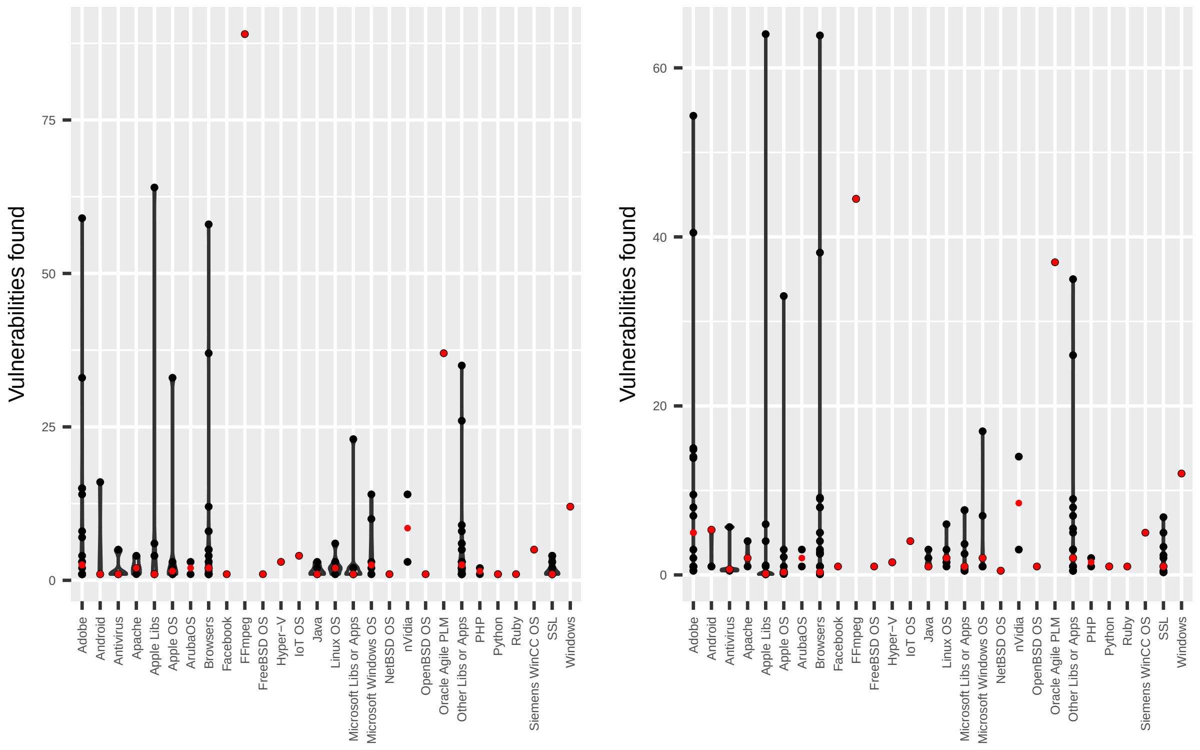

Figure 2: Google: Vulnerabilities by software. ***Left: GaE (Groups as Entities); Right: GaP (Groups as Parts).***

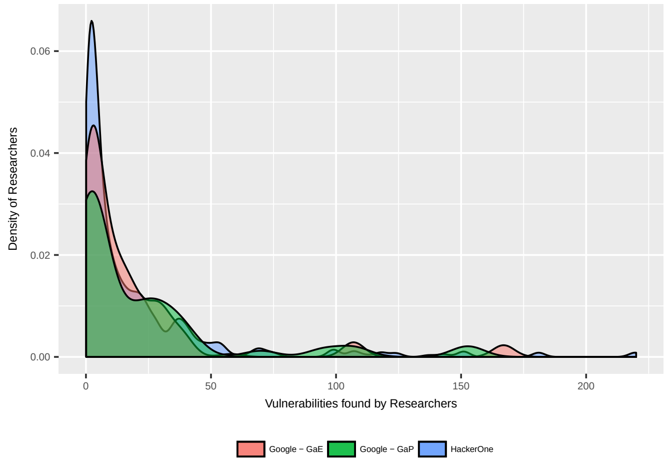

Figure 3: Distribution of number of vulnerabilities found by researchers.

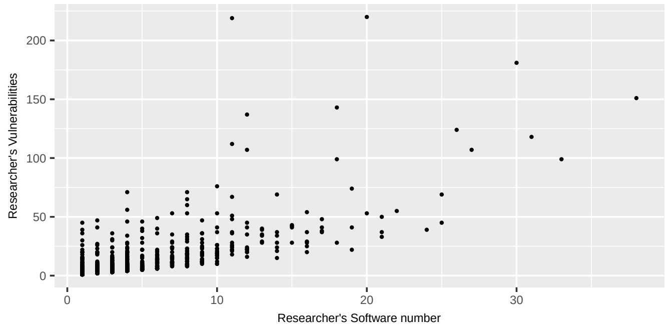

Figure 4: HackerOne: Number of vulnerabilities found by each researcher with respect to the number of software analyzed.

whose nickname is *sergeym* [13]. In Figure 9, we can see the vulnerabilities found by *sergeym* in his, apparently, erratic activity. This researcher's work is, on the one hand, polarized on Yahoo!, which is the outlier in the dataset, but on the other hand it spreads across more than two dozen of different software. This behavior encouraged us to inspect furthermore the dataset in order to find how we could rationally explain the working style of an extremely active researcher as *sergeym*. In fact, he has found a total number of 181 vulnerabilities working on 30 distinct software on the HackerOne platform. In details, he has found 89 vulnerabilities - 49% of his overall findings - in one software, Yahoo!, and the remaining 92 vulnerabilities in other 29 distinct software. Furthermore, in his activity, he has found less than 5 vulnerabilities in 22 distinct software.

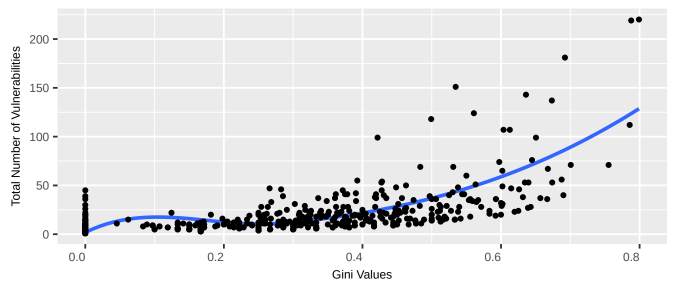

Figure 5: HackerOne: Researchers for Gini Inequality Coefficient and number of vulnerabilities found.

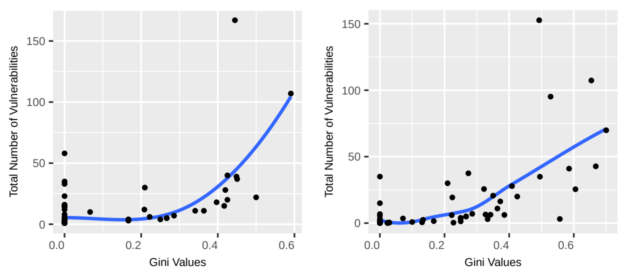

Figure 6: Google: Researchers for Gini Inequality Coefficient and number of vulnerabilities. *Left: GaE; Right: GaP.*

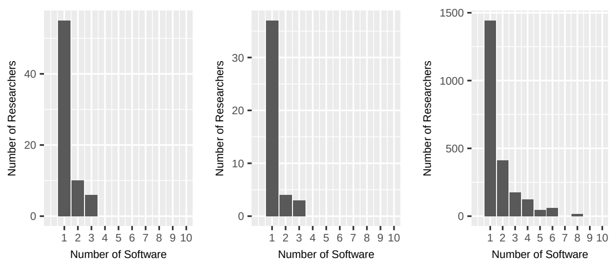

Figure 7: Researchers with Gini coefficient equal to zero: *Left: GaE; Center: GaP; Right: HackerOne*

.

We then focused more on the HackerOne dataset because of its larger size and heterogeneous membership.

5 PRODUCTIVITY IN BUG BOUNTY PROGRAMS

In this section, we describe how we proceeded in the definition of productivity metrics aiming at evaluating researchers performance with respect to different profiles[3]. Next, we swap the context and look at the software in the bug bounty programs aiming at discovering whether there is a higher probability to find a vulnerability in one bug bounty program than in the other.

[3]Productivity, assuming the typical definition of economics, is "the ratio of what is produced to what is required to produce it. Usually this ratio is in the form of an average, expressing the total output of some category of goods divided by the total input of, say, labour or raw materials" [16].

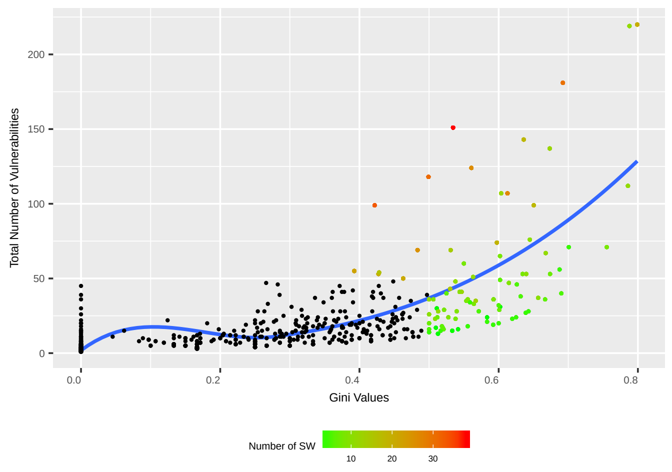

Figure 8: HackerOne: Colored dots are the Hard-Working Researchers group.

.

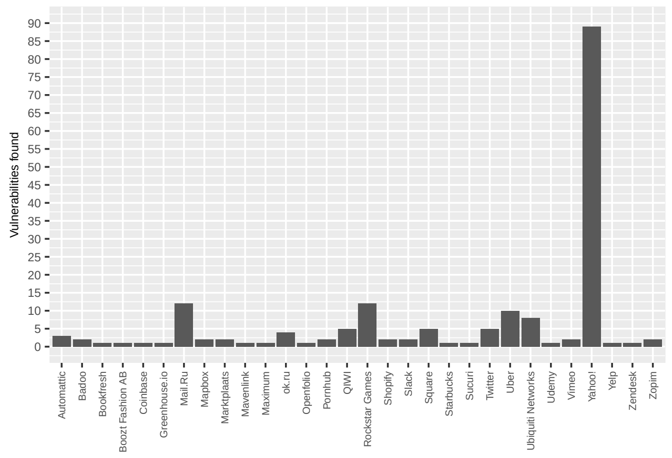

Figure 9: HackerOne: *sergeym*'s apparently unusual pattern of activity

.

For the first analysis, we consider *Researcher Productivity* metric, by assuming that the number of different software where a researcher has found vulnerabilities is the input of the productivity formula, and the total number of vulnerabilities found by a researcher is the output. The ratio of the two factors will give us the average productivity of researchers per software.

In the second analysis, we look to *Software Productivity* taking the total number of researchers who have found at least one vulnerability in the software considered as the input, and the total number of vulnerabilities found in the same software, by any researcher who worked on it, as the output. The ratio of these two factors will give us the average productivity for a certain software per researcher.

5.1 Researcher Productivity

More specifically, we defined the Researcher Productivity metric, as the ratio between the number of unique vulnerabilities (s)he found and the number of different software from which vulnerabilities have been discovered (see Eq. 1). In Figure 10, the Researcher Productivity is shown for the Google and the HackerOne datasets. It can be observed that each dataset is characterized by a large group of researchers who have worked on few software and found few vulnerabilities. Beyond this set, there is a wide dispersion of values, with some remarkable cases of extremely high productivity, as for our definition.

$$Productivity_R = \frac{|Vulnerability_R|}{|Software_R|} \qquad (1)$$

Then, we combined the results of Figure 10 to compare the different regression lines. The slope of regression lines in Figure 11 represents the model of the relationship between the number of software a researcher has worked on and the number of vulnerabilities (s)he found, and it is another way to inform about the productivity of a population of researchers. A higher slope coefficient for the regression line expresses a higher value of Researcher Productivity for the specific group.

5.2 Hard-Working Researchers Productivity

Differently from the previous case, here we have selected the Hard-Working Researchers subset, which is only referred to the HackerOne dataset, and analyzed the Researcher Productivity.

Figure 12 shows the distribution of the Hard-Working Researchers compared with the two Google datasets. As expected, it could be seen that the distribution of the Hard-Working Researchers is more sparse than the full dataset. In Table 1, we have detailed the regression functions. It is important to notice that the Google Vulnerability Research line, both in the GaE and GaP dataset, has a steeper slope than the HackerOne's, either for the full dataset or for the Hard-Working Researchers subset. In its turn, the Hard-Working Researchers subset has a steeper slope than the full HackerOne dataset. This implies that researchers working in the Google bug bounty program have a higher average Researcher Productivity than researchers working on the HackerOne platform. In fact, we can see that a researcher working on the HackerOne platform has found, on average, less vulnerabilities than a researcher who works on the same number of software in the Google program. As expected, HackerOne Hard-Working Researchers are more productive than the average researcher of the full dataset, and closer to the professional model of Google researchers.

Table 1: Researchers Productivity: Regression functions.

$Vuln_R$ and Sw_R are the shortened form of $Vulnerability_R$ and $Software_R$ defined in Eq. 1. Slope of the function is the bold line.

Bug Bounty	Regression Function
HackerOne - Full	$Vuln_R = -2.21 + \mathbf{3.22}.Sw_R$
HackerOne - HWR	$Vuln_R = -12.03 + \mathbf{3.89}.Sw_R$
Google - GaE	$Vuln_R = -0.18 + \mathbf{6.04}.Sw_R$
Google - GaP	$Vuln_R = -9.61 + \mathbf{8.65}.Sw_R$

5.3 Software Productivity

Software Productivity metric is defined as the ratio between the number of researchers who worked on a software and the number of unique vulnerabilities found in that particular software, by all researchers that worked on it (see Eq. 2). In Figure 13, the Software Productivity is shown for the Google and the HackerOne dataset. As for the previous case, there is a clear concentration of points representing software with poor productivity, as for our definition, for which few vulnerabilities have been found by few researchers. Some outliers are present, which exhibit high Software Productivity. In particular, in the HackerOne dataset is clearly visible the anomaly represented by Yahoo! software, identified by the solitary point in the upper right corner of the plot (see Figure 13:*Right*).

$$Productivity_S = \frac{|Vulnerability_S|}{|Researcher_S|} \qquad (2)$$

We further analyzed the data by considering the Yahoo! software as an outlier dominating the statistics. For this reason, we performed an analysis *with Yahoo!* and another one *without Yahoo!*, in order to better understand the patterns of action of HackerOne researchers for which, apparently, Yahoo! software represented a distinct category of software. As for the previous case, we compared the results for Google datasets with HackerOne (see Figure 14) and derived the regression functions (see Table 2). Similar to the previous case focused on researchers, also with respect to the Software Productivity metric, the Google bug bounty program proved more productive than the HackerOne. Considering HackerOne only, the dataset *with Yahoo!* is more productive than the dataset *without Yahoo!*, which means that when researchers work on Yahoo! software, on average they obtain more vulnerabilities than with other software. This suggests a special role played by Yahoo! software among HackerOne researchers, with regard to their pattern of activity.

Table 2: Software Productivity: Regression functions. Full dataset.

$Vuln_S$ and Res_S are the shortened form of $Vulnerability_S$ and $Researcher_S$ defined in Eq. 2. Slope of the function is in bold.

Bug Bounty	Regression Function
HackerOne - *without Yahoo!*	$Vuln_S = -9.93 + \mathbf{2.27}.Res_S$
HackerOne - *with Yahoo!*	$Vuln_S = 62.97 + \mathbf{3.59}.Res_S$
Google - GaE	$Vuln_S = 0.44 + \mathbf{5.87}.Res_S$
Google - GaP	$Vuln_S = 1.63 + \mathbf{4.42}.Res_S$

5.4 Software Productivity for Hard-Working Researchers

Finally, as in a previous case, we isolated the Hard-Working Researchers and conducted a specific analysis. Figure 15 presents the distribution of points in the two cases, *with Yahoo!* and *without Yahoo!* and regression lines, compared with the Google datasets. In Table 3, the regression functions are made explicit.

These observations confirm that the Hard-Working Researchers group of HackerOne has a professional pattern of activity - when the outlier Yahoo! software is excluded - with a Software Productivity close to Google's.

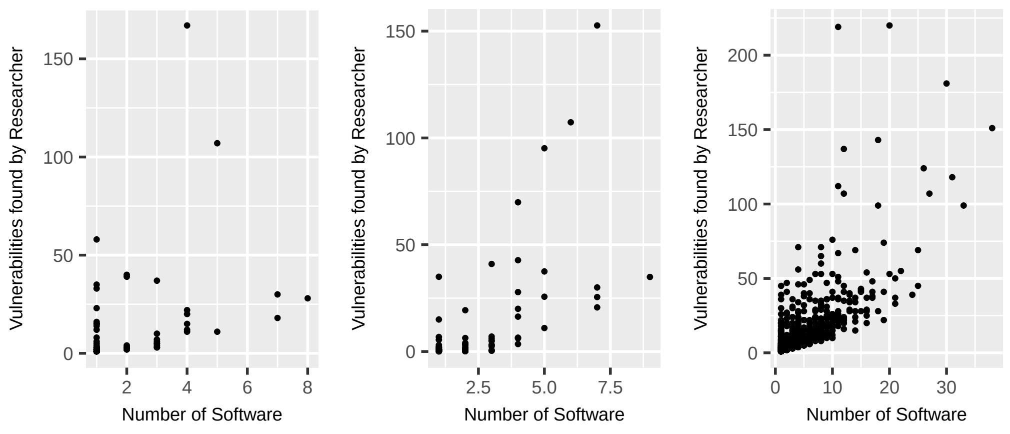

Figure 10: Researchers Productivity: Full dataset. *Left: GaE*; *Center: GaP*; *Right: HackerOne*

.

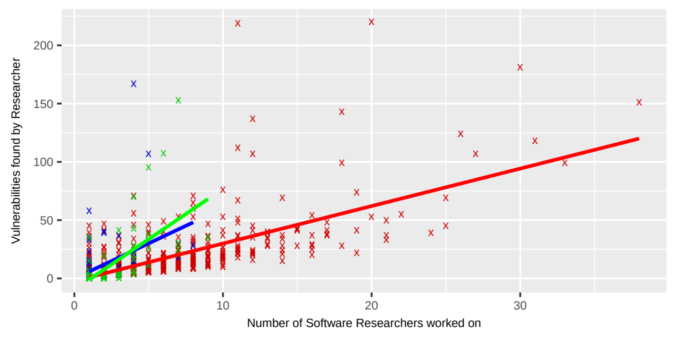

Figure 11: Researchers Productivity: Full dataset. *Red:* HackerOne; *Blue:* Google - GaE; *Green:* Google - GaP.

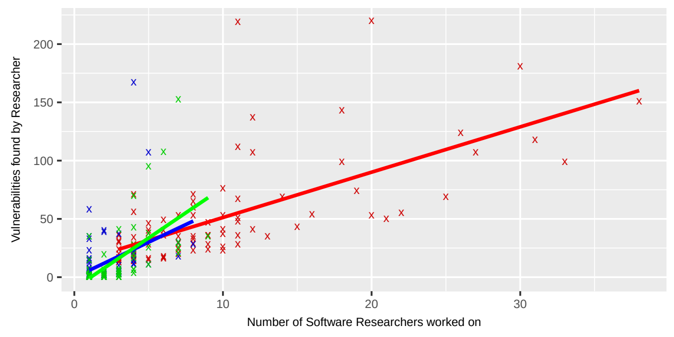

Figure 12: Researchers Productivity: Hard-Working Researchers subset. *Red:* HackerOne; *Blue:* Google - GaE; *Green:* Google - GaP.

In addition, we gained a better understanding about the anomalous nature of the Yahoo! software. Not just it exhibits an average Software Productivity much higher than the other software when the full HackerOne dataset is considered, but the same prevalence,

Table 3: Software Productivity: Regression functions. Hard-Working Researchers subset.

$Vulns_S$ and Res_S are the shortened form of $Vulnerabilitys_S$ and $Researchers_S$ defined in Eq. 2. Slope of the function is in bold.

Bug Bounty	Regression Function
HackerOne - *HWR without Yahoo!*	$Vulns_S = -12.6 + \mathbf{5.83}.Res_S$
HackerOne - *HWR with Yahoo!*	$Vulns_S = 59.58 + \mathbf{11.49}.Res_S$
Google - GaE	$Vulns_S = 0.44 + \mathbf{5.87}.Res_S$
Google - GaP	$Vulns_S = 1.63 + \mathbf{4.42}.Res_S$

even amplified, appears when the Hard-Working Researchers subset is considered. By including the Yahoo! software, the Software Productivity of the Hard-Working Researchers largely exceeds even that of Google. This may signal the fact that looking for Yahoo! vulnerabilities is a goal even of the more experienced and professionalized researchers of the HackerOne platform, which apparently alternate a pattern of activity similar to Google researchers with one more opportunistic exploiting the Yahoo!'s many vulnerabilities.

Another information that these results provide is about the heterogeneous nature of the HackerOne platform, which hosts software that exhibit large differences in Software and Researcher Productivity. Researchers on the HackerOne platform have profiles and patterns of activity much different than those in the Google program. This represents a clear warning against generalizations in the analysis of open platforms like HackerOne.

5.5 Group Productivity

To further analyze the role of Yahoo! software and the behavior of HackerOne researchers, we introduced another metric, that we have called Group Productivity, defined for a certain subset of researcher G as the ratio between the number of vulnerabilities found by members of G ($|Vulnerability_G|$) and the cardinality of G (see Eq. 3. It simply measures the average productivity, in term of discovered vulnerabilities, of researchers of a certain group. In particular, we were interested in isolating researchers working on

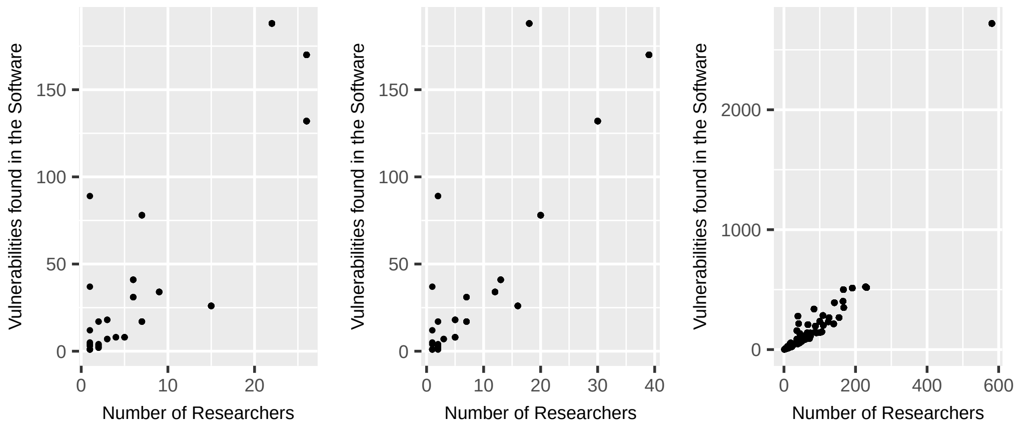

Figure 13: Software productivity: Full dataset. *Left: GaE*; *Center: GaP*; *Right: HackerOne*

.

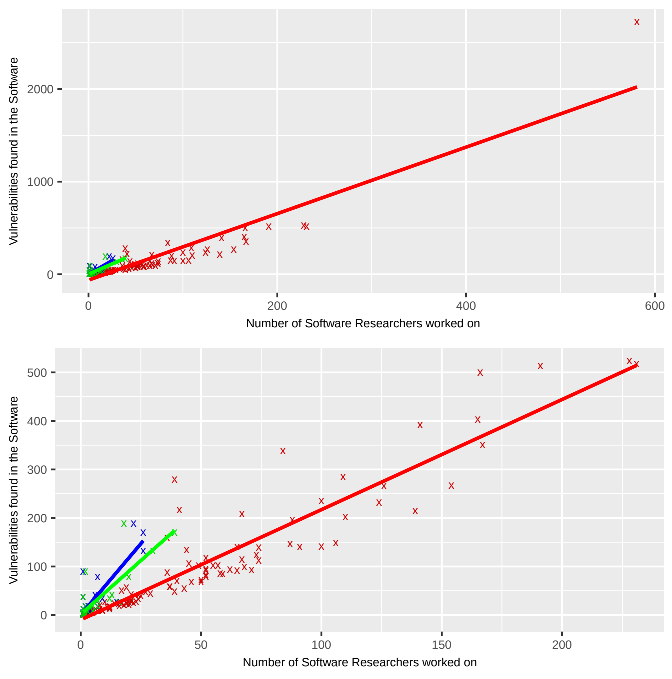

Figure 14: Software Productivity: Full dataset. *Top:* With Yahoo!; *Bottom:* Without Yahoo!.

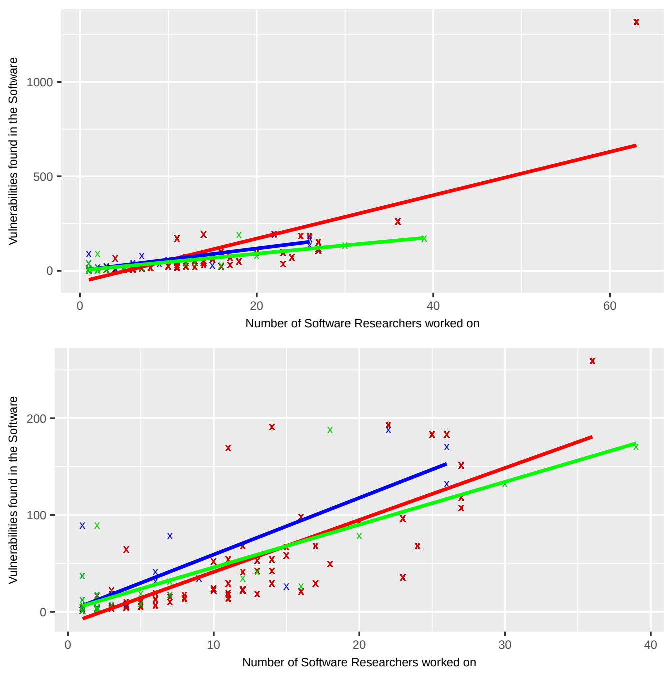

Figure 15: Software productivity: Hard-Working Researchers subset. *Top:* with Yahoo!; *Bottom:* without Yahoo!

.

the Yahoo! software from the others. Table 4 shows the different subsets considered with: *number of vulnerabilities discovered by the group*, *number of researchers included in the group*, and the *Group Productivity value*.

$$Productivity_G = \frac{|Vulnerability_G|}{|G|} \tag{3}$$

With respect to the Group Productivity, it is clear how researchers hunting for vulnerabilities on Yahoo! obtain more vulnerabilities compared to others. In case of the full dataset, the Group Productivity of those working on Yahoo! is close to Google's (i.e., this equivalence signals that just comparing the average number of discovered vulnerabilities between not comparable bug bounty programs may lead to severely mistaken conclusions). In addition, also by selecting the professional subset Hard-Working Researchers and looking to Group Productivity, the results could be misleading, by showing an extremely high productivity largely dominated by a single outlier, like the Yahoo! software.

Table 4: Group Productivity

Bug Bounty	No. of Vulns	No. of Researchers	Group Productivity
Google - GaE	939	84	**11.18**
Google - GaP	939	85	**11.05**
HackerOne - *full dataset*	13370	2360	**5.67**
HackerOne - *without researchers working on Yahoo!*	5871	1779	**3.30**
HackerOne - *only researchers working on Yahoo!*	7499	581	**12.91**
HackerOne - *HWR subset*	4588	91	**50.42**
HackerOne - *HWR subset without researchers working on Yahoo!*	766	28	**27.36**
HackerOne - *HWR subset only researchers working on Yahoo!*	3822	63	**60.67**

6 CONCLUSIONS

With this work, we wished to gain a better understanding of bug bounty programs. We showed that by generically compare different programs may lead to wrong conclusions, because it is not uncommon that they present characteristics (e.g., population, incentive mechanisms) that make them not comparable without a careful analysis. We have also presented examples of not trivial patterns of activity adopted by experienced security researchers, a group far from the naive image of the low-hanging fruits amateurish hacker. These results could help to better evaluate the true benefits a bug bounty programs could carry out. A limitation of our work is the lack of an analysis that weights the efforts with respect to awards (i.e., a cost/benefit analysis approach). Unfortunately, data about awards were too incomplete.

REFERENCES

[1] Lillian Ablon and Andy Bogart. 2017. Zero Days, Thousands of Nights. *RAND Corporation* (2017). https://www.rand.org/pubs/research_reports/RR1700/RR1751/RAND_RR1751.pdf

[2] Akemi Takeoka Chatfield and Christopher G Reddick. 2017. Cybersecurity innovation in government: A case study of US pentagon's vulnerability reward program. In *Proceedings of the 18th Annual International Conference on Digital Government Research*. ACM, 64–73.

[3] European Commission DIGIT. 2019. European Commission raises bounty awards to challenge developers. https://joinup.ec.europa.eu/collection/eu-fossa-2/news/ready-challenge

[4] Eduardo Vela Nava. 2017. Vulnerability Rewards Program: 2016 Year in Review. https://security.googleblog.com/2017/01/vulnerability-rewards-program-2016-year.html

[5] Chris Evans, Eric Grosse, Neel Mehta, Matt Moore, Tavis Ormandy, Julien Tinnes, and Michal Zalewski. 2010. Rebooting Responsible Disclosure: a focus on protecting end users. https://security.googleblog.com/2010/07/rebooting-responsible-disclosure-focus.html

[6] Matthew Finifter, Devdatta Akhawe, and David Wagner. 2013. An Empirical Study of Vulnerability Rewards Programs. *Proceedings of the 22nd USENIX Security Symposium* (2013).

[7] Google Inc. 2017. Google Security Reward Programs. https://www.google.com/about/appsecurity/

[8] Google Inc. 2017. Google Vulnerabilities Research. https://www.google.com/about/appsecurity/research/

[9] HackerOne Inc. 2016. 2016 Bug Bounty Hacker Report - Who are these bug bounty hackers? https://www.hackerone.com/resources/2016-bug-bounty-hacker-report

[10] HackerOne Inc. 2017. Bug Bounty, Vulnerability Coordination. https://www.hackerone.com

[11] HackerOne Inc. 2017. Hacker Activity. https://www.hackerone.com/hacktivity

[12] HackerOne Inc. 2017. The hacker-powered security report 2017. https://www.hackerone.com/resources/hacker-powered-security-report

[13] HackerOne Inc. 2018. Sergey Markov (sergeym) - Hacker Activity. https://www.hackerone.com/sergeym

[14] Keman Huang, Michael Siegel, Stuart Madnick, Xiaohong Li, and Zhiyong Feng. 2016. Diversity or concentration? HackersâĂŹ strategy for working across multiple bug bounty programs. In *Proceedings of the IEEE Symposium on Security and Privacy*, Vol. 2.

[15] Keman Huang, Michael Siegel, and Madnick Stuart. 2018. Systematically understanding the cyber attack business: A survey. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 70.

[16] John W. Kendrick and Marvin Frankel. 2018. Productivity (Economics). *Encyclopaedia Britannica* (2018). https://www.britannica.com/topic/productivity

[17] Microsoft Corporation. 2017. Security Tech Center - Microsoft Bug Bounty Programs. https://technet.microsoft.com/en-us/security/dn425036

[18] Graham Pyatt. 1976. On the interpretation and disaggregation of Gini coefficients. *The Economic Journal* 86 (1976), 243–255. Issue 342.

[19] Jukka Ruohonen and Luca Allodi. 2018. A bug bounty perspective on the disclosure of web vulnerabilities. *arXiv preprint arXiv:1805.09850* (2018).

[20] The International Organization for Standardization. 2014. Information technology, Security techniques , Vulnerability disclosure. *ISO/IEC 29147:2014(E)* (2014). https://www.iso.org/standard/45170.html

[21] The President of the United States. 2017. Vulnerabilities Equities Policy and Process for the United States Government. https://www.whitehouse.gov/sites/whitehouse.gov/files/images/External%20-%20Unclassified%20VEP%20Charter%20FINAL.PDF

[22] The U.S. Department of Justice (DoJ) Criminal Division Cybersecurity Unit. 2017. A Framework for a Vulnerability Disclosure Program for Online Systems. https://www.justice.gov/criminal-ccips/page/file/983996/download

[23] VentureBeat. 2014. How Bugcrowd uses crowdsourcing to uncover security flaws faster than the bad guys do (interview). https://venturebeat.com/2014/08/18/how-bugcrowd-uses-crowdsourcing-to-uncover-security-flaws-faster-than-the-bad-guys-do-interview/

[24] Mingyi Zhao, Jens Grossklags, and Peng Liu. 2015. An empirical study of web vulnerability discovery ecosystems. *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS)* (2015). https://s2.ist.psu.edu/paper/An-Empirical-Study-of-Web-Vulnerability-Discovery-Ecosystems.pdf

[25] Mingyi Zhao, Aron Laszka, Thomas Maillart, and Jens Grossklags. 2016. Crowdsourced security vulnerability discovery: Modeling and organizing bug-bounty programs. In *The HCOMP Workshop on Mathematical Foundations of Human Computation, Austin, TX, USA.*