Bug Bounty Programs for Cyber-Security: Practices, Issues and Recommendations

Suresh S Malladi, *Sam Walton College of Business, University of Arkansas*

Hemang C Subramanian, *College of Business, Florida International University*

Drawing upon crowdsourcing, Bug Bounty Programs (BBPs) are entering the mainstream security practice in organizations. We analyze five main areas of BBP practice namely: scoping of BBPs, timing of crowd engagement, submission quality, firm-researcher communication and hacker motivation. We discuss issues in each area and recommend practices to enhance BBP effectiveness. This paper informs research and practice about issues and best practices in crowdsourcing information security for timely discovery and remediation of vulnerabilities.

Keywords: Bug-Bounty, Hackers, Cyber-Security, Crowdsourcing

Introduction

Bug bounty programs (BBPs) are becoming a mainstay in the security strategy of organizations [1]. Firms such as Apple, Microsoft, etc. and institutions such as the Department of Defense (DoD) regularly operate BBPs. Until recently, dedicated security testing teams performed Blackbox, Whitebox and penetration tests etc. to discover vulnerabilities in firm's products. However, systems are becoming complex[2] and the nature of vulnerabilities is becoming unpredictable, thereby limiting the firm's ability to trace critical vulnerabilities[3]. Given this, firms are increasingly leveraging BBPs to crowdsource both discovery and fixing of vulnerabilities[4]. HackerOne, a leading BBP platform, summarized that high to critical vulnerabilities reported on its platform increased by 22% in 2017 with 24% resolved among them. Rewards are increasing to offer up to $75,000 for critical vulnerabilities while 80% of vulnerabilities reported by researchers were valid[14]. BBPs can be effective because firms (a) access a global pool of security researchers who compete with hackers in pre-empting discovery and exploitation of vulnerabilities (b) reward researchers only for valid vulnerabilities discovered. There are 3 categories of BBPs as shown below.

Table 1. BBP categories

Type	Examples	Characteristics
Institutional BBPs	Microsoft, Facebook, Google, US DoD	• Hosted directly by software vendors who set policies and compensation
Platform BBPs	BugCrowd, HackerOne, TippingPoint	• Legitimate intermediary host of simultaneous BBPs for multiple organizations. • Reward amount is determined by the host. • Jointly host BBPs for companies who run institutional BBPs as well as use platform BBPs under hybrid model
Private Intermediary BBPs	Zerodium, Sonosoft	• Purchase vulnerabilities from researchers to sell downstream. • Rewards higher prices compared to legitimate programs.

BBPs are distinct from traditional crowdsourcing because the problem space itself is unknown in information security. Intermediaries in traditional crowdsourcing help corporations to source solutions for well-defined problems by scoping the problems, broadcasting them and coordinating solution sourcing[2]. Here, the problem space is known, and the crowd is used to solicit solutions. An example is the Netflix Prize. However, BBPs are different; firms list rules for rewards and provide access to testing environments. Researchers identify problems in software products and frequently propose solutions. Hence, while traditional crowdsourcing explored only the solution space, BBPs operate when both problem and solution space are unknown. Table 2. depicts the security crowdsourcing possibilities for problem–solution scenarios.

Table 2. Security crowdsourcing mix with problem-solution scenarios

	Problem Known	**Problem Unknown**
Solution Known	• In-house software testing ➢ Black-box testing ➢ White-box testing	• Security research ➢ Simulation ➢ Mechanism design ➢ Root-cause-analysis
Solution Unknown	• In-house research-development • Outsourced security testing • Hackathons	• BBPs ➢ Open-ended ➢ Fuzzing competitions ➢ Invite-only BBPs

Despite this potential, comprehensive understanding of the usage of BBPs needs more analysis of extant practices to identify areas for improvement. While firms often treat BBPs like black-box (or outsourced) testing, BBPs pose unique challenges unknown in traditional crowdsourcing (or) outsourcing (e.g. hacker disappointment, retaliation, or, unsolicited exploits). Thus, certain practices limit their effectiveness. In this paper, we analyze 41 prominent BBPs using grounded theory and summarize the features adopted by existing BBPs and issues in their implementations. We then combine three decades of industry and research experience in information security, crowdsourcing, and digital platforms to suggest remediations that can proactively stretch BBP's practices to provide long-term defensive solutions beyond "vulnerability discovery"[5].

Research Methodology

We use grounded theory[6] to analyze existing BBP practices from 41 BBPs hosted as institutional BBPs or through intermediaries . These programs were selected based on (a) firm's total expected customer base (b) total number of vulnerabilities disclosed on BBP (c) length of the program (d) market capitalization, if applicable (e) potential practice maturity gauged by researchers. Firms in our focal analysis included Microsoft, Google, Facebook, Apple and Mozilla etc. which hosted institutional BBPs as well as participated in platform BBPs under hybrid model. Further, we analyzed firms/institutions such as the US Department of Defense, Netflix, Apache, Toyota, Starbucks etc. which hosted BBPs through two prominent platform-based intermediaries i.e. HackerOne and Bugcrowd. We analyzed these two intermediaries due to their reputation and scale. The five criteria mentioned above collectively signify the critical mass of vulnerability discovery, reputation of the BBP host which attracts high-quality researchers seeking prestige, longevity of the program, ability of the host to invest and maintain enduring standards for crowdsourcing and firms in varied industries using security crowdsourcing.

During analysis, both the authors used 10 focal BBP policy documents and independently developed open codes by observing patterns. For this pilot analysis, we used data from Google, Facebook, Microsoft and Western Union etc. which met the five criteria. Next, the authors independently coded relationships among the open codes to develop axial codes. These axial codes were grouped to form selective codes[6]. Once the coding schema was developed, both researchers independently coded 31 additional BBPs selected from institutional BBPs, BugCrowd's hosted BBPs and HackerOne BBPs. Selective coding continued till it resulted in five practice themes among the analyzed programs. We achieved an inter-rater reliability(IRR) of 98% and a Cohen's Kappa of 0.89[6]. We confined to 41 BBPs because patterns were repetitive and coverage for all codes emerged. Table 3 summarizes research methods used to develop the theory of BBP practice.

Table 3. Research Methods

Construct	Process	Research method
Practice areas (Themes)	Selective coding	Grounded theory
Features and sub-features	Axial codes and open codes	
Issues	Excluded features and practice areas from firms' current practice	Grounded theory, inductive reasoning
Recommendations for issues	Literature survey (academic research and industry reports) and author expertise	Inductive reasoning from literature, author expertise.

Below we present our results and the theory of BBP practice.

Theory of BBP Practice - Practice Areas, Issues and Recommendations

Each section below details the main practice area(theme), the corresponding features, issues and recommendations pertaining to BBP practice. We noted that firms are on a continuum in implementing these features in BBP practice areas; therefore, leaving room for improvement.

Scoping of BBPs

The feature descriptions pertaining to Scoping of BBPs are enumerated below

- **Single product and current version:** BBP scoped to test current version of products.
- **Old and new interdependencies:** Scope includes testing the product interdependencies with old and new product versions.
- **Third-party interdependencies:** Scope includes testing interdependencies with third-party products
- **Acquired firm interdependencies:** Scope includes testing interdependencies with acquired products.
- **Generic products:** Scope includes all the firm's products and commissioned versions.

Issues and Recommendations:

Firstly, we noted that BBP operators often focus exclusively on new products due to resource constraints and the novelty factor which attracts researchers [7]. However, it is well-documented that many vulnerabilities are often discovered in legacy systems [8]. Hence, focusing on new products creates blind spots in older products that are easily exploited. Second, not testing dependencies and interrelationships with other products in firm's portfolio including acquired products and products imbibing third-party components can create security loopholes. We rarely observed this emphasis and hence recommend that when launching BBPs, firms should audit products, interrelationships, and potential areas where gaps can emerge amongst both own and external products. Firms should create a clear blueprint which identifies product areas that need more security testing, and then prioritize specific areas as touchpoints for crowd collaboration. Third, extremely narrow BBP focus such as confining to UI testing creates traps, when both internal and external experts duplicate efforts by testing similar aspects. External researchers possess expertise in many areas which are unavailable internally. To benefit from crowdsourcing, vendors should use BBPs to look beyond discovering vulnerabilities. Firms should seek expertise in developing tools and techniques that aid in both discovering and fixing vulnerabilities and in automating the discovery process. For example, BugCrowd solicits crowd expertise for developing fuzzing and debugging tools to automate vulnerability discovery. Tools for vulnerability discovery uncover previously unknown vulnerabilities in mature products and prevent attackers from hoarding them[5]. We recommend a process progression from discovery until automation of discovery as shown below for BBPs to become more effective additions to security practice.

Vulnerability discovery → Vulnerability fixing → Vulnerability detection tools → Discovery automation

To accomplish this, firms should publish the scope, integration, and compatibility requirements and prioritize areas where vulnerabilities can inflict damage. In addition, firms should encourage researchers with additional incentives to contribute fixes and tools that automate detection. We observed that exemplar firms such as Microsoft adopt such proactive practices while firms that are maturing in their practices may restrain for fear of soliciting attacks.

Timing of Crowd Engagement

The feature descriptions pertaining to timing of crowd engagement are enumerated below:

- **Post-Release:** BBPs are hosted after launching products into market. They can be open-ended or invite-only (for some minor releases).
- **Beta testing:** BBPs coincide with beta testing phase of projects. Most beta-testing BBPs are invite-only to engage pre-selected researchers.
- **Pre-Beta testing:** BBPs hosted during alpha or development phase of projects. This requires specialist researchers and are usually run as short-term programs, e.g., fuzzing competitions and invite-only programs. This is usually an exception.
- **Short-term competition:** BBP is short-term and time-boxed such as flex bounty or fuzzing competition.
- **Ongoing security audit:** Crowd gets ongoing access to test all products to find evolving vulnerabilities. This complements internal testing.

Issues and Recommendations:

We observed that many firms often engage with external researchers either late into the development lifecycle or after the product launch. Such delays may result in the firms often trailing hackers in detecting vulnerabilities. While early exposure to software before beta-testing through open-ended BBPs can attract unwanted attention to security flaws, engaging pre-qualified researchers through invite-only programs before/during beta testing is effective to recognize vulnerabilities early. For example, Microsoft used crowd to test Internet Explorer 11 from early development until beta-testing[9]. Depending on the lifecycle methodology adopted (e.g. waterfall, iterative, or Scrum) firms can pre-qualify skilled researchers to engage them early. This fosters a loyal and skilled researcher pool to attain scale in the program and discover vulnerabilities sooner. The following list recommends the BBP engagements during a firm's software product life-cycle.

- **Requirements and Design** : Test plan and budgeting for BBPs
- **Development Phase**: Fuzzing competition (private BBPs)
- **Public and Private Beta:** Invite-only BBPs
- **Product Market Launch**: either Public or Invite-only BBPs
- **Post-Release**: Open-ended or invite-only or fuzzing competitions

Summarily, while BBPs provide opportunities to enhance security, timing the external engagement is very critical. Once planned and budgeted, crowd collaboration through BBPs at every phase of product development will (a) challenge and motivate researchers, (b) develop an inclusive approach to internalize external talent, (c) enrich engagement by explicitly discussing security problems with researchers and (d) develop specific solutions for priority issues. Firms would benefit by building foundations for long-term defensive strategies rather than episodic engagements[1].

Submission Quality

Feature descriptions pertaining to submission quality in BBPs are listed below.

- **Reputation model:** Having a rating and reputation model for assessing reporters who consistently submit valid reports; to reduce false submissions.
- **Rate-limiting:** Rate-limiting submissions from individual reporters to significantly reduce false positives.
- **Promotional and Talent incentives:** Award additional incentives to first-time participants, and offer talent incentives such as retention bonuses, Swags etc.
- **Exploit proof:** Researchers should provide 'proof of concept' test case as a pre-requisite for high-quality submission.
- **Validation rewards:** Higher quality exploits receive higher rewards; pre-empts false positives.
- **Prioritization:** BBP policies explicitly prioritize vulnerability categories important to the firm. High priority vulnerabilities command higher payouts. Programs should list bounty priority e.g. P3-P4-P5 bugs are ineligible.
- **Differential incentives:** Reward policies that provide higher rewards for higher quality and accuracy. For e.g. if two researchers submit similar vulnerabilities, researcher who has

consistently submitted higher accuracy vulnerabilities gets higher rewards.

- **Process for acceptance:** A transparent process notifying researchers of the different stages of vulnerability acceptance and reward status.

Issues and Recommendations:

BBPs attract large numbers of invalid, or partial submissions (i.e. 50%-70% per HackerOne). While firms attempt to improve response times for each submitted vulnerability and validate/invalidate reports, researchers increase their submissions with hopes of earning more compensation. Quality issues arise from misaligned motivations because researchers are more mindful of numbers than quality of submission. To correct this imbalance, reporting and governance must be streamlined to improve submission quality.

From reporting standpoint, firms should encourage researchers with higher rates of valid submissions to submit even more. Often, platforms limit the number of reports that are submitted within a timeframe, e.g. HackerOne's rate-limiter model [10]. While rate-limiting may be effective, governance policies should not constrain participation. Instead, having differential incentive policies which reward higher accuracy can increase valid submissions. Alternately, firms can penalize (or blacklist) researchers for excessive invalid submissions. This approach improves quality by forcing researchers to self-regulate their submissions. However, penalizing can also be detrimental if researchers hesitate to submit work for fear of penalty.

From governance standpoint, some companies create invite-only BBPs instead of open programs[7]. Invite-only programs either use the firm's own contacts list or are managed by intermediaries to track highly-skilled researchers and invite them to participate. Table 4 summarizes the variants of BBPs.

Table 4. Variants of BBPs

Type of BBP	Description	Phase
Invite-only	BBP 'variant' managed either by the firm or by platform intermediary wherein only top contributors/researchers are invited to participate. This is usually expensive because vendors pay a premium to attract talent.	Beta launch or immediately after product launch
Fuzzing competition	This complements internal testing. Low risk but firms must provide access to internal testbeds and test tools.	Pre-Beta or Post-launch
Open-ended BBP	Firms run ongoing BBPs after product launch.	Post-launch
Short-timeframe competitions (e.g. Bugathon)	Special instances such as product integration or new module/app release prompts a short-term event. Often used to scout for specialized talent amongst researchers.	One-time competitions post-launch etc.

Using crowd to find and fix vulnerabilities leads to four uncertainties. First, outsiders access internal systems during testing, making the systems susceptible to live attacks. Creating separate test environment for researchers reduces risk to in-service systems such as cyber-physical systems[2]. Second, the effectiveness of any vulnerability fix is uncertain until the system is under attack. Firms should also fix vulnerabilities frequently to reduce their relevance.

Third, internal engineers or external researchers may find vulnerabilities and withhold reporting them, known as the cobra-effect. While cobras cannot be eliminated, we suggest two counter-measures. First is about timed incentivization i.e. vendors can formulate options-based contracts to pay only if no cobra is detected within a specific timeframe. Second is to redesign compensation mechanisms so that developer payments are reduced upon detection of cobras[11]. Further, auditing of vulnerabilities and holding developers accountable prevents internal cobras and protects the development process.

Another sought-after defense strategy is to eliminate an entire class of vulnerabilities (e.g. SQL injection or stack-memory leaks). For example, Microsoft and Adobe redefined their BBPs and shifted focus from vulnerability discovery to soliciting defense ideas which address entire classes of vulnerabilities. Microsoft's Blue Hat Prize contests reward mitigation bypasses for memory-based attacks, thereby handling memory-based vulnerabilities[11]. Finally, BBPs do not replace internal testing but only complement internal practices. A standardization of internal practices is the best defense for firms [12].

Firm-Researcher Communication

Feature descriptions pertaining to Firm-Researcher communication in BBPs are enumerated below.

- **Reporting guidelines:** Firms should formulate clear guidelines for reporting vulnerabilities. Contact information for reporting is made clear. Firms sometimes include contact details e.g. email ID's etc. for the process handlers. Researchers should be provided with a testing environment on which to test the vulnerability and an email-alias for the researcher to test yet remain anonymous. Forfeiture of legal pursuit upon vulnerability disclosure is sometimes made explicit to assure researchers.
- **Reward timeline:** Clearly mandated timeline for decision regarding validation and reward.
- **Compensation(Low):** Compensation terms can be very low compared to the exploit's possible damage.
- **Transparent reward mechanism:** Transparency into reward mechanism by providing a direct point of contact, etc.
- **Unclear reward process:** No clear process stating the reward criterion, reward decision, etc.
- **Extra credit for vulnerability fixing:** Provide extra credits or bonus for suggesting fixes to vulnerabilities.
- **Reward ranges:** Explicitly state price ranges by vulnerability category. If possible, state mean reward ranges so that researchers can negotiate

Issues and Recommendations:

Firms should recognize that BBPs are not hands-off efforts. Many negative incidents arise when researchers face obstacles in reporting discoveries. Firms should dedicate technical and human resources to handle, analyze, and follow-through every valid submission in a timely manner. If resources are not available, firms can choose to host invite-only BBPs. But BBPs without resource commitment will be detrimental. Poor communication can harm an organization’s reputation. For example, when Oracle Corporation’s chief security officer warned researchers against reverse engineering code, hacker resentment forced Oracle to withdraw the statement. In many cases, lack of responsiveness or refusal (even legitimate) by the firm to disclose a latent vulnerability publicly, can foster resentment or sabotage. To counter the issues with communication, we recommend four practices.

First, the reporting process should be simplified. Researchers should be guided on how to report and reproduce issues without violating laws or the firm’s policies. Alternately, the firm could use platforms (e.g. HackerOne) to manage the submissions process.

Second, while researchers should not be privy to the reward review process, trust issues arise if submitters evaluate vulnerabilities differently than the firm. Yahoo! was criticized when it offered $12.50 merchandise coupons to researchers who reported two critical XSS vulnerabilities. Explaining reward ineligibility is also crucial to foster trust and clarity. Clear metrics based on bug priority, difficulty to fix, business impact, etc. will help the firm to be fair. For e.g. while Facebook does not state reward eligibility and how reward decisions are made, their policy outlines how different parameters affect rewards (e.g., business impact, quality, value of the target).

Third, transparency and responsiveness will build trust. BBP guidelines should explain the structure, scope, and products covered. This includes range of compensation and average prior compensation. The range of compensation and timelines of BBP are also important to manipulate behavior. Tarsnap allocated higher bounties during pre-release phase, and Microsoft limited its BBPs to pre-release period. Firms should clarify how they reward overlapping submissions. Facebook rewards the first responder, whereas Tarsnap splits the bounty when submissions overlap.

Fourth, since many researchers are international, heavy legal parlance or too many conditions will preclude crowd participation. Minimal governing elements without legalese and insisting responsible multi-lingual disclosure would support the practice. Assurances of no legal pursuit lowers uncertainty and increases trust with researchers.

Upon maturation of the program, we recommend firms should solicit CASE tools, practices and techniques to augment existing internal practices.

Managing Hacker Motivation

Feature descriptions pertaining to managing Hacker Motivation using BBPs are enumerated below.

- **Commensurate Reward structure:** Rewards should be commensurate and competitive for similar vulnerabilities when compared to other BBPs.

- **Validation of vulnerabilities:** Validation of reported vulnerabilities and reward decisions must be timebound and transparent.
- **Public Recognition:** Recognize publicly and privately the contributions of researchers.
- **Hall of Fame:** List top contributors on a hall of fame. List new comers contributing vulnerabilities.
- **Reputation scores:** Upon seeking researcher permission, list all contributors by assigning a rank/reputation score to them.
- **Dynamic Compensation:** Dynamic reward ranges instead of fixed dollar amount for a particular class of vulnerability; final award determined based on complexity of vulnerability and quality of reporting
- **Price Negotiation Ability:** Possibility to negotiate prices based on severity of vulnerabilities discovered; if approved.
- **Pre-selected Researcher pool:** Creating a high-quality talent pool by identifying & pre-selecting exceptional researchers; also serves as recognition of talent.

Issues and Recommendations:

While scale is important for BBP operators to attract talent, and in the long run decreases operating costs to make the platforms viable, large scale BBPs with wide scope can cause fleeting loyalty. Researchers scout for frequent rewards and for creating a reputation in the research community. Increasing chances to earn both helps retain talent. In parallel, not cultivating a loyal researcher pool and internalizing it to a firm's strategy is as unproductive as a fleeting crowd. Invite-only BBPs can create a committed base of researchers [10], but the effectiveness of invite-only or open-BBPs still depends on challenging and motivating the researchers. Researchers can be motivated only when they work on high priority challenges. Firms should signal that the engagement is not only about testing products but also creating safe products. By explicitly recognizing that researchers possess more capabilities, researchers should be invited early in the lifecycle and given a voice to recommend ways to find new vulnerabilities and to suggest best practices. For example, Mozilla Foundation incentivizes finding issues in Firefox by increasing payments depending on the bug's criticality and quality of reporting. Mozilla offers higher rewards if researchers help patch such issues [13].

A well-defined scope will synchronize the problem space and solution space, thereby assisting to receive the maximum number of exploits from focused experts. Firms can run time-boxed tournaments with well-defined scope, which is a cost-effective way to identify vulnerabilities. For example, Google runs "*pwnium*" browser penetration contests that attract browser penetration experts to find issues. Similarly, organizing flex bounties, which can be 2-4-week, short-term bounties with a small focus group of researchers, fosters a milestone-driven environment.

While firms award non-financial incentives like reputation scores, we observed that unlike in other forms of crowdsourcing, financial compensation is the sole motivation for researchers. Higher payments allow to recruit researchers and to compensate them for their skills and costs in discovering vulnerabilities. High bounties also signal the firm's commitment to BBPs. Furthermore, higher rewards can manipulate behaviors and potentially prevent researchers from engaging in black market trade or moving to another BBP.

A few guidelines about rewards are: First, payments should be high to entice researchers if systems handle mission-critical data [14]. Second, if the firm has advanced process maturity, then payments should be commensurate to hire the best talent that understands complex processes[15]. Finally, if testing requires situational knowledge (like specific setups), compensation should consider time and effort involved. Beyond compensation, firms should support researchers with necessary technical and situational knowledge. Finally, payments should be used to encourage researchers to fix vulnerabilities in addition to discovery. Paying disproportionately large rewards for discovering vulnerabilities reduces incentives to fix them because researchers migrate to other opportunities where they earn compensation for their discoveries.

Conclusion

BBPs have conceptual origins in crowdsourcing and outsourcing, yet they pose unique challenges which arise from the unknown problem (and/or solution) space in security (Ref. Table 2). Our paper lists five major practice areas researched from 41 BPPs. These are: Scoping, Timing of Crowd Engagement, Submission Quality, Firm-Hacker Communication and Managing Hacker Motivation. We list key features in existing programs, issues within each practice area and recommendations for addressing them. Appendix A summarizes our recommendations and the exemplar firms practicing them. Finally, a recent industry survey found that BBPs are increasing in scale with self-taught researchers contributing from 100 countries, especially emerging economies[14]. If firms can implement right supportive practices, they can create a skilled pool of researchers to collaborate and preempt security issues societies face today.

References

1. Denning, D.E.: 'Toward more secure software', Communications of the ACM, 2015, 58, (4), pp. 24-26
2. Biro, M., Mashkoor, A., Sametinger, J., and Seker, R.: 'Software Safety and Security Risk Mitigation in Cyber-physical Systems', IEEE Software, 2018, 35, (1), pp. 24-29
3. McGraw, G.: 'Building secure software: better than protecting bad software', IEEE Software, 2002, 19, (6), pp. 57-58
4. LaToza, T.D., and van der Hoek, A.: 'Crowdsourcing in software engineering: Models, motivations, and challenges', IEEE software, 2016, 33, (1), pp. 74-80
5. Ghosh, A.K., Howell, C., and Whittaker, J.A.: 'Building software securely from the ground up', IEEE software, 2002, 19, (1), pp. 14
6. Strauss, A., and Corbin, J.: 'Grounded theory methodology', Handbook of qualitative research, 1994, 17, pp. 273-285
7. Hatton, L., Spinellis, D., and van Genuchten, M.: 'The long-term growth rate of evolving software: Empirical results and implications', Journal of Software: Evolution and Process, 2017, 29, (5)
8. Ruohonen, J., Rauti, S., Hyrynsalmi, S., and Leppanen, V,:'A Case Study on Software Vulnerability Coordination', Information and Software Technology, Elsevier, June 2018
9. Rashid, F.: 'Extortion or fair trade? The value of bug bounties', InfoWorld, 2105
10. Zhao, M., Laszka, A., and Grossklags, J.,: 'Devising effective policies for bug-bounty platforms and security vulnerability discovery', Journal of Information Policy, 2017, 7, pp.372-418.
11. Egelman, S., Herley, C., and Van Oorschot, P.C.: 'Markets for zero-day exploits: Ethics and implications', in Editor (Ed.), NSPW, ACM, 2013, pp. 41-46
12. Evans, D., and Larochelle, D.: 'Improving security using extensible lightweight static analysis', IEEE software, 2002, 19, (1), pp. 42-51
13. Finifter, M., Akhawe, D., and Wagner, D.: 'An Empirical Study of Vulnerability Rewards Programs', in USENIX Security Symposium, Berkeley, CA, 2013, pp.273-288
14. Help Net Security.: 'Hacker-powered security is reaching critical mass', HelpNetSecurity Online, July 2018.
15. Siponen, M.: 'Information security standards focus on the existence of process, not its content', Communications of the ACM, 2006, 49, (8), pp. 97-100

Appendix A: Best Practices for Enhancing BBP Effectiveness

Below, we summarize BBP best practices (with exemplar BBPs) for each practice area, which if followed can increase program effectiveness.

Recommendations/Best Practices	Exemplar BBPs
Scoping of BBP	
• Strategize BBPs to audit beyond new products. • Focus on old and new product versions, interdependencies amongst own products, versions and third-party software • Create a holistic blueprint of products and dependencies, identify all versions, proprietary and open-source components, and integration touchpoints that need special attention.	Google's open-ended BBP
• Use BBPs to develop tools and techniques that automate vulnerability discovery (e.g., fuzzers and debuggers)	Bugcrowd
Timing of Crowd Engagement	
• Engage researchers early in the development lifecycle, for example, pre-beta or beta testing phase	Adobe, Google, Tezos Foundation
• Run ongoing open-ended or invite-only programs to solicit long-term defensive strategies against entire classes of vulnerabilities	Microsoft's Blue Hat Prize
Submission Quality	
• Tie rewards and bonuses to submissions with exploitation techniques and fixes (even exclusively).	Vimeo, MS-Windows BBP, Android BBP

• Institute differential rewards for high-quality reports with attack scenarios and vulnerability duplication	
• Penalize invalid reports to deter indiscriminate submissions. Award reputation scores. • Limit the number of reports a researcher can submit within a timeframe to enhance accuracy	HackerOne, Google
• Proactively scope high-risk areas for researchers to test (where critical vulnerabilities persist)	CoinBase, Massachusetts Institute of Technology(MIT)
• Host invite-only BBPs and identify top-tier researchers for continuous engagement.	Google's Pwnium browser penetration contest and ongoing timeboxed BBPs
Firm-Hacker Communication	
• Devise clear reporting guidelines and assign dedicated resources to handle vulnerabilities and researcher communication	Apache, Apple, Facebook
• Maintain transparency and responsiveness with clear ETA about vulnerability validation and reward status	Etsy, Quora, GitHub.
• Ensure no legal action for reporting	Facebook, Netflix, HubSpot, GitHub
• Highlight upfront exceptions ineligible for reward, for example, low-criticality vulnerabilities as ineligible and issues reproducible in single domain being eligible for only one award. This enhances focus and mitigates researcher resentment	Western Union
Managing Hacker Motivation	
• Institute competent pricing w.r.t. to other BBPs and dynamic pricing based on quality of reports. Provide incentives for assistance in fixing vulnerabilities • Maintain transparency in reward ranges and eligibilities	Mozilla, Apache
• Nurture talent pool through invite-only BBPs toward continuous engagement with identified researchers	Google Chrome penetration contests and ongoing timeboxed BBPs
• Recognize contributions publicly • Support researcher knowledge requirements	GitHub, Apple, Under Armour

Suresh Malladi is an assistant professor of Information systems at Sam Walton College of Business, University of Arkansas. Suresh holds a Ph.D. in Information systems from University of Michigan's Ross School of Business. Suresh has published in premier academic and practitioner journals on information systems and software engineering. Prior to his academic career, Suresh worked for several years leading security projects at CSC.

Hemang Subramanian is an assistant professor of information systems and business analytics at FIU's Business School. Hemang holds a Ph.D. in Information Technology Management from Georgia Tech. His work has been published in Information Systems Research and Communications of the ACM. Prior to his academic career, Hemang worked for more than a decade at IBM and Yahoo! in security engineering and managerial roles.