The Rules of Engagement for Bug Bounty Programs

Aron Laszka[1], Mingyi Zhao[2], Akash Malbari[3], and Jens Grossklags[4]

[1] University of Houston
[2] Snap Inc.
[3] Pennsylvania State University
[4] Technical University of Munich

Abstract. White hat hackers, also called ethical hackers, who find and report vulnerabilities to bug bounty programs have become a significant part of today's security ecosystem. While the efforts of white hats contribute to heightened levels of security at the participating organizations, the white hats' participation needs to be carefully managed to balance risks with anticipated benefits. One way, taken by organizations, to manage bug bounty programs is to create rules that aim to regulate the behavior of white hats, but also bind these organizations to certain actions (e.g., level of bounty payments). To the best of our knowledge, no research exists that studies the content of these program rules and their impact on the effectiveness of bug bounty programs.
We collected and analyzed the rules of 111 bounty programs on a major bug bounty platform, HackerOne. We qualitatively study the contents of these rules to determine a taxonomy of statements governing the expected behavior of white hats and organizations. We also report specific examples of rules to illustrate their reach and diversity across programs. We further engage in a quantitative analysis by pairing the findings of the analysis of the program rules with a second dataset about the performance of the same bug bounty programs, and conducting statistical analyses to evaluate the impact of program rules on program outcomes.

1 Introduction

Opposing malicious hackers, so-called *white hats* or *ethical hackers* strive to contribute to the security efforts of organizations by finding and reporting security vulnerabilities. Various factors motivate white hats, such as monetary benefits, increasing reputation, or job opportunities. Others aim to acquire knowledge by identifying bugs in systems.

White hats frequently work for specific target organizations under the umbrella of paid or unpaid bug bounty *programs* [9, 12, 25]. Further, these programs are now often facilitated by bug bounty *platforms* such as HackerOne, BugCrowd, Cobalt, etc. As part of bug bounty programs, organizations allow white hats to perform ethical hacking on their systems, to identify the loopholes that their internal security teams could not identify (given personnel, time, expertise, and cost constraints) and which could become important targets of black

hats. Platforms facilitate the process by, for example, managing the payment of bounties, serving as a point of contact for conflicts between white hats and bug bounty programs or even law enforcement, and perhaps most importantly, acting as a central point of attraction for white hats and organizations alike.

In this paper, we are contributing to existing work on vulnerability discovery by conducting the first study of bug bounty programs' *rules of engagement*, which are the program rules governing the interaction between white hats and organizations. They fulfill at least two key functions. First, they state for each bug bounty program the expectations regarding white hats' behaviors when they engage in vulnerability discovery on the program's site, and when they submit vulnerability reports to the program. Second, they also bind organizations to certain actions, e.g., the size of bounty payments for specific types of discovered vulnerabilities, expected speed of resolving identified issues, etc.

A careful management of these various factors specified in the program rules may also help to address two key problems, a high number of reports that are later categorized as *invalid* [13] and a high number of *duplicate* findings [26]. The annual reports from bug bounty platforms show that the resulting outcomes can be quite inefficient (see, for example, [5, 6]).

Going beyond the direct relationship between a program and white hats, the rules likely also shape the competitive process between the different programs on a platform. The stated terms may contribute to attract white hat researchers, or they may dissuade them from working for a particular program. All these factors are so far largely unexplored.

In this paper, we collected and analyzed the program rules of 111 bounty programs on a major bug bounty platform, HackerOne. We qualitatively study the contents of these rules to determine a taxonomy of statements governing the expected behavior of white hats and organizations. We also report specific examples of rules to illustrate their reach and diversity across programs. We further engage in a quantitative analysis by pairing the findings of the analysis of the program rules with a second dataset about the performance of the same bug bounty programs, and conducting statistical analyses to evaluate the impact of program rules on program outcomes.

We will proceed as follows. In Section 2, we discuss related work. In Section 3, we describe our dataset. In Section 4, we present qualitative results. We then provide a quantitative investigation of the program rules and program performance including a regression analysis in Section 5. In Section 6, we discuss various aspects of the rules along with some suggestions for program improvements. We offer concluding remarks in Section 7.

2 Related Work

In the academic domain, several studies have focused on the discovery of software vulnerabilities (e.g., [4, 7, 17, 18, 21, 23]), and software vulnerability markets [1–3, 19]. And, in recent years, there has also been a growing research interest in bug bounty programs. Researchers have conducted multiple empirical analyses of independently run bug-bounty programs. Studying the incentives and practices

of organizations and white hats initiating and participating in such programs, respectively, is crucial to understand their economic viability and impact on security. For example, Finifter et al. empirically investigated the Google Chrome and Mozilla bug-bounty programs [9], and suggested that these programs are more cost-effective compared to hiring full-time security researchers in terms of finding security flaws. In an effort to better understand the human side of vulnerability discovery, Edmundson et al. conducted an experiment where participants were asked to identify seven security vulnerabilities embedded in a code sample, but no participant was able to accomplish this task alone. However, when the researchers collected a random sample of 50% of the participants, the probability of finding all bugs increased to 95% [8].

Researchers have also studied the dynamics of bug bounty platforms [25, 11, 15]. Zhao et al. conducted a comprehensive study of two emerging bug bounty platforms, Wooyun and HackerOne, to understand their characteristics, trajectories and impact [25]. One key finding of their work is that not only top contributors are important for bug bounty platforms, but that also the long tail of white hat researchers makes significant and diverse contributions (as a group). In follow-up research, Maillart et al. empirically studied reward distribution and hacker enrollments of public bounty programs on HackerOne and found that growing rewards cannot match the increasing difficulty of vulnerability discovery, and thus hackers tend to switch to newly launched programs to find bugs more easily [15]. Our work may help to understand how programs aim to attract (or retain) white hats by offering attractive rules of participation.

These research efforts also helped to identify hurdles which may limit the further growth of bug-bounty platforms and programs. The data suggests that bug-bounty platforms suffer from a high rate of submitted reports which for a variety of reasons are considered invalid. The percentage of invalid reports is currently significant, ranging from 35% to 55% on different platforms [14]. Programs may try to regulate the in-flow of invalid reports by adjusting the rules and incentives to discourage certain behaviors [13]. Further, due to the decentralized nature of vulnerability discovery, white hats may discover the same issues and file reports which are then recorded as so-called duplicates [14]. This problem can potentially be alleviated by designing an allocation plan for white hats' efforts and diversifying the workforce [26].

We are unaware of any work which has investigated the program rules for vulnerability discovery to complement the aforementioned research efforts.

3 Dataset

Since its inception in 2012, HackerOne has continuously grown as a community and has been attracting numerous white hats and organizations to participate in its bounty bounty platform. By September 2017, white hats on HackerOne have successfully contributed over 50,000 bug reports (which have been fixed), and have been paid bounties totaling over $20 million. Participating organizations have thanked over 4,500 hackers.

We collected data in January 2016. For the (then available) 112 public programs on HackerOne's website, we downloaded the program descriptions and the announced minimum bounty paid by the program. Of the 112 organizations, 52 register themselves with a clause of paying *"no minimum bounty."* In our further analysis, we consider 111 organizations by excluding HackerOne, which runs a bounty program on its own platform, and consider only outside organizations participating on HackerOne.

Of these 111 programs, we were able to download the *detailed history* of rule description changes, bugs resolved, and hackers thanked for 77 programs. For each one of these programs, we determined the date of the last major update to the rule description before January 2016[5], and we counted the number of bugs resolved and hackers thanked in the interval between the last major rule update and January 2016. By dividing these numbers with the length of the time interval, we were able to determine the rate of hackers thanked and bugs resolved for the program description that was in effect in January 2016.

In the subsequent sections, we focus on the program descriptions of the bounty programs, which include the rules of engagement, and provide a qualitative analysis of their contents. We also conduct several analyses correlating discernible features of these program rules with important metrics such as the number of bugs resolved and hackers thanked.

We further pair the aforementioned data with an additional dataset. We collected, following the methodology of previous research [24, 25, 15], data about the average bounty and the age of bounty programs, and their Alexa ranks (which is based on web traffic data). This additional data is used for a deeper assessment of the rules' impact in a regression analysis.

We are also aware of the limitations of this dataset. One limitation is that we do not have data for private bug bounty programs, which are only accessible to a selected group of (internal) researchers. In addition, other competing bug bounty platforms, such as BugCrowd, could affect white hat behaviors on HackerOne as well. However, our current dataset does not include these competing platforms.

4 Qualitative Study

On a high level, each line of a program description carries with it some meaning and may provide important information to white hats who are interested in a particular bug bounty program. On HackerOne, each organization has been given the liberty to specify the description of the program in its own way, which on the one hand promotes diversity, but on the other hand may complicate matters for white hats and may affect the comprehensiveness of the information covered. Since there is no framework for structuring rules, the necessary first step in studying rules is to construct a *general taxonomy*. In this section, we aim to provide such a taxonomy of the contents of the rules on the basis of "what they are trying to convey" to capture a generic structure for the program description.

[5] We used the Python `difflib` implementation of the Ratcliff/Obershelp matching algorithm [20] with parameter 0.9.

To achieve this objective, the program descriptions were parsed statement by statement and in iterations to extract information and to tabularize the different statements contained in the rules according to the evolving taxonomy. We also cross-verified the extracted information to check whether we followed a consistent classification process. If a statement in the rules was found to fit in more than one category, then it was marked accordingly. If a statement conveyed no concrete guidance to the white hat, we categorized it under "other instructions." Based on this process, the rules specified by the 111 organizations could be categorized and defined according to the following taxonomy. To save space, we list example rule statements in Appendix A and only reference them in the main text.

4.1 In-scope areas

Statements of this type define the exact scope of the bug-bounty program. The organizations state typically the list of system and product areas on which white hats should work. The majority of the organizations will list their core *production websites* as the target of bug bounty. Some organizations also list *staging websites*, and encourage or require white hats to conduct vulnerability research only against them (Example 1). Some organizations may also provide source code to white hats (Example 2). In addition to web applications, there may be other types of components, such as *APIs*, *mobile applications*, and *desktop applications*, which are in scope. Further, vulnerability discovery may also extend to *physical products* with digital components. For example, ToyTalk allows the search for "a security issue in our products or service" which include a doll and a playhouse with voice capabilities.

4.2 Out-of-scope areas

Each organization can also explicitly specify all the domains and areas that they do *not* wish the white hats to work on. We have identified the following reasons for listing an area as out-of-scope. First, organizations typically exclude web applications (e.g., blog, support, community, etc.) hosted by third-parties, as these websites are not controlled by the organization, and/or have low risk (e.g., no user data) (Example 3). Second, customer websites or services are usually out of scope as well (Example 4). Third, some organizations also exclude areas that belong to business partners or subsidiaries (Example 5). Fourth, as discussed above, organizations may set up staging sites, but also *explicitly* declare their production site to be out of scope (Example 6).

4.3 Eligible vulnerabilities

This category provides additional detailed rules focused on the types of vulnerabilities which the organizations want the white hats to find. In general, organizations encourage white hats to spend their effort on those types of vulnerabilities which they likely consider of particular severity to their organizations. Frequently mentioned vulnerability types across many organization are:

SQL Injection, Remote Code Execution, Cross-Site Request Forgery, Directory Traversal, Cross-Site Scripting, Information Disclosure and Logical Issues.

Rules can be fairly precise and may even include additional conditions such as the potential for financial damage. One example is from Coinbase (an exchange for digital assets) (Example 7). However, some organizations may not rely on specifying a set of vulnerability types. For example, Envoy's rules state that reports should be about "*issues that are very clearly security problems.*"

4.4 Non-eligible vulnerabilities

Certain types of vulnerabilities are often excluded from being rewarded with a bug-bounty, because they have very low or no security risk from the perspective of an organization. Frequently mentioned examples include Self-XSS, Logout CSRF, no maximum password length, etc. Some of these issues may also be rather easy to identify for a white hat. Listing such non-eligible vulnerabilities in an upfront manner can reduce the cost of processing reports which may even be declared invalid. Please note that invalid reports are very common, and a significant challenge to bug bounty programs [13]. Denial of Service (DoS) is another typical type of non-eligible vulnerability as some organizations already know their general vulnerability to this type of attack, or doubt that any white hat report would yield novel insights (Example 8).

4.5 Deepening engagement with organizations

This category includes further instructions to the white hats (going beyond the scope and eligible vulnerabilities categories) in regards to how they can better engage in vulnerability research for the organization. Specifying such instructions helps the white hats to align their effort with the organization's interest, and to more likely find bugs which will be rewarded. For example, some organizations ask white hats to create dedicated test accounts (Example 9). Another interesting case is Square's Capture-the-flag (CTF) challenge within its bug-bounty program. Basically, Square hides a secret flag inside its system, and whoever finds it can qualify for a $1,000 reward. Understanding the impact of such mechanisms on white hat engagement is an interesting aspect for future work.

4.6 Prohibited or unwanted actions

Rules in this category list further instructions to the white hats in regards to what they should not do (going beyond the scope restrictions and non-eligible vulnerabilities) when they are searching for vulnerabilities for the organizations. These instructions specify detailed bounds to the work of white hats which organizations may use to protect their business interests, while participating in crowdsourced security research.

There are several subcategories within this rule. First, many organizations forbid or limit the use of automated scanning, because they can lead to a large amount of false positive reports, and may cause a significant amount of traffic to the site (Example 10). Another subcategory rule disallows interaction with

other users' accounts, in order to reduce the risk potentially caused by vulnerability research (Example 11). Third, other dangerous activities, such as social engineering (Example 12) and physical access to a data center (Example 13), are also prohibited.

Disregarding rules in this section may disqualify the white hats from receiving a bounty reward or participating in the program in the future. Violations could also cause white hats to face legal actions against them or exclusion from the entire platform.

4.7 Legal clauses

Some organizations explicitly specify details of legal issues related to bug bounty. Statements of one subcategory promise not to bring legal actions against white hats, if rules are followed (Example 14). Another subcategory is to remind the white hats to comply with all relevant laws and regulations (Example 15). Another type of statements withhold the right to modify the rules at anytime (Example 16). We will analyze the occurrence of legal statements in Section 5.1.

4.8 Participation restrictions

Although bug-bounty platforms are known for their openness to welcome white hats from around the world, some organizations explicitly exclude certain types of individuals from participating. Some organizations disallow their employees to participate, possibly in fear of misaligned incentives (Example 17). Organizations might also restrict participation based on white hats' nationality (Example 18). Some programs include explicit age restrictions (Example 19).[6]

4.9 Submission guidelines

In this category, the organizations may describe what kind of details about discovered vulnerabilities they wish to have included in reports submitted by the white hats. Some organizations are very particular about report standards and they expect reports in a specific format with sufficient details like screenshots, pages visited, etc. (Example 20).

4.10 Disclosure guidelines

Organizations may also state whether they allow white hats to engage in public disclosure of the identified problems, or they may ask white hats to give them enough time to triage and fix the issue before public disclosure. We will discuss this aspect in Section 5.1 (Example 21). In addition, a bug bounty platform may have specific rules that apply to all programs concerning disclosure. HackerOne, for example, listed a process in its Vulnerability Disclosure Guidelines.[7]

[6] HackerOne's Privacy Policy (https://www.hackerone.com/privacy/) states as a general policy that "we welcome minors to submit reports to HackerOne." However, the site is not directed at minors below 13 who would need to have their parents/guardians submit vulnerability reports and to set up an account.

[7] HackerOne's Vulnerability Disclosure Guidelines (https://www.hackerone.com/disclosure-guidelines/)

4.11 Reward evaluation

Rules in this category specify the concrete point system or evaluation process that the organization uses to determine whether white hats' submissions are eligible for rewards or appreciation. Some organizations list detailed reward evaluation criteria for different types of vulnerabilities. For example, Twitter provides a table in its rules statement which matches reward amounts to specific types of vulnerabilities, areas of the site, and various other conditions. Other organizations may simply specify a minimum reward (Example 22).

Another rule in this category is a "Duplicate Report Clause," which states whether only the first submission of a particular vulnerability is eligible for a reward, or whether later submissions may also receive (partial) rewards (Example 23). We will analyze the occurrence of such statements in Section 5.1. Further, organizations often state that they have the final decision authority whether a reward will be given (Example 24).

In addition, rewards are not restricted to monetary bounties, but could also represent other forms of appreciation, such as hacker points or swags (Example 25). As previously stated, about 50% of the organizations do not pay any monetary rewards.

4.12 Company statements

In our efforts to iteratively classify rules, this last category contains statements which are more of a description rather than clear instructions or other reward-relevant information. A key objective of this category appears to be demonstrating an organization's willingness to improve security, and to collaborate with the white hat community (Example 26).

5 Quantitative Analysis

In this section, we provide an exploratory quantitative analysis of our rule dataset. Note that for measuring the rates of bugs resolved and hackers thanked, we restrict our analysis to the 77 programs for which detailed history was available. This restriction ensures that we compute the rates for each program in a time interval when the rules of the program did not change significantly.

We first study whether more detailed rule descriptions lead to greater success (Section 5.1). To answer this question, we study the relationships between description length and the number of bugs resolved and hackers thanked by a program. We show that programs with longer descriptions generally tend to be more successful, which suggests that detailed program descriptions are an important factor in success. Then, we investigate the readability of program rules using established metrics from the field of readability studies (Section 5.1), and show that the readability of program descriptions could be significantly improved in practice. Next, we study three important clauses that program rules may include: duplicate reports, legal actions and public disclosure (Section 5.1). We show that the presence or absence of these clauses can have a very strong impact on the success of a program, which implies that organizations need to include them

in their rules if they wish to be successful. We also study statements for staging sites, test accounts, and source code availability in the program description. Finally, we perform a detailed regression analysis (Section 5.2), and study the combined effects of description length, various clauses, etc.

5.1 Descriptive Analysis

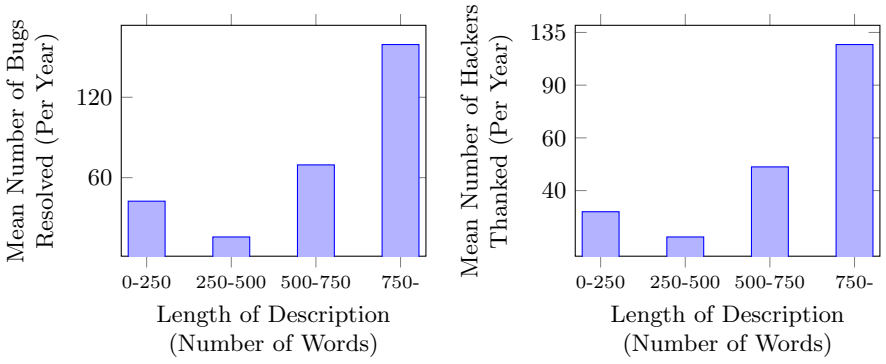

Fig. 1. Statistics for programs grouped by description length (measured as number of words). Please note the logarithmic scaling of the vertical axes.

Length of Bug Bounty Rules The average length of program rules is 481 words ($N = 77$). The shortest description is 72 words (Vulners) and the longest one is provided by ownCloud with 1,744 words. To provide an initial overview of the data, we categorize the organizations into four groups based on program description length (measured as number of words). These groups contain programs with word counts of 1) 0 - 250 words ($N = 13$), 2) 250 - 500 words ($N = 36$), 3) 500 - 750 words ($N = 16$), and 4) 750+ words ($N = 12$). Figure 1 shows the average rates of bugs resolved and hackers thanked for each group. Note that the rates were computed for each program over a time interval in which the description was unchanged, dividing the number bugs resolved and hackers thanked by the length of the interval in years. Tables 2, 3, and 4 in Appendix B list the descriptive statistics of word count, bugs resolved, hackers thanked and bounty paid for all organizations (Table 2), organizations paying minimum bounty (Table 3) and organizations paying no minimum bounty (Table 4), respectively.

It is noteworthy that the *length of program rules is positively associated with the average rate of bug resolved as well as the average rate of hackers thanked* (see Figure 1 and Table 2). These observations also hold for all organizations paying a minimum bounty ($N = 44$, see Table 3). For organizations that are not paying a minimum bounty ($N = 33$), these relationships hold *very consistently* (see Table 4). However, there is no obvious trend observable for the relationship between the length of program rules and the average bounty paid by programs for valid discoveries (see Tables 2 and 3).

Readability of Program Rules We computed various metrics for the readability of program rules. For brevity, we report the results only for the Flesch Reading-Ease Score [10] here (see Figure 2), which is an established metric in the field of readability studies. Results for other metrics are shown in Figure 6 in Appendix D, but they do not differ in a meaningful way regarding the following basic observations.

The higher the score, the easier a document is to read. Scores towards 100 indicate that a minor in 5th grade would likely understand the document without problems. A score of 30 and below typically requires a college degree. Law review articles and technical documents frequently score in the 30s.

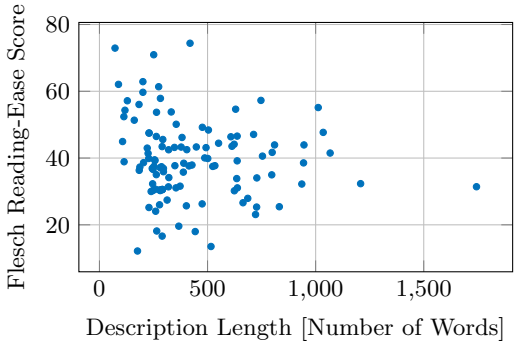

Fig. 2. Length and readability of program descriptions.

We find that the average level of the Flesch Reading-Ease Score for the sample of program rules is 39.6, indicating a set of documents requiring some college education (on average). The least readable document scored 12.2, whereas the most readable document had a score of 74.4. There are 18 program rules that score below 30, which indicates documents that are very difficult to read.

While our analysis does not yet account for the specific characteristics of program rule documents (e.g., technical terminologies, tables etc.), it is indicative that *improvements could be undertaken to make these documents more approachable*. Perhaps in contrast to the going practice for many forms of legal agreements, program rules should be written with the intention of being read and understood by white hats who search for vulnerabilities in a particular program. A lack of readability may be a contributing factor to inefficient outcomes, and might encourage white hats to prefer other programs.

Statements about Duplicate Reports, Legal Actions, and Public Disclosure In the taxonomy section, we already provided an initial overview of the various rules included in the program description. In the following, we study three key rules numerically.

First, we recorded whether the program rules include a duplicate report clause, i.e., whether the organization explicitly specifies if submitting a duplicate

report will affect the white hat's eligibility for a bounty or not. Please note that duplicates occur very frequently in practice, and may pose a significant problem. For Google's bug bounty program and on the BugCrowd platform, the number of duplicates is higher than the number of valid reports. The ratio of duplicates is lower on HackerOne, but still substantial. It is therefore likely that white hats prefer to work with programs that are aware of this challenge and discuss it in their program rules.

Second, we identified programs that have some form of a legal action clause. Using a legal action clause, an organization informs white hats under what conditions it may (or may not) bring a lawsuit against them. Due to several highly-publicized incidents, where companies sued white hat hackers, or prevented them from speaking at conferences or other events, we believe that such statements can influence a white hat's decision to work for a specific program.

Third, we investigated which programs include a specific statement regarding public disclosure. Organizations may be particularly concerned about the internal security of their systems and applications, hence they may prohibit white hats from disclosing any identified vulnerabilities to other entities for a specified time period or until the bug has been fixed. HackerOne's Vulnerability Disclosure Guidelines allow white hats to publicly disclose information about bugs 180 days after they have submitted the report. Hence, organizations who take the extra step to alter their program policy may have specific concerns, and the presence of such a clause may also influence white hat behavior.

In general, we believe that specifying these three policies is indicative of a better developed program by the organization. To verify this, we investigated the 111 programs and found that 51 organization mention at least one of the three clauses in their programs. For these 51 organizations, we further show their status in Figure 3. Only 10 out of these 51 organizations have all three clauses.

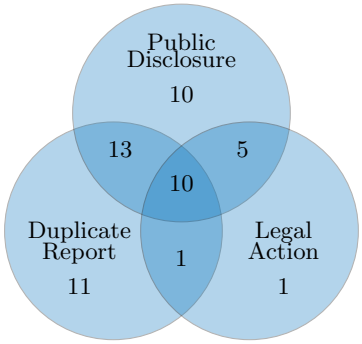

Fig. 3. Venn diagram explaining extensibility of rules.

In Figure 4 (and Table 5 in Appendix C), we provide descriptive statistics of how the presence or absence of these three rule clauses are related to the rate of hackers thanked and rate of bugs resolved. We observe that the presence of these

rules is associated with more active programs. Both the rate of hackers thanked and the rate of bugs resolved are *significantly higher on average for programs that include these statements* in their program rules.

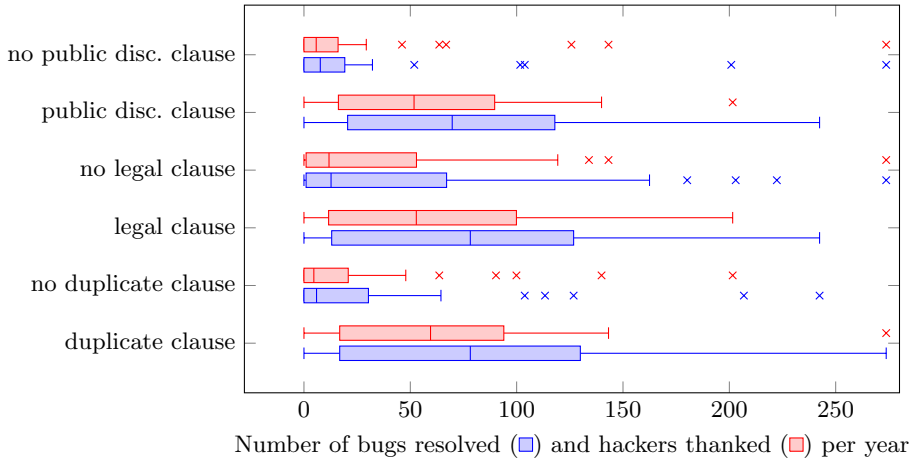

Fig. 4. Statistics for duplicate report, legal action, and public disclosure clauses.

Statements about Staging Sites, Test Accounts, and Source Code Availability Other important indicators of sophistication are whether the organization provides white hats with a staging site for identifying vulnerabilities, whether it asks them to create designated testing accounts, and whether it allows them to download a copy of the application/software for testing.

Our classification shows that 5 out of 111 organizations have staging sites, 24 ask white hats to use a test account, and 13 provide source code of the application/software. When we investigate whether the availability of a staging site or source code impacts the rate of approved vulnerability reports and hackers thanked, we do not observe a very strong pattern (see Figure 5 below and Table 6 in Appendix C). However, the requirement to use test accounts appears to have positive impact.

5.2 Regression Analysis

For the findings stated above, further quantitative analysis is needed to substantiate the observed effects. Particularly, we want to study the combined effects of rule features, including the length of the rule (L), the Flesch Reading-Ease Score that measures the readability of the description (R), the existence of legal action, duplicate report, public disclosure, staging site, test account, and source code download statements (LE, DU, DI, ST, TA, SC) on the success of a program, which is measured by the number of bugs resolved (V). Therefore, we build the following least square regression model:

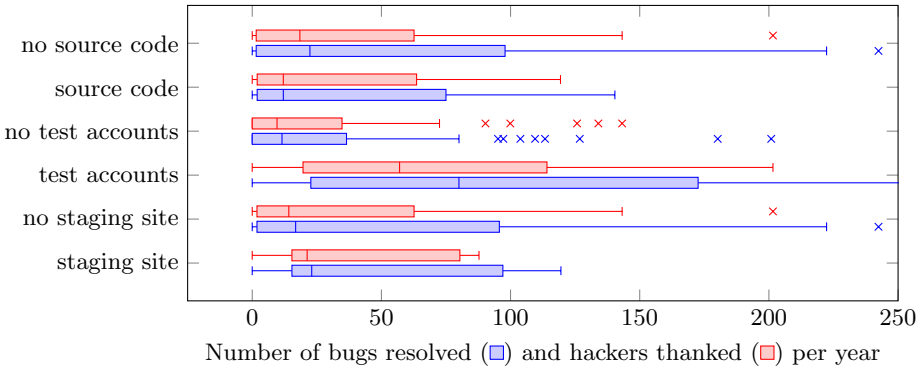

Fig. 5. Statistics for statements of staging sites, test accounts, and source code.

$$V = \beta_0 + \beta_1 L + \beta_2 R + \beta_3 LE + \beta_4 DU + \beta_5 DI + \beta_6 ST + \beta_7 TA + \beta_8 SC + \boldsymbol{\beta_9 Z} + \epsilon. \quad (1)$$

In the regression model, we have considered other characteristics of bug bounty programs that could affect both the success of a program and the textual features, in order to mitigate the correlated omitted variable bias. More specifically, we add three control variables (represented as a vector $\boldsymbol{Z}$ in the regression model), based on previous work [25, 15]: B is the average bounty paid by the program,[8] T is the age of the bug bounty program, and A is the log of the Alexa rank of the organization's website.[9] Alexa rank proxies for the complexity of the website, and a more complex website is likely to have both longer rule descriptions and inherently more vulnerabilities to find.

The results of our regression analysis can be found in Table 1[10]. We incrementally add the factors to the regression model, beginning with a simple model explaining the number of discovered vulnerabilities through the varying length of program rules. Other more complex models follow.

Our first observation is that the length of the rule description is positively correlated with the number of vulnerabilities discovered. The correlation is significant in three out of the four models. Several hypotheses could explain this positive correlation. First, a long rule may indicate that the organization spends more effort on improving the engagement with white hats (e.g., by giving more guidance on what to look for), which in turn makes white hats more productive.

[8] Since not all programs disclose their average bounty, we have to restrict our analysis to 58 data points in this subsection.

[9] A lower value of Alexa rank represents a more popular website. For example, an Alexa rank of 1 indicates the most-visited website.

[10] Note that we use data from the entire history of each bug bounty program. We have also tested the models using only data available after the last major rule update of each program. The regression analysis shows the same directionality of effects, but the dataset is much smaller to report a robust analysis.

Table 1. Regression Results

VARIABLES	(1) V	(2) V	(3) V	(4) V
Length of the rule (L)	0.18***	0.09*	0.09*	0.01
	(0.04)	(0.04)	(0.04)	(0.05)
Average bounty (B)		0.12*	0.12*	0.09*
		(0.05)	(0.05)	(0.05)
Age of the program (T)		0.05	0.05	0.13***
		(0.04)	(0.04)	(0.05)
Log(Alexa rank) (A)		-4.65	-4.30	-4.20
		(2.86)	(2.98)	(2.88)
Readability score (R)			-0.51	
			(0.79)	
Has legal clause (LE)				23.04
				(27.41)
Has duplicate report clause (DU)				47.39*
				(22.08)
Has public disclosure clause (DI)				60.41**
				(24.45)
Has staging site (ST)				1.10
				(40.70)
Has source code (SC)				45.56*
				(27.03)
Asks to use test accounts (TA)				1.01
				(26.78)
Constant	-15.21	23.21	34.25	-14.40
	(18.95)	(39.10)	(45.93)	(39.34)
Observations	54	54	54	54
R-squared	0.27	0.43	0.44	0.57

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

Second, a long rule could also be associated with larger scope, leading to more opportunities for finding bugs.

We do not observe a significant correlation between the readability score and the number of bugs resolved. We see positive and statistically significant coefficients for the existence of duplication clause, disclosure statements, and source code.

In summary, the results indicate that rules with more content (e.g., more detailed list of included / excluded areas and issues) and explicit statements on duplication, disclosure, etc., are associated with more bugs resolved. This suggests that rules could have indeed a tangible impact on bug bounty program performance, and organizations should spend more effort to maintain and improve their rules. On the other hand, we are also aware of limitations of our regression analysis. Particularly, the sample size is rather small, and the dataset has some additional limitations, as we have discussed in Section 3. As such, a potential future work item is to include more data points into our regression. One could also consider other bug bounty platforms, such as BugCrowd and Cobalt, to conduct a cross-platform study.

6 Discussion

The emergence of bug bounty platforms allows many different organizations, such as Yahoo!, General Motors, and even the U.S. Department of Defense, to harvest the power of the global white hat hacker community for improving security. However, as previous research shows (e.g., [13]), effectively and efficiently engaging white hats is a challenging task. In addition, there are risks for both sides. For organizations, there are security risks associated with vulnerability research and disclosure. For white hats, there are legal risks to worry about as well as a potential lack of adequate appreciation of their findings. The program rules serve as a key method to control the risk and facilitate engagement.

Currently, program rules are primarily created by participating organizations independently. Therefore, they vary by content, length, style and many other factors. The bug bounty rule taxonomy we assembled in Section 4 is a first step toward organizing and studying these widely different bug bounty rules. Based on HackerOne's public bug bounty programs, we created 12 categories of rule statements. In the future, this taxonomy can be referenced by organizations when they create or update their own bug bounty rules. Further, the taxonomy also provides a basis for academia to further analyze program rules on different bug bounty platforms.

Our research mainly focuses on the rules of individual bug bounty programs. However, it is also possible that a platform-wide rule influences hacker engagement. We also examined the platform rule created by HackerOne, and determined that it only provides some high-level guidance, and that it primarily refers to individual program's rules for critical issues. In addition, we find a potential issue with the following statement in the platform rule: *"Security Teams will publish a program policy designed to guide security research into a particular service or product. You should always carefully review this policy prior to submission as they will supersede these guidelines in the event of a conflict."* It is surprising that the guidelines do not state that white hats should read the policy *before* doing vulnerability research, as the investigative process can bring harm to an organization's system if not properly conducted. We suggest that platforms shall work more closely with individual programs to make the platform rules consistent with the diverse program rules. Also, we suggest platforms to create more comprehensive rules for cases not covered by individual rules.

7 Conclusion

As bug bounty platforms gain in perceived sophistication and impact, the participation of organizations and white hats will likely continue to increase. As such, it will become increasingly important to appropriately manage the rules which govern the interactions between the different stakeholders.

Our analyses demonstrate that bug bounty programs are on average associated with better success characteristics if the level of comprehensiveness of their rules of engagement increases. We demonstrate this finding for a high level metric (i.e., program length) as well as detailed characteristics such as the presence

of legal action clauses, rules for duplicate submissions and rules for public disclosure. These observations are novel to the research literature on bug bounty platforms. We anticipate that our analysis will be motivation to bug bounty programs and platforms to pay greater attention to the detailed rules in order to provide a fair and more effective workplace for white hat researchers.

Further work is desirable to solidify and extend our findings. In particular, we plan an additional iterative analysis of the program rules to extract more performance-relevant criteria and to embed them in the statistical analysis. Further, the scope of the analysis could be broadened to include more bug bounty platforms in order to add robustness to the findings.

Acknowledgment: We thank the anonymous reviewers for their comments. The research activities of Jens Grossklags are supported by the German Institute for Trust and Safety on the Internet (DIVSI).

References

1. Algarni, A., Malaiya, Y.: Software vulnerability markets: Discoverers and buyers. International Journal of Computer, Information Science and Engineering 8(3), 71–81 (2014)
2. Bacon, D., Chen, Y., Parkes, D., Rao, M.: A market-based approach to software evolution. In: 24th ACM SIGPLAN Conference Companion on Object Oriented Programming, Systems, Languages, and Applications (2009)
3. Böhme, R.: A comparison of market approaches to software vulnerability disclosure. In: Müller, G. (ed.) Emerging Trends in Information and Communication Security, pp. 298–311. Springer Berlin Heidelberg (2006)
4. Bozorgi, M., Saul, L., Savage, S., Voelker, G.: Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 105–114 (2010)
5. Bugcrowd: The state of bug bounty (July 2015)
6. Bugcrowd: The state of bug bounty (June 2016)
7. Clark, S., Frei, S., Blaze, M., Smith, J.: Familiarity breeds contempt: The honeymoon effect and the role of legacy code in zero-day vulnerabilities. In: Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC). pp. 251–260 (2010)
8. Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A., Wagner, D.: An empirical study on the effectiveness of security code review. In: Engineering Secure Software and Systems (2013)
9. Finifter, M., Akhawe, D., Wagner, D.: An empirical study of vulnerability rewards programs. In: USENIX Security Symposium (2013)
10. Flesch, R.: A new readability yardstick. Journal of Applied Psychology 1948(32), 221–233 (1948)
11. Huang, K., Siegel, M., Madnick, S., Li, X., Feng, Z.: Poster: Diversity or concentration? Hackers' strategy for working across multiple bug bounty programs. In: 37th IEEE Symposium on Security and Privacy (S&P) (2016)
12. Kuehn, A., Mueller, M.: Analyzing bug bounty programs: An institutional perspective on the economics of software vulnerabilities. TPRC Conference Paper (2014)
13. Laszka, A., Zhao, M., Grossklags, J.: Banishing misaligned incentives for validating reports in bug-bounty platforms. In: 21st European Symposium on Research in Computer Security (ESORICS). pp. 161–178 (2016)

14. Laszka, A., Zhao, M., Grossklags, J.: Devising effective economic policies for bug-bounty platforms and security vulnerability discovery. Journal of Information Policy 7, 372–418 (2017)
15. Maillart, T., Zhao, M., Grossklags, J., Chuang, J.: Given enough eyeballs, all bugs are shallow? Revisiting Eric Raymond with bug bounty markets. Journal of Cybersecurity (2017)
16. Mc Laughlin, H.: SMOG grading - A new readability formula. Journal of Reading 12(8), 639–646 (1969)
17. Ozment, A.: The likelihood of vulnerability rediscovery and the social utility of vulnerability hunting. In: Workshop on the Economics of Information Security (WEIS) (2005)
18. Ozment, A., Schechter, S.: Milk or wine: Does software security improve with age? In: USENIX Security Symposium (2006)
19. Ransbotham, S., Mitra, S., Ramsey, J.: Are markets for vulnerabilities effective? MIS Quarterly 36(1), 43–64 (2012)
20. Ratcliff, J., Metzener, D.: Pattern-matching: The gestalt approach. Dr Dobbs Journal 13(7) (1988)
21. Rescorla, E.: Is finding security holes a good idea? IEEE Security & Privacy 3(1), 14–19 (2005)
22. Senter, R., Smith, E.: Automated readability index. Tech. rep., DTIC Document (1967)
23. Shahzad, M., Shafiq, M., Liu, A.: A large scale exploratory analysis of software vulnerability life cycles. In: International Conference on Software Engineering (2012)
24. Zhao, M., Grossklags, J., Chen, K.: An exploratory study of white hat behaviors in a web vulnerability disclosure program. In: 2014 ACM CCS Workshop on Security Information Workers (2014)
25. Zhao, M., Grossklags, J., Liu, P.: An empirical study of web vulnerability discovery ecosystems. In: 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS) (2015)
26. Zhao, M., Laszka, A., Maillart, T., Grossklags, J.: Crowdsourced security vulnerability discovery: Modeling and organizing bug-bounty programs. In: HCOMP Workshop on Mathematical Foundations of Human Computation (2016)

A Example Bug Bounty Rule Statements

We list example rule statements below:

1. *"Please report serious vulnerabilities in our website (https:// staging.factlink.com), proxy (https://staging.fct.li), or other components (our annotation library, Wordpress plugin, browser extensions, and gems)"* (Factlink)

2. *"Please note that Binary.com's front-end code is open-sourced at [...] - please feel free to report any vulnerabilities found in this code by submitting a pull-request in github"* (Binary)

3. *"Not in scope: shopify.asia, go.shopify.com and investors.shopify. com are operated by third parties, and are not in scope"* (Shopify)

4. *"Any Sucuri customer website are out of the scope of this disclosure program"* (Sucuri)

5. *"Mattel websites and services are owned and operated by Mattel and are explicitly outside the scope of this bug bounty program"* (ToyTalk)

6. *"Please only use our staging environments for testing, they are otherwise identical to production"* (Factlink)

7. *"In general, anything which has the potential for financial loss or data breach is of sufficient severity, including: XSS, CSRF, Authentication bypass or privilege escalation, Click jacking, Remote code execution, Obtaining user information, Accounting errors"* (Coinbase)

8. *"We are generally not interested in DoS vulnerabilities that are perceived by a lack of rate-limiting or captcha. As a web-scale service, our threshold for rate limiting is higher than you would probably expect. Of course, if you think you have found an exception to this rule, please let us know"* (Automattic)

9. *"Please create a free account and (pen) test away our GhostMail, ChostChat and GhostBox"* (Flox)

10. *"Please note that automated testing is not permitted! System will ban you permanently if you do"* (DigitalSellz); *"If you employ automated scanning tools, their requests must be rate limited to not exceed 3 requests per second without prior approval"* (Vimeo)

11. *"Do not attempt to gain access to another user's account or confidential information"* (Adobe)

12. *"You are not allowed to conduct social engineering attacks against our support team"* (Coinbase)

13. *"While researching, we'd like to ask you to refrain from: [...] Any physical attempts against BitHunt property or data centers"* (BitHunt)

14. *"In order to encourage responsible disclosure, we promise not to bring legal action against researchers who point out a problem provided they do their best to follow the above"* (Openfolio)

15. *"You must comply with all applicable laws in connection with your participation in this program. You are also responsible for any applicable taxes associated with any reward you receive"* (Twitter)

16. *"Yahoo reserves the right to change or modify the terms of this program at any time"* (Yahoo)

17. *"Yahoo employees and contingent workers, as well as their immediate family members and persons living in the same household, are not eligible to receive bounties or rewards of any kind under the Yahoo Bug Bounty Program, whether hosted by Yahoo or any third party"* (Yahoo)

18. *"You must be eligible to work within the U.S.; meaning you are a U.S. citizen, a noncitizen national of the U.S., a lawful permanent resident, or an alien authorized to work within the U.S."* (Hack the Army)

19. *"You must be 18 years of age or older. Please be an adult when messaging us. We want to work with serious security professionals only"* (Envoy)

20. *"Share with us the full details of any problem found. Detailed steps on reproducing the bug. If valuable, please include any screen-shots, links you clicked on, pages visited, etc. Provide us with a concrete attack scenario. How will the problem impact Bookfresh or our customers? Put the problem into context"* (BookFresh)

21. *"Provide us a reasonable amount of time to resolve the issue before any disclosure to the public or a third-party"* (ownCloud)

22. *"Minimum reward is $100 for security vulnerabilities. The reward depends on the vulnerability severity and will be paid via HackerOne only. Every researcher with accepted vulnerability will be mentioned on http://hackerone. com/algolia/thanks"* (Algolia)

23. *"We only reward the first reporter of a vulnerability"* (DropBox)

24. *"Twitter will determine in its discretion whether a reward should be granted and the amount of the reward"* (Twitter)

25. *"Post on our Hall of Fame. Your very own Informatica Bug Bounty T-Shirt With More Awesome Swag to Come"* (Informatica)

26. *"Security and privacy are top priorities at Coursera. We believe that no technology is perfect and that working with skilled security researchers across the globe is crucial in identifying weaknesses in our technology"* (Coursera)

B Statistics for Programs Grouped by Description Length

Table 2. Statistics for Programs Grouped by Description Length

Number of Words	Number of Programs	Mean Number of Words	Mean Number of Bugs Resolved (Per Year)	Mean Number of Hackers Thanked (Per Year)	Mean Bounty Paid
0-250	13	200	49	34	225
250-500	36	340	36	28	392
500-750	16	633	67	48	59
750-	12	1011	189	123	105
Overall	77	481	69	48	250

Table 3. Statistics for Programs Paying a Minimum Bounty (Grouped by Description Length)

Number of Words	Number of Programs	Mean Number of Words	Mean Number of Bugs Resolved (Per Year)	Mean Number of Hackers Thanked (Per Year)	Mean Bounty Paid
0-250	8	211	74	49	366
250-500	15	318	13	10	941
500-750	12	630	68	49	79
750-	9	1042	194	143	140
Overall	44	532	76	55	437

Table 4. Statistics for Programs Paying No Minimum Bounty (Grouped by Description Length)

Number of Words	Number of Programs	Mean Number of Words	Mean Number of Bugs Resolved (Per Year)	Mean Number of Hackers Thanked (Per Year)	Mean Bounty Paid
0-250	5	182	10	8	0
250-500	21	355	53	41	0
500-750	4	641	66	46	0
750-	3	918	176	65	0
Overall	33	415	59	39	0

(Please note that the last column, which shows that no bounties were paid, is correct.)

C Statistics for Programs Grouped by Clauses

Table 5. Statistics for Duplicate Report, Legal Action, and Public Disclosure Clauses

Clause	Present	Number of Programs	Mean Number of Bugs Resolved (Per Year)	Mean Number of Hackers Thanked (Per Year)
duplicate	yes	35	117	79
	no	42	29	22
legal action	yes	17	134	95
	no	60	50	35
public disclosure	yes	38	102	73
	no	39	36	24

Table 6. Statistics for Staging Sites, Test Accounts, and Downloading Source

Clause	Present	Number of Programs	Mean Number of Bugs Resolved (Per Year)	Mean Number of Hackers Thanked (Per Year)
staging site	yes	5	51	41
	no	72	70	49
test account	yes	24	133	97
	no	53	40	26
source code	yes	13	40	34
	no	64	75	51

D Further Readability Analysis

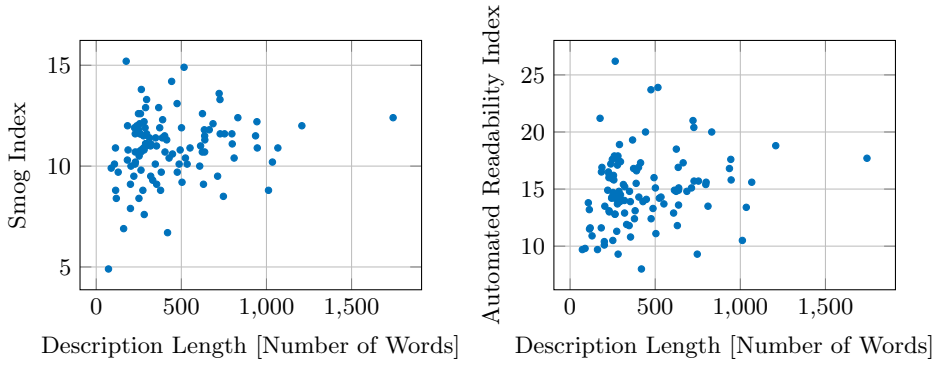

Fig. 6. Program descriptions' length and readability, measured using Smog Index [16] and Automated Readability Index [22].