Global Information Assurance Certification Paper

BUG BOUNTY PROGRAMS

Enterprise Implementation

GIAC (GCFA) Gold Certification

Author: Jason Pubal, jpubal@mastersprogram.sans.edu
Advisor: Tanya Baccam

Accepted: December 2017

Abstract

Bug bounty programs are incentivized, results-focused programs that encourage security researchers to report security issues to the sponsoring organization. These programs create a cooperative relationship between security researchers and organizations that allow the researchers to receive rewards for identifying application vulnerabilities. Bug bounty programs have gone from obscurity to being embraced as a best practice in just a few years: application security maturity models have added bug bounty programs and there are standards for vulnerability disclosure best practices. Through leveraging a global community of researchers available 24 hours a day, 7 days a week, information security teams can continuously deliver application security assessments keeping pace with agile development and continuous integration deployments complementing existing controls such as penetration testing and source code reviews.

I started a bug bounty program at a fortune 500 global financial company. This paper reflects the research used to justify the program, the project to implement it, operational processes in use, and lessons learned along the way.

Introduction

Bug bounty programs are incentivized, results-focused programs that encourage security researchers to report security issues to the sponsoring organization. They create a "cooperative relationship between security researchers and organizations that allow researchers to receive rewards for identifying application vulnerabilities" without the risk of legal action (Bugcrowd, 2015, p. 1). This relationship helps companies to identify and resolve security vulnerabilities they might not otherwise find

Traditional application testing models do not scale well to meet the demands of modern development methodologies. They are expensive, resource intensive, and are only a point in time assessment. Agile development and DevOps have given us faster development and deployment cycles. Vulnerabilities occasionally slip through even the most mature application security programs. Modern application security assessments need to be on demand and continuous with an instant feedback loop delivered to developers to keep pace. Bug bounty platforms offer a worldwide community of researchers working 24/7; leveraging this community can supplement an organization's application security program, ensuring a known quantity finds those vulnerabilities before they are exploited by malicious actors.

It is critical that organizations have a way for external parties to tell them about potential security vulnerabilities, leveraging their internal vulnerability management process to triage and remediate them. However, "94 percent of the Forbes Global 2000 do not have known vulnerability disclosure policies" (HackerOne, 2017, p. 19). According to Bacchus, "when starting a bug bounty program, you're essentially adding a new stream of bugs into your existing vulnerability management process" (2017, p. 10). Incentivizing those vulnerability reports by paying researchers who submit unique, actionable bugs enables a company to crowdsource security testing to thousands of individuals looking to make our connected Internet ecosystem more secure.

1.1. Vulnerability Management

Vulnerability management plays a crucial role in finding a variety of technical vulnerabilities in an environment, prioritizing the resulting risk, and improving the overall security posture by addressing those likely to lead to incidents. A vulnerability is

"a weakness in the software or hardware that allows use of a product beyond its design intent with an adverse effect on the software, system, or data" (Foreman, 2010, p. 61). An exploited vulnerability could result in a disruption in availability or loss of data confidentiality and integrity. Vulnerability is a component of risk. Harris (2012) defines risk as "the likelihood of a threat agent exploiting a vulnerability and the corresponding business impact" (p. 57). Expressed as a formula, that is:

$$Risk = \frac{Threat \times Vulnerability}{Countermeasures} \times Value$$

Breaking down the components of risk, threats are someone or something that can do harm. Vulnerabilities are the weaknesses of an asset that allows the threat to cause harm. Countermeasures are precautions that have been taken. Value is the monetary worth of the asset representing the potential loss as the result of an incident. Information security is about managing risks to sensitive data and critical resources. There will always be some amount of risk present, the role of information security is to bring it in-line with organizational policies.

Vulnerability management is the "cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities" (Foreman, 2010, p. 1), and can be an effective means of managing the risk to an enterprise. An organization's vulnerability management practices often include activities such as the following:

- Infrastructure Vulnerability Management Scanning
- Source Code Review
- Penetration Testing

1.2. Vulnerability Disclosure

New vulnerabilities are discovered and published daily. An individual or organization may discover a security flaw through casual evaluation of a product, by accident, or as a result of focused analysis and testing. Once a vulnerability is discovered, what can the researcher do with their discovery? It largely depends on their motivation. This kind of security research is done by: companies wanting to ensure either the products they are creating or the products they are using are secure, hobbyists wanting to

learn information security or gain notoriety, security firms wanting to profit from or market their knowledge of security flaws, criminal organizations wanting to defraud the public for financial gain, and nation states wanting to either defend their country or find ways to subvert others. While some motivation is malicious, a lot of this research is done with the intent of discovering a vulnerability and reporting it to the software's creator or making it publicly known so that it can be fixed to make the product and the interconnected Internet ecosystem more secure. However, providing the information to the vendor or the public carelessly can be dangerous. Inappropriate disclosure of a vulnerability could cause a delay in resolution or give attackers the information they need to exploit it.

If a company is not prepared to receive vulnerability reports, there can be a long delay before it ends up with someone who can act on it. With no clearly defined mechanism such as an easy to find web portal or email address, vulnerability finders may try to use other communication mechanisms such as a customer service email address or marketing social media accounts. There is a chance the person who initially receives it might not know how to handle it and may simply discard the information. Those reports might not make it to information security personnel at all. And if they do, without a risk-based triage process in place, it is likely to get treated as a high priority event or security incident. This is inefficient and wastes internal resources. Incidents are short-lived events, so occasionally items fall through the cracks that a dedicated and specific process would handle.

Because of the dangers of mishandled vulnerability reports, various organizations have published best practices for safely reporting vulnerabilities and for how vendors can receive and act on those reports in a transparent and consistent fashion. For example, the National Infrastructure Advisory Council's Vulnerability (NIAC) published the Disclosure Framework published in 2004, and the International Organization for Standardization's (ISO) published the Vulnerability Disclosure Standard in 2014. According to HackerOne, "federal agencies and standards bodies recommend vulnerability disclosure policies" such as the National Highway Traffic Safety

Administration (NHTSA), Food and Drug Administration (FDA), and Federal Trade Commission (FTC) (2017, p. 21).

According to the ISO Vulnerability Disclosure Standard, vulnerability disclosure is "a process through which vendors and vulnerability finders may work cooperatively in finding solutions that reduce the risks associated with a vulnerability" and "it encompasses actions such as reporting, coordinating, and publishing information about a vulnerability and its resolution" (2014, p 1). Organizations that create software and are responsible for the security of their products and services can use vulnerability disclosure to learn about security issues and to help create resolutions or mitigations for those flaws.

The ISO Vulnerability Disclosure Standard tells vendors that in order to deal adequately with vulnerability reports, they need to have a mechanism for securely receiving those reports and a policy for how they are going to deal with them. The standard states, "Since vulnerability information may be used to attack vulnerable products and online services, sensitive vulnerability information should be communicated confidentially. Vendors may wish to provide secure confidential methods for finders to report vulnerability information" (IOS/IEC 29147:2014, p. 9). This can include publishing a Pretty Good Privacy (PGP) public key for the researcher to encrypt the information before reporting or have a secure TLS encrypted form available online. Also, "Vendors should define their responsibilities in a vulnerability disclosure policy" (IOS/IEC 29147:2014, p. 10). The policy can contain items such as how to contact the vendor, communication expectations, what should be contained in the report, and services that are in as well as out of scope.

1.3. Vulnerability Management & Disclosure Interaction

Vulnerability management and vulnerability disclosure are related processes. While vulnerability management deals with the analysis, prioritization, and remediation of security flaws regardless of the source of vulnerability, vulnerability disclosure deals with the interface between organizations and third parties who discover and report potential vulnerabilities. This interaction is shown below in Figure 1.

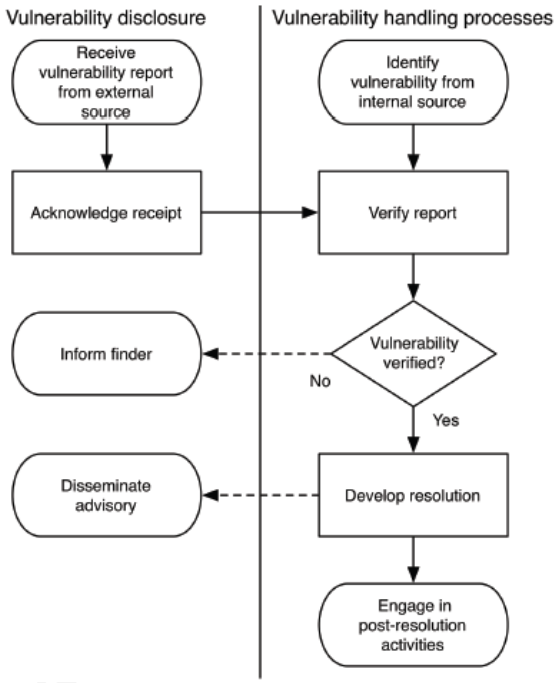

Figure 1: Vulnerability disclosure and vulnerability handling interaction. From "IOS/IEC 29147:2014." 2014.

Large companies are often mandated by a compliance regime to have a vulnerability management process in place. For example, the Payment Card Industry's (PCI) Data Security Standard (DSS) requires merchants that accept credit or debit cards as payments to "maintain a vulnerability management program" which includes both patch management and secure application security development practices (2016, p. 5). Vulnerability disclosure essentially sits on top of a vulnerability management process – it becomes another one of many vulnerability discovery sources that vulnerability management seeks to resolve.

1.4. Incentivizing Security Research & Vulnerability Disclosure

How can vulnerability disclosure be leveraged to reduce risk in an enterprise? What behaviors could be encouraged to maximize that risk reduction? Table 1 outlines a number of incentives that would encourage desirable behaviors.

Behavior	Incentive
If a security vulnerability is found, report it.	Pay a monetary reward for reported vulnerabilities.
Have security researchers assess an	Make that application eligible for rewards by

application for security flaws.	putting it in scope of the vulnerability rewards program.
Remediating higher criticality vulnerabilities reduces risk more. Discover and report high criticality vulnerabilities.	Pay a higher monetary reward for higher criticality vulnerabilities.
Focus on testing the security in specific parts of an application, such as new functionality in an upcoming release.	Pay a higher monetary reward for areas of focus.
Do not publicly disclose vulnerability information until the flaw is fixed.	Create a policy that a researcher must agree to before participating. If the researcher does not follow the rules, they are illegible for future rewards.

Table 1: Behaviors and incentives.

Building an incentives program around vulnerability disclosure allows an enterprise to effectively crowdsource application security assessments. By paying for reporting security flaws, security researchers are incentivized to spend time discovering vulnerabilities in a particular application. Offering higher amounts of payments for flaws that are more desirable incentivizes researchers to spend their time discovering and reporting those vulnerabilities. Publishing a policy that researchers must agree to with retribution of not being able to participate if those rules are not followed incentivizes a host of other desirable behaviors.

1.5. Bug Bounty Programs

A bug bounty program is "an incentivized, results-focused program that encourages security researchers to report security issues to the sponsoring organization" (Bugcrowd, 2016, p. 4). It is a rewards program offered by an organization to external parties, authorizing and incentivizing them to perform security testing on the organization's assets.

Bug bounty programs seem to have gone from novelty to get embraced as a best practice in just a few years. Bugcrowd sees the trend of non-technology companies adopting bug bounty programs saying that "organizations from more 'traditional' industries have seen year over year growth of over 217% on average, including financial services, automotive, healthcare, education, telecommunications" (2016, p. 11). The Building Security in Maturity Model added operating a bug bounty program in a recent update, citing that 4% of its participants have programs (McGraw, 2015, p. 19). The

number of companies using crowdsourced application assessments is growing in a trend that has gained momentum in recent years.

1.5.1. Benefits

In their 'The State of Bug Bounty' report, Bugcrowd presented the results of a survey that elicited what companies running a bug bounty program found their benefits to be. The survey states,"The top three points of perceived value of bug bounties are the diversity in skill sets and methods used by hackers, the volume of bug hunters testing their applications and the pay for results model" (2016, p. 8). Other responses included positive external marketing of their security program and building a relationship with the security researcher community.

To further enumerate the benefits, Table 2 presents comparison of a bug bounty program to a more traditional security assessment.

	Penetration Testing	**Bug Bounty Program**
Time	A penetration test is a time-bound activity, typically lasting one or two weeks.	A bug bounty program is not time bound. A researcher can spend as long as they want to pursue an attack vector of interest. This is more akin to how an actual attacker would find a flaw.
Personnel	Depending on the scope and time allotted, a penetration test is generally assigned one or two people.	There are no personnel constraints to a bug bounty program. Researchers are like an elastic security team that needs to be incentivized. If the right incentive is provided, tens or hundreds of researchers might participate.
Cost	A penetration test has a set scope, number of resources, and cost. For a two-week engagement, one may pay $20,000. There is no guarantee that the penetration test will find any valid vulnerabilities.	Researchers are only paid for verified, original findings. They do not get paid for their time or for findings that another researcher already reported. So, only unique, valid vulnerabilities cost money.
Methodology	A professional penetration tester is going to follow a standard	Researchers will likely not follow a standard methodology. Testing

	methodology, and conduct similar testing each assessment.	styles will vary greatly per researcher, potentially leading to creative and out-of-box testing that finds new and unique vulnerabilities.
Tools	A penetration testing team is likely to have their preferred toolset. There will be some variance in what tools are used out of the toolset for a given test. Following a methodology, they are likely to use a similar toolset for each assessment.	There will not be a standard toolset across all of the security researchers involved. So, the variance in tools used will be significantly greater.
Scope Coverage	Professional penetration testers typically follow some standard methodology that provides assurance that the entire breadth of the scope was covered. However, because of the time constraint, there might be items that did not get the depth of coverage needed to find something.	A security researcher might not follow any methodology. There is no assurance of breadth of coverage. However, because a bug bounty program is typically not time-bound, what is covered is likely to get covered in greater depth.

Table 2: Penetration testing vs bug bounty.

1.5.2. Augmenting Pentesting with Bug Bounties

Traditionally, one of the most influential and effective ways to provide security feedback in the development lifecycle is via penetration testing. During penetration testing, penetration testers manually test the application using real-world attack scenarios resulting in high fidelity, actual findings of not only how the application is vulnerable but what the impact of that security flaw is. For example, an online shopping cart application is vulnerable to SQL injection, and because of that, the penetration tester was able to exfiltrate users' names, passwords, addresses, and credit card numbers. Because it is so effective, it has been adopted by various compliance regimes. PCI DSS, for example, requires that in scope applications are pentested "at least annually and after any significant infrastructure or application upgrade or modification" (2016, p. 101).

In a waterfall model, where there is a series of security touchpoints along the software development lifecycle, penetration testing is done towards the end right before an application is deployed to production. It is a deployment gate that cannot be passed

until pentesting is complete. If a major security flaw is discovered, the deployment is delayed until the code can be fixed. A manual process that usually takes a week or two and has some chance of delaying an application deployment even more is the antithesis of devops. In a world where production code deployments are highly automated weekly, daily, or even multiple times per day activities, this kind of manual security testing can no longer be a gate. Traditional software development lifecycle security interactions are shown in Figure 2.

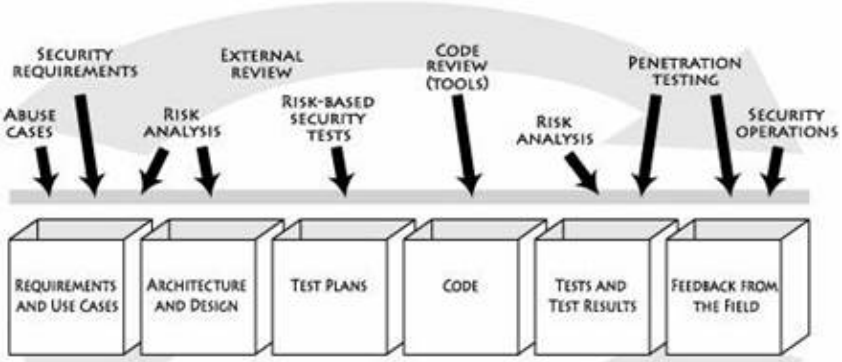

Figure 2: SDLC security interaction. From "Software Security" 2006.

Application development efforts can modernize this manual security testing control by adding bug bounties and using them to augment the penetration testing program. Bug bounties are not a replacement for penetration tests. However, because of their ongoing nature, bug bounties provide continuous feedback. Pentesters have a limited amount of time for each engagement, typically only a week or two. During that time, they try to cover the breadth of an application's entire attack surface often limiting their depth of coverage. Combining penetration tests with a bug bounty program enables them to be focused on high risk, new, or recently changed application functionality. Penetration testers have the benefit of bug bounty findings as a starting point for where to take a closer look. The bug bounty testers maintain a breadth of coverage allowing the penetration testing to be more valuable by directing the focus of the penetration testers.

Zane Lackey, Founder and Chief Security Officer at Signal Sciences, argues that there is a depth of coverage and frequency continuum. Dynamic application security

testing (DAST) scanning is automated so it can frequently be done but is unable to cover the application deeply. Penetration testing is manual and can be performed by a team with white box knowledge of the application. It can have perfect depth of coverage but is resource intensive and therefore limited in frequency. Bug bounties sit in the middle, covering far more of the application than DAST scanning and occurring far more frequently than penetration testing (AppSecUSA "Practical tips for web application security in the age of agile and DevOps", 2016). Lackey depicted the continuum with Figure 3.

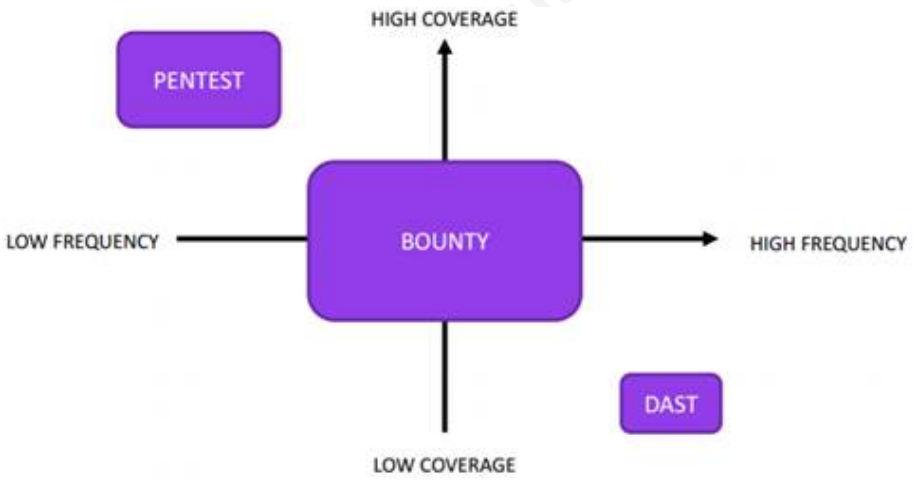

Figure 3: Coverage and frequency continuance. From "Practical tips for web application security in the age of agile and DevOps" 2016.

The combination of bug bounties and penetration testing allows for a decoupling of pentesting as a traditional gate to application deployments, while maintaining a balance of high application attack surface coverage and continuous security feedback to developers. Adding a bug bounty program helps information security by providing continuous feedback on real-world attack scenarios by real-world hackers to developers as releases to production occur.

1.5.3. Signal vs. Noise

Signal-to-noise is a useful way to describe the quality of vulnerability reports in a bug bounty program. It is the ratio of valid report submissions to the invalid ones.

- Signal. The number of valid reports. These are reproducible, non-duplicate, high-priority submissions.

- Noise. The number of invalid reports. These are non-reproducible, duplicate, out-of-scope, or too low priority submissions that are not useful.

Signal-to-noise affects the cost of the ownership of the program. The time an organization spends analyzing invalid report submissions is overhead in the program. Looking at data across all of the programs hosted by Bugcrowd in their State of Bug Bounty report, "the signal-to-noise ratio appears to follow the common 80/20 rule. Across all programs the signal value is a collective 20%. The other 80% of submissions were marked invalid or duplicate" (2015, p. 15). Corporate bug bounty programs need to be staffed appropriately to handle an expected amount of overhead.

1.5.4. Incentives: Monetary, SWAG, Recognition

The incentives associated with a bug bounty do not need to be purely financial. Other options include giving the researcher public recognition or sending them a physical item like a t-shirt. The list below discusses types of incentives.

- Monetary. Financial rewards are the default for the majority of today's bug bounty programs.
- SWAG. A meaningful physical item can be a source of pride for a security researcher. It can be a physical manifestation of bragging rights. It also serves as marketing for an organization's security program. A security researcher wearing a company's bug bounty t-shirt to a security conference can both show that researcher is a successful hacker and create additional interest in the program.
- Recognition. Researchers often want recognition for their efforts. A list on a website that shows which researchers have provided valid vulnerability reports is referred to as a wall-of-fame. Before financial rewards were popular, a wall-of-fame was the primary mechanism for vendors to reward researchers who responsibly disclosed security flaws.

Incentive options do not have to be either-or; programs often have a combination. For example, most both give a financial award and have a wall-of-fame. A prepaid debit card with a custom logo can serve as both a monetary reward and SWAG. Other

programs, most notably Facebook, have had success with such an approach (Krebs, 2011).

Figure 4: Facebook's bug bounty debit card. From "Bugs Money" by Krebs, B. 2011.

1.5.5. Software-as-a-Service Bug Bounty Platforms

When launching a vulnerability disclosure or bug bounty effort, an organization may build a mechanism for receiving and tracking vulnerability reports. This can be as simple as publishing an email address, secure@company.com, to a "Contact Us" page on a website and waiting for the submissions to arrive. Another option is to use a company that specializes in and hosts a bug bounty platform. A bug bounty platform is a specialized application with web forms tailored for collecting and tracking vulnerability information securely. These platforms have the added benefit of already having thousands of interested security researchers ready to start assessing new applications. SaaS Bug Bounty platforms include the following:

- Bounty Factory - https://bountyfactory.io
- Bugcrowd - https://bugcrowd.com
- Hackerone - https://hackerone.com
- Synack - https://synack.com

1.5.6. Researcher Reputation Score

Most of the bug bounty platforms mentioned above have the concept of researcher reputation scoring. When a researcher submits a valid bug, their score increases. If they submit an out of scope bug, a report is closed as not having enough information, or they

violate the policy, their reputation score goes down. Researchers who submit valid vulnerability reports, tend to submit higher criticality findings, and follow the rules, generally have higher reputation scores. When launching a private program, an organization can utilize the reputation scores by only inviting researchers who meet the organization's required criteria.

1.5.7. Private vs. Public Programs

Bug bounty programs can be run as public or private engagements. Historically, the only way to conduct a bug program was by publicly announcing it. With the rise of bug bounty platforms which include pools of security researchers, it is now possible to secretly invite a small subset of those researchers to participate in a private program.

- Public. A public bug bounty program is announced publicly, generally published to the public Internet and is available for anyone to participate.
- Private. Private bug bounty programs are by invitation. They are open only to select group of previously known researchers.

A public program gets more interest, and therefore, more vulnerability report submissions. They also come with the benefit of marketing an organization's information security program, indicating to customers and partners that security is a priority. However, the signal-to-noise ratio is lower than average, making it more time intensive and costly.

Invitation only, or private bug bounty programs, have become an increasingly popular option. An organization does not get the benefit of marketing their information security program, but can invite only the researchers that are known to follow policy and have the particular skill set the organization requires. Private programs have a higher than average signal-to-noise ratio. Only inviting researchers who are known to follow the rules decreases some of the risk factors of doing a bug bounty program. According to Bugcrowd, "organizations looking to reap the benefits of a traditional public bug bounty program are utilizing private, on-demand and ongoing, bounty programs more and more. 63% of all programs launched have been private" (2016, p. 5). Starting with a private program at launch has become a popular choice of larger, risk adverse organizations.

1.5.8. Time-bound vs. Ongoing

Most bug bounty programs are ongoing. That is, once an application is in scope for a public bug bounty program, it will remain in scope for rewarded vulnerability reporting for years. Generally, this scope will increase over time. As an example, Google launched their bug bounty program in 2010; www.google.com has been in an active bug bounty program for seven years. Google's scope increases as they make acquisitions, putting the new property in scope six months after the acquisition to give them time to perform reviews and remediation internally (Google, 2017).

- Ongoing. Ongoing bug bounty programs are long-term, continuous reward programs that incentivize researchers for vulnerability reports.
- Time Bound. Time-bound bounties are short-term programs. They are generally private, invitation-only programs that last two to four weeks.

Time-bound programs are effective options for limited budget or short-term needs. For example, if there is a new release with new features in an application that needs to be tested, an invitation-only, time-bound bug bounty can get set up quickly. Combine that with an incentive structure that awards the specific functionality on which the researchers should focus and that new functionally will get a quick security assessment. Time-bound bounties are also a good way for an organization to try the bug bounty concept.

1.5.9. Bug Bounty Program Types

Combining of the concepts above, Table 3 lists the common types of bug bounty programs.

Program	Visibility	Incentive	Scope
Vulnerability Disclosure Program: An organization has a mechanism for receiving vulnerability reports from third parties.	Public	Recognition / Wall-of-Fame	Broad. The purpose is to accept any vulnerability a researcher discovers.

Public Bug Bounty Program: An organization is incentivizing researchers to submit vulnerability reports.	Public	Usually monetary, but can be swag or something else of value.	Has a well-defined scope. Mature programs are fairly broad, but there is a clear delineation between what should and should not be tested.
Private Bug Bounty Program: An organization invites specific researchers to participate in an incentivized bounty.	Private	Monetary	Small scope.
Time Bound Bug Bounty: An organization wants to crowdsource testing for a single application on a limited budget.	Usually private, but can be public	Monetary	Usually a single application.

Table 3: Bug bounty program types.

2. Bug Bounty Enterprise Implementation

This paper reflects a real-life enterprise bug bounty implementation at a fortune 500, global financial company. It outlines the project phases to stand up the program, and the operational processes that are in use today.

2.1. Project Schedule

The Bug Bounty Program implementation took place as a project with two distinct phases. Phase one was planning focused on process development and selecting a vendor that provides a bug bounty platform. Phase two focused on implementation, conducting a pilot to test the vendor and our processes, and then gradually increasing application scope and public exposure leading up to the launch of a public, bug bounty program. Figure 5 shows the two phases in a Gantt chart.

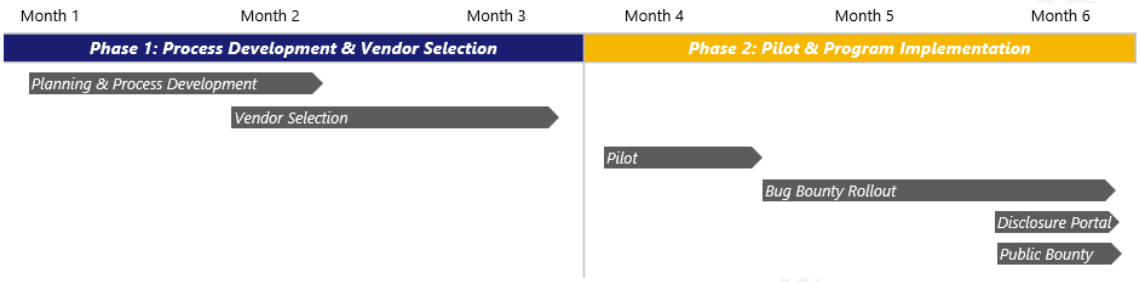

Figure 5: Project Gantt chart.

2.1.1. Phase 1: Planning, Process Development, and Vendor Selection

The first phase focused on laying the groundwork: developing processes and selecting the platform that best fit the requirements.

1. Develop Policy and Processes. Work with stakeholders to develop a vulnerability disclosure policy, processes for handling vulnerability reports, and a communication plan.
2. Bug Bounty Platform Vendor Selection. Gather bug bounty platform requirements, have vendors conduct demonstrations of their platforms, and analyze how each vendor meets the requirements. After receiving quotes from vendors, select the vendor who meets the requirements for the most reasonable cost.

2.1.2. Phase 2: Pilot and Program Implementation

The second phase focused on program implementation: first conducting a pilot to ensure the vendor and processes worked, then gradually increasing application scope leading up to the launch of a public, bug bounty program.

1. Pilot. Select an application, and work with application owners to conduct a pilot with the selected bug bounty platform. A time-bound, private bug bounty ran for two weeks.
2. Private Program Launch and Application Enrollment: After minor tweaks to processes based on lessons learned from the pilot, work with application owners and enroll applications into a private bug bounty program.

3. Launch Vulnerability Disclosure Portal. While a bug bounty program incentivizes researchers to assess in scope applications, the bug bounty platform is also used as a general vulnerability disclosure front door. In the event that someone wants to responsibly disclose a vulnerability in an application outside of that scope, the policy allows for submission of vulnerabilities that impact any application, while making it clear only the applications in scope for bug bounty are eligible for rewards.
4. Launch Public Bug Bounty Program. For the launch of the public program, all of the applications that were working well in the private program were put in scope for a public program. This involves publically publishing the vulnerability disclosure policy, tuned over time with lessons learned from the pilot and private program, along with the well-defined scope of applications, to the bug bounty platform.

2.2. Vendor Selection

There are a number of vendors with vulnerability disclosure, bug bounty, or crowdsourced pentesting platforms. Three vendors were compared: two focused on vulnerability disclosure and bug bounty, and one with a private crowdsourced pentest focus. They were scored against eight major requirements with 41 sub-requirements in a weighted scoring rubric, as seen below in Table 4.

Requirements	Weight	Bugcrowd Score	Bugcrowd Weighted Score	HackerOne Score	HackerOne Weighted Score	Synack Score	Synack Weighted Score
Provide vulnerability disclosure and bug bounty platform capable of running private, public, and one-off engagements.	100						
Researchers can view and must agree to Vulnerability Disclosure Policy prior to participating.	100						
Reward researcher upon verification, remediation, and approval of vulnerability report.	100						
Vendor provides vulnerability disclosure and bug bounty managed services.	100						
API access to data for integrating, importation, and correlation with other vulnerability discovery sources.	70						
Alerting, notification, reporting, metrics, key performance indicators, and analytics of vulnerability data.	70						
Platform Functionality	50						
Other: Legal, PR, Etc…	50						

Table 4: Vendor rubric.

Since the requirements for each company and program will be different, the details of the requirement and how each vendor scored are not included. The differences between the vendors are summarized by Katie Moussouris, founder and CEO of Luta Security, in the following presentation slide.

BugCrowd	HackerOne	Synack
The Easy Button for Triage	Platform for Power Users	Secret Squirrels
Great if you need triage support, less so if you don't want triage outside your company's eyes only	Great if you want automation for your own vulnerability handling, less so if you lack the internal talent to use it	Great if you want a crowd-sourced penetration test under NDA, less so if you need a broader pool of eyes

Table 5: Bug bounty service providers at a glance. From "Lessons learned from running big bug bounty programs" by Moussouris K. 2016.

2.3. Application Enrollment

Each application has a three-step enrollment process into the bug bounty program. After ensuring that the application has gone through the required traditional security assessments such as source code reviews and penetration testing, an application can be put in scope of a private bug bounty. Then, if the application does not show any adverse effect to the assessment traffic, it will be added to the scope of the public bug bounty. This process is described in Figure 6.

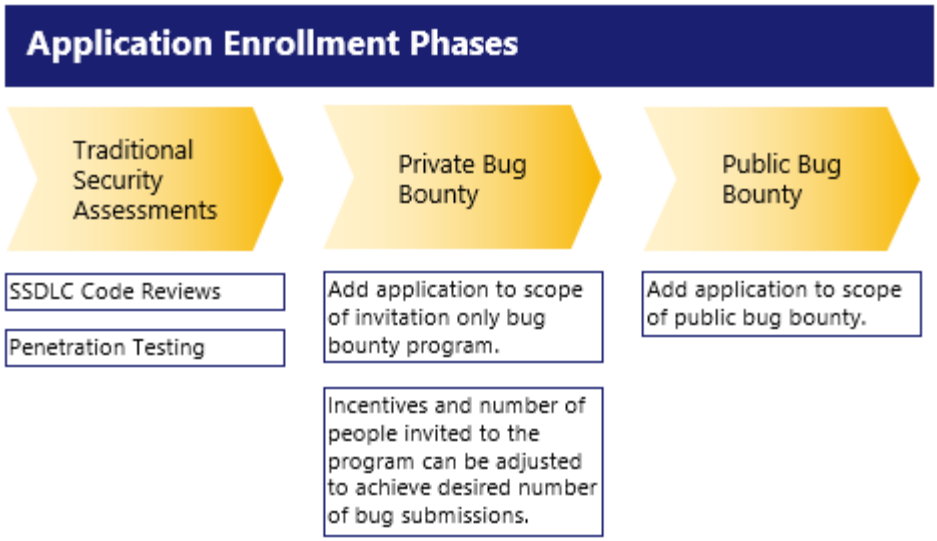

Figure 6: Application enrollment steps.

- Step 1: Traditional security assessments. Review the results of code reviews and penetration testing to ensure the application has been internally tested and does not have any open critical or high findings. If there are any findings that have an acceptable business risk, add those as exceptions in the bug bounty policy as to not incentivize researchers to look for them and to make sure no one expects a reward for finding them.
- Step 2: Private Bug Bounty. A private, invitation-only bounty will be stood up for the application. Invite only a select number of researchers with a high platform reputation score. Since these folks are more likely to follow the rules of our policy and we can adjust the number of invited researchers, the risks associated with a bug bounty program are reduced. The signal-to-noise ratio and overall report quality are also better using researchers with a high reputation score. While in the private bounty, any low hanging fruit should be discovered and addressed. The private bounty also helps the information security team better gauge the responsiveness of the application owner and developers.
- Step 3: Public Bug Bounty. If all goes well in a private bug bounty, the application will be added to the scope of the public bug bounty. This may result in a higher number of vulnerability reports, so both the information security and application teams need to be prepared to address them. Over a short period, that spike in reporting should normalize.

2.4. Vulnerability Disclosure & Bug Bounty Policy

A Vulnerability Disclosure or Bug Bounty Policy is what a company posts, usually publicly on the Internet, to provide researchers the rules of engagement for the program. It sets the expectations for how both parties will behave throughout the process. The purpose of a bug bounty is to incentivize researchers to look for and report what we are interested in finding; so point researchers in the right direction by outlining the policy, scope, rules, areas of focus, and exclusions.

For pertinent examples, see:

- https://hackerone.com/uber
- https://hackerone.com/twitter

2.4.1. Program Scope

The bug bounty policy needs to tell researchers what to test. "The single most important thing that you can do to ensure a successful program is to define a clear scope, leaving nothing open to interpretation. A bounty's scope informs the researchers what they can and cannot test, and points them to key targets" (Bugcrowd, 2016). It is also essential to explicitly call out what is not in scope. The most common example listed as out of scope are hosts that resolve to third-party services. Bug bounty platform vendors using a hacker reputation penalize researchers for making out-of-scope submissions, so it is vital to set correct expectations of what is not rewardable before testing begins. This helps prevent frustrating researchers by making sure they do not expect to get a reward for a vulnerability discovered on an otherwise in scope target.

Let the researchers know what is important for them to test. Focus areas may include specific bug types, particular functionality, new features, or whatever needs special attention. For complex or unintuitive targets or functionality like APIs, consider providing documentation to explain how the application works so researchers can focus on testing rather than figuring out the application.

Guide researchers in the right direction by accurately articulating any exclusions. Good examples of exclusions are vulnerabilities that can easily be found with automated scanners or DoS attacks that could cause impact to the application.

Targets

https://www.example.com

https://developer.example.com

https://api.example.com

Out of scope

https://blog.example.com

Jason Pubal jpubal@mastersprogram.sans.edu

Focus

The company is launching software development kits this month that enables third-party developers to deploy faster merchant and peer-to-peer payments in the mobile apps they create. Developers can leverage prepackaged code that can read QR and NFC tags, extracting important merchant and user details in the payment process. **Valid submissions in the SDK or API will be awarded double.**

Exclusions

Do not conduct social engineering attacks against company employees, partners, or customers.

Do not test the physical security of company offices.

Do not perform any testing that could harm our services such as denial of service (DoS) attacks.

Figure 7: Bug bounty program page example.

2.5. Bug Bounty Workflow

Once an application has been added to the scope of the bug bounty program, researchers will start testing the application and submitting vulnerability reports. This workflow is how to triage the finding, communicate it to the application team, reward the researcher, and remediate the vulnerability. This process is visually represented in Figure 8.

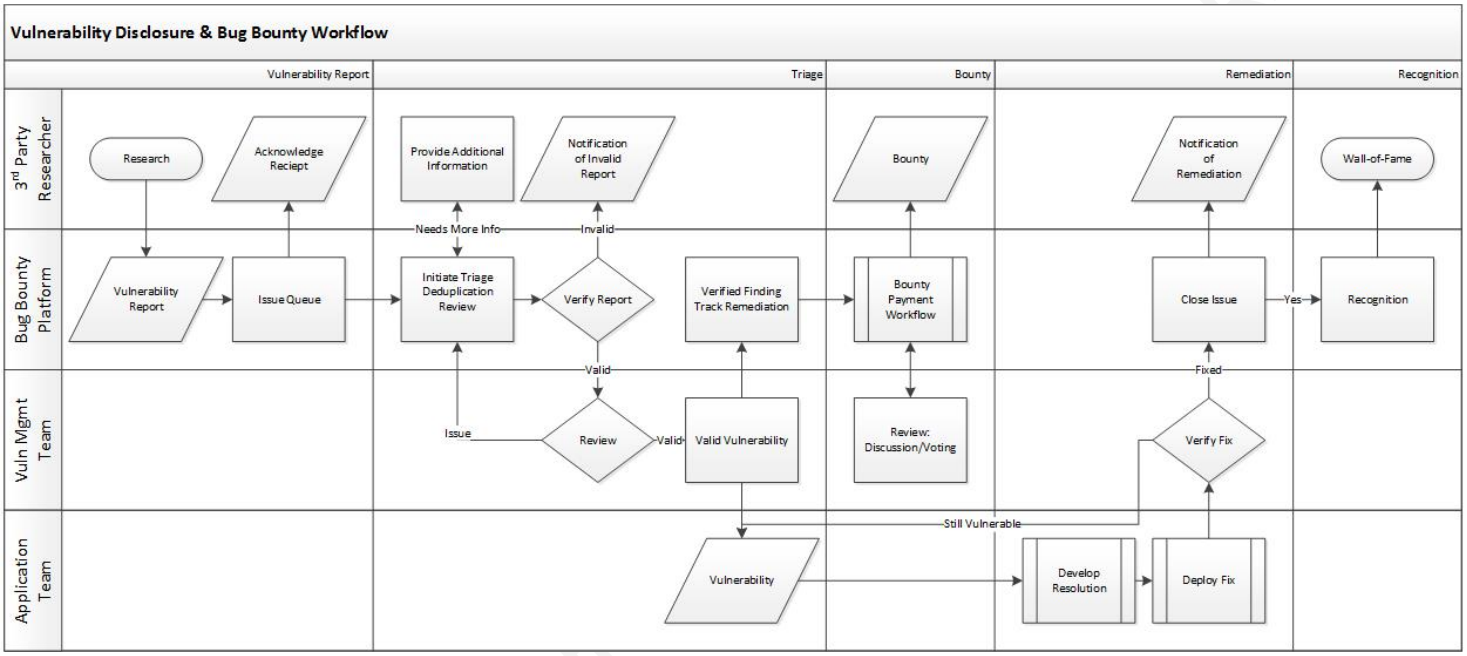

Figure 8: Bug bounty workflow.

2.5.1. Vulnerability Report

A third-party researcher will interact with the bug bounty platform to submit a vulnerability report. That report would go into the queue, and the platform will send the researcher an acknowledgment. Vulnerability reports sitting in the issue queue await triage.

2.5.2. Triage

The information security vulnerability management team works on the triage queue analyzing each report, assessing the application with the potential issue, and determining if the vulnerability is valid. If the vulnerability is either invalid, a duplicate, or previously known through another vulnerability detection process, the issue is closed in the queue. That closure could then kick off a notification to the researcher letting them know the finding is invalid. If the report was not detailed enough and needs additional information from the researcher, the platform will send them a notification. If the vulnerability report is determined to be a valid vulnerability, it would then get reported to the appropriate application team for remediation.

Jason Pubal jpubal@mastersprogram.sans.edu

2.5.3. Bounty

When it is determined that the researcher has submitted a valid vulnerability, they can be paid the appropriate amount for the criticality of the vulnerability through the bug bounty platform. Bounty payments typically happen either after the vulnerability is triaged and confirmed, or after it is fixed. More specifically, "About one out of every five will pay when the vulnerability is validated (18%), and half will pay when a vulnerability is resolved (48%), and the remainder pay on a case-by-case basis (34%)" (HackerOne, 2017, p. 12). In this workflow, the researcher is paid after the vulnerability is triaged and confirmed as valid. Paying researchers earlier in the workflow incentivizes them to continue looking for and submitting vulnerabilities.

2.5.4. Remediation

The internal remediation process will be similar to the remediation process for any other application vulnerability. Due to the nature of an externally reported security issue, the risk and prioritization might be higher than similar vulnerability types for similar applications. The appropriate application or support team will develop and implement a fix for the vulnerability. After the fix is deployed, the vulnerability management team will retest the application to ensure it was remediated. If the vulnerability still exists, the vulnerability management team will continue to work with the application team. If it is remediated, the issue can be closed within the bug bounty platform, triggering notification to the third-party researcher and the recognition step.

2.5.5. Acknowledgement

Companies can give the researcher public recognition that they successfully submitted a valid security vulnerability that was acknowledged and fixed. The bug bounty platform should first ensure that the third-party researcher accepts being publicly credited, as they may wish to remain anonymous.

2.6. Vulnerability Rewards

The monetary incentive paid to researchers will depend on the severity of the vulnerability, the inherent risk of the application, and the maturity of the bug bounty program. The higher the severity of the vulnerability, the higher the payout will be within

the severity tiers. For example, initially a program could pay out $100, $300, $900, and $1,500 for low, medium, high, and critical bugs respectively. Over time, the applications will be more secure and vulnerabilities harder to find. As time passes, bug submissions tend to drop and increasing the rewards keeps researchers' interest high. If researchers must spend more time discovering vulnerabilities against a hardened application, the incentives have to be adjusted. For example, at the end of the first year of a program, one could pay out at a higher rate of $300, $900, $2,700, and $15,000 for low, medium, high, and critical bugs respectively. For a comparison with other programs, HackerOne states that "through May 2017, the average bounty for a critical issue paid to hackers on the HackerOne Platform was $1,923. For all vulnerabilities reported of any severity, the average bounty payout was $467" (2017, p. 15). See the chart below (Table 6) for examples of vulnerability types and bounty payouts.

Severity	Impact	Example Vulnerability Types	Security Program & Vulnerability Disclosure Maturity		
			Initial $100 - $1,500	Intermediate $200 - $5,000	Advanced $300 - $15,000
Critical	Vulnerabilities that cause a privilege escalation from unprivileged to admin or allow for remote execution, financial theft, etc.	• Remote Code Execution • Vertical Authentication Bypass • SQL Injection with significant impact • XML External Entities Injection with significant impact	$1,500	$5,000	$15,000
High	Vulnerabilities that affect the security of the platform	• Lateral authentication bypass • Stored XSS with significant impact • CSRF with significant impact • Direct object reference with significant impact • Internal SSRF	$900	$1,800	$2,700
Medium	Vulnerabilities that affect multiple users and require little or no user interaction to trigger	• Reflective XSS with impact • Direct object reference • URL redirect • CSRF with impact	$300	$600	$900
Low	Vulnerabilities that affect singular users and require interaction or significant prerequisites to trigger (MitM)	• SSL misconfiguration with little impact • SPF configuration problems • XSS with limited impact • CSRF with limited impact	$100	$200	$300
Acceptable	Non-exploitable vulnerabilities in functionality. Vulnerabilities that are by design or are deemed acceptable business risk to the customer	• Debug information • Use of CAPTCHAs • Code obfuscation • Rate Limiting, etc.	$0	$0	$0

Table 6: Vulnerability rewards.

2.7. Metrics

These metrics and key performance indicators can be used to help manage a bug bounty program.

2.7.1. Security Testing Coverage

Security testing coverage tracks the percentage of applications in the organization that have been subjected to security testing. That is, out of all the web applications the organization uses, how many of them are enrolled in the bug bounty program? This is expressed as a percentage.

$$STC = \frac{\text{count (of enrolled applications)}}{\text{count (deployed applications)}} * 100$$

2.7.2. Mean-Time to Mitigate Vulnerabilities

Mean-time to mitigate vulnerabilities measures the average amount of time required to remediate an identified vulnerability. This is a measure of the organization's or development team's performance. It also captures how long, on average, the window is in which a vulnerability can be exploited. This is expressed in a number of days.

$$MTTMV = \frac{\sum \quad (\text{Date of Mitigation} - \text{Date of Detection})}{\text{Count (Mitigated Vulnerabilities)}}$$

2.7.3. Signal and Noise

Signal and noise are measures of vulnerability report quality. Signal is the percent of valid reports received. Noise is the percent of invalid reports received. Both are expressed as a percentage.

$$Signal = \frac{\text{Number of Valid Reports}}{\text{Total Number of Reports}} * 100$$

Jason Pubal jpubal@mastersprogram.sans.edu

$$Noise = \frac{\text{Number of Invalid Reports}}{\text{Total Number of Reports}} * 100$$

Signal and noise are generally presented as a ratio. The signal-to-noise ratio is presented as the reduced ratio of Signal over Noise.

$$Singal - to - Noise\ Ratio = \frac{\text{Signal}}{\text{Noise}}$$

2.7.4. Vulnerability Reports Received

The number of vulnerability reports received is an indication of researcher interest in the program. Report submissions can be trended over time as shown in Figure 10. This will show spikes when a new program is launched, the scope of an existing program is changed, or research of a new attack goes public. When interest in the program is low for an extended period, this could indicate an opportunity to raise the reward incentives.

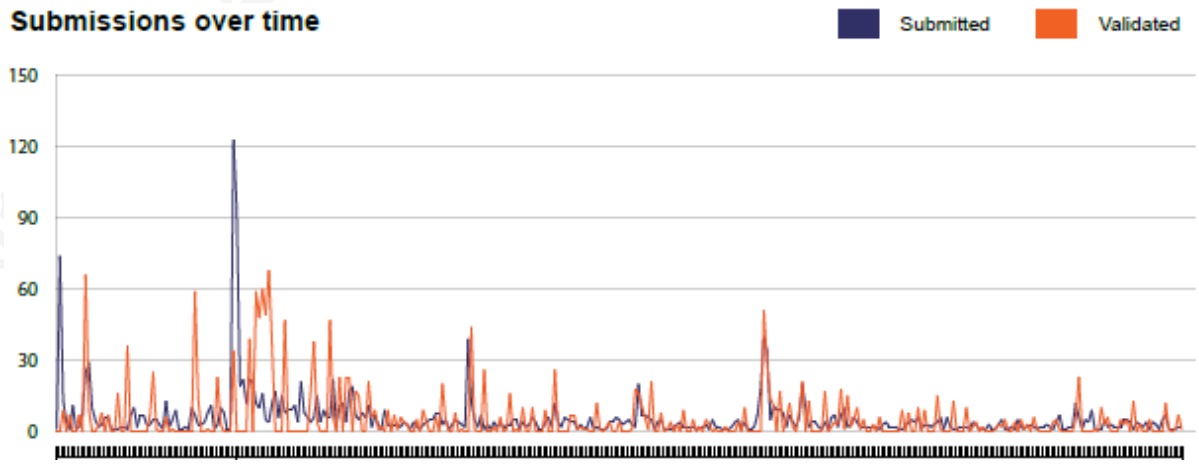

Figure 9: Vulnerability report counts over time. From "The State of Bug Bounty" 2016.

2.7.5. Top 10 Vulnerabilities

OWASP maintains a list of top ten vulnerabilities to raise awareness about application security. While this could be a starting point for application security training, it would be more helpful to know what an organization's specific pain points are. With this, the security team can create custom training modules based on what the organization's development teams have trouble coding. This can be visually represented

with a pie chart to easily see how much of an issue the vulnerability is compared to others, as seen in Figure 10.

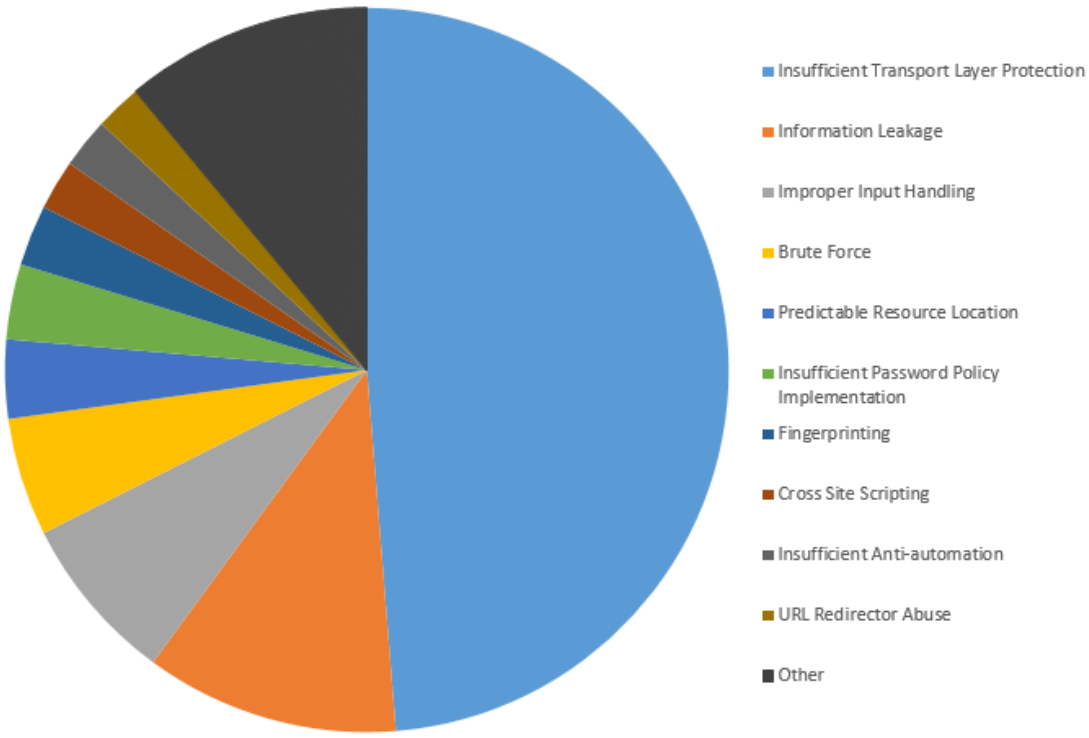

Figure 10: Top 10 vulnerabilities.

2.7.6. Vulnerabilities per Application

Vulnerabilities per application is a count of the number of vulnerabilities found in a specific application. This is expressed as a cardinal number. This can be visually represented with a bar chart to easily see how different applications compare against each other. Below is a Top 10 Vulnerable Applications chart.

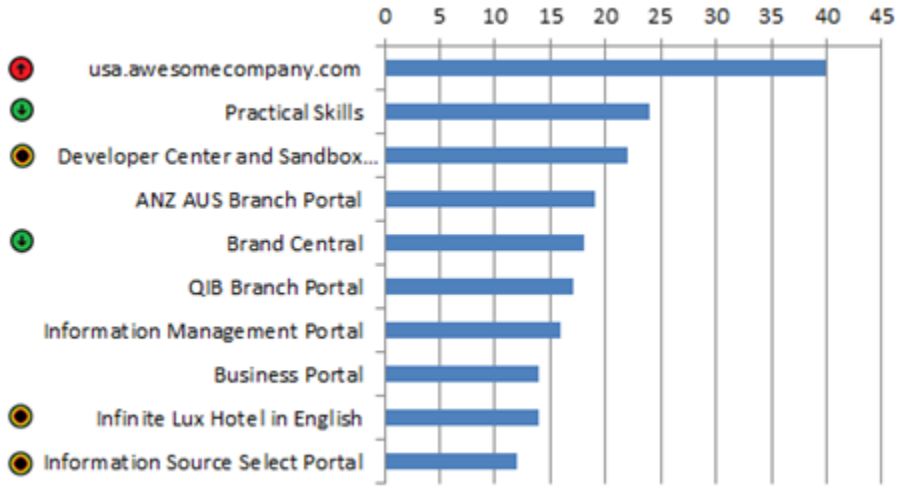

Figure 11: Top 10 vulnerable applications.

Having a top ten vulnerable applications list helps to focus remediation efforts. It points security to the development teams responsible for these applications targeting them with the training developed from the top ten vulnerabilities list.

Keeping track of changes in the top 10 is also useful. If an application drops in rank, the application team is likely doing a good job fixing issues. If an application goes up on the list, reprioritize and act accordingly. If a new application shows up on the list, it may be an application that is being tested for the first time and presents a high risk to the organization. Report the vulnerabilities to the appropriate team and start working with them on remediation.

2.7.7. Defect Counts per Risk Rating

Another important metric is a defect count by risk rating. This can be represented as a point in time with a bar chart and trended over time with a line graph as in Figure 12. One could Look at the trending of these defect counts along with the total number of applications being assessed to see how the trend is related to the amount of applications enrolled in the program. With this, it is easy to see how much risk exists in the environment and to get a sense for the effectiveness of the program.

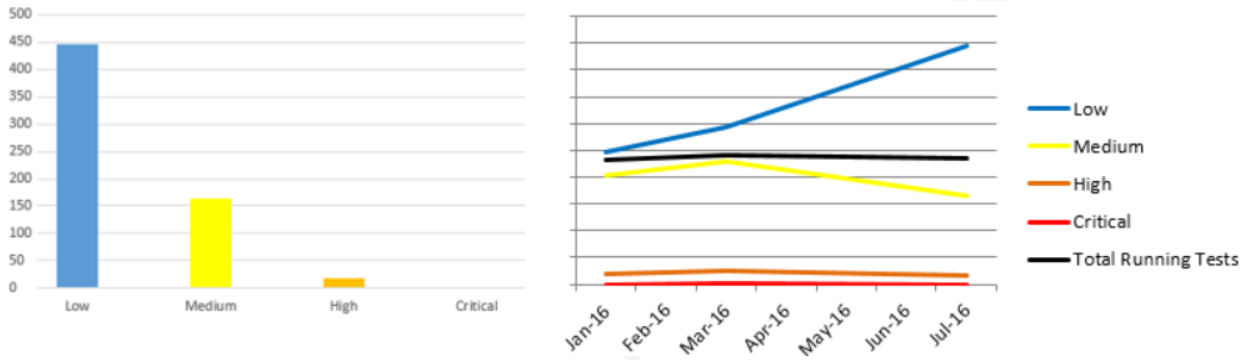

Figure 12: Vulnerability Risk Ratings

2.7.8. Vulnerability Coordination Maturity Model

The Vulnerability Coordination Maturity Model, created by Moussouris, rates five capability areas as basic, advanced, or expert (2015). These capabilities are Organizational, Engineering, Communications, Analytics, and Incentives, as further detailed below.

- Organizational: At a basic level, an organization has decided to receive and respond to vulnerability reports and has executive support. At an advanced level, there is a policy and process that follow some framework or best practices. At an expert level, there is dedicated budget and personnel.
- Engineering: At a basic level, engineering has a way to receive vulnerability reports and a way to track them internally. At an advanced level, there is a dedicated bug tracking software, such as JIRA, along with documented risk-based decisions around exceptions to fixing vulnerabilities. At an expert level, there is root cause analysis and a feedback loop to the development lifecycle to eliminate classes of vulnerabilities.
- Communications: At a basic level, there is an ability to receive vulnerability reports, along with a way to communicate those vulnerabilities to stakeholders. At an advanced level, there are repeatable and custom communications for a broad range of stakeholders like

customers and media. At an expert level, that information can be shared in a structured and automated fashion.

- Analytics: At a basic level, the number of severity of findings are tracked over time. At an advanced level, that tracking leads to root cause analysis with a feedback loop to development. At an expert level, an organization also uses data about threats to prioritize remediation and detect attacks against known vulnerabilities.
- Incentives: At a basic level, a vulnerability disclosure policy states that no legal action will be taken for submitting vulnerability reports and that recognition can be given to researchers. At an advanced level, financial incentives are given for reporting vulnerabilities. At an expert level, a company has knowledge about vulnerability markets where applicable bugs are being traded and tailors financial rewards to disrupt them.

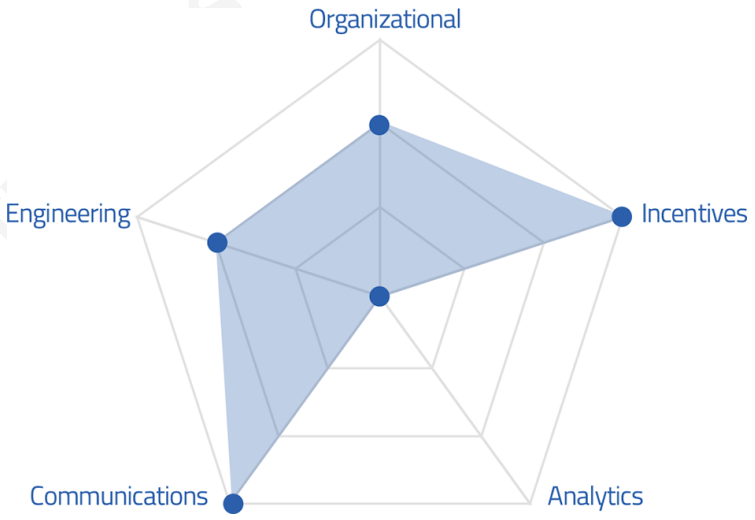

Figure 13: Sample spider graph. From "Vulnerability Coordination Maturity Model." By Moussouris K. 2015.

It is recommended to rate a company along the maturity model, have an idea of what maturity level the organization wants to get achieve, and conduct a gap analysis of what it will take to get there.

2.8. Additional Stakeholder Considerations: Legal & PR

Bug bounty programs are public facing. Two important but often forgotten stakeholders are legal and public relations. Bug bounty programs can be the first time

information security or engineering personnel are dealing directly with external entities. Note that everything shared through the program is done so on behalf of the company.

2.8.1. Legal

Companies that implement a bug bounty program should ensure the involvement of a legal team. Legal concerns that tend to come up are liability if a bug bounty researcher exploits an application coming into contact with sensitive information such as customer's personally identifying information (PII), a researcher publically disclosing a vulnerability before it is fixed, Know Your Customer (KYC) anti-money laundering laws, and researchers who might be in locations where the US has active sanctions.

According to Denaro, a lawyer in DC, "the legal exposure an organization has from using the crowd is basically the same as using any other means of pentesting that you would traditionally buy" (2016). He goes on to say the bug bounty policy is a legal contract with researchers and can provide the same protection as a contract in place with a more traditional penetration tester. Using a bug bounty platform vendor to manage and pay researchers covers some of the other concerns involving paying researchers bounties, effectively transferring the risk to them. Include the legal team, and have them review the bug bounty policy along with any bug bounty platform vendor's terms and conditions.

2.8.2. Public Relations

Another important stakeholder that is often ignored is the public relations (PR) or marketing teams. Doing a Google News web search for "bug bounty" will show articles about various companies launching their own bug bounty programs. One of the benefits of a public bug bounty program is the positive marketing of a company's security program, which can be amplified with a coordinated PR statement. Since bug bounty programs are externally facing, it is recommended to include the PR and corporate communications teams to ensure they have input on the verbiage and branding being used. Even if a company is running a private bug bounty program, with the proliferation of information security news outlets and blogs, chances are some news will go public about the program. It is a good idea to have a statement ready for when that happens.

3. Conclusion

Bug bounty programs are increasingly being embraced from startups to financial institutions and even high-security environments like the US Pentagon (Larson, 2017). These crowdsourced security assessments have a lot of benefits and are worth considering, especially in applications with quick development and release cycles or companies without dedicated application security personnel on staff.

I started a bug bounty program at a fortune 500 global financial company. This paper outlines the research used to justify and prepare for the program, the project steps to compare vendors and get things started, along with workflow and processes currently being utilized. Take the lessons learned from this project to stand up a time-bound, private bug bounty of your own and see if this type of security assessment is a fit for your organization.

As a parting thought, if you do embark on this journey remember to have empathy and patience. The people testing your applications and submitting bugs will have a range of security skills, English literacy, education, and professionalism. Each report represents an amount of time that someone spent testing your application, documenting the results, and submitting them to you with no guarantee of receiving anything in return. The more you foster the relationship with the community, the more value you will get out of the program.

Jason Pubal jpubal@mastersprogram.sans.edu

4. References

Bacchus, A. (2017) Bug Bounty Field Manual. Retrieved September 29, 2017, from https://www.hackerone.com/resources/bug-bounty-field-manual

Bugcrowd. (2016) Anatomy of a bounty brief. Retrieved October 5, 2016, from https://pages.bugcrowd.com/hubfs/PDFs/Anatomy-Bounty-Brief.pdf

Bugcrowd. (2015) The State of Bug Bounty. Retrieved October 5, 2016, from https://pages.bugcrowd.com/state-of-bug-bounty-download.html

Bugcrowd. (2016) The State of Bug Bounty. Retrieved October 5, 2016, from https://pages.bugcrowd.com/hubfs/PDFs/state-of-bug-bounty-2016.pdf

Bugcrowd. (2017) The State of Bug Bounty. Retrieved September 29, 2017, from https://www.bugcrowd.com/resource/2017-state-of-bug-bounty/

Chambers, J., & Thompson, J. (2004). *Vulnerability Disclosure Framework*. National Infrastructure Advisory Council. Retrieved October 6, 2016, from https://www.dhs.gov/xlibrary/assets/vdwgreport.pdf

Denaro, J. (2016). Bug hunting and the law. Retrieved October 5, 2016, from https://pages.bugcrowd.com/hubfs/PDFs/state-of-bug-bounty-2016.pdf

Foreman, P. (2010). *Vulnerability management*. Boca Raton: CRC Press.

Google. *Google Vulnerability Reward Program (VRP) Rules.* Retrieved September 16, 2017, from https://www.google.com/about/appsecurity/reward-program/index.html

Jason Pubal jpubal@mastersprogram.sans.edu

HackerOne. (2017). The hacker-powered security report. Retrieved September 29, 2017, from https://www.hackerone.com/sites/default/files/2017-06/The%20Hacker-Powered%20Security%20Report.pdf.

Harris, S. (2012). *CISSP all-in-one exam guide*. Berkeley, Calif: Osborne

ISO/IEC. (2014). *29147:2014 Information technology – Security Techniques – Vulnerability Disclosure*. Retrieved October 6, 2016, from http://standards.iso.org/ittf/PubliclyAvailableStandards/c045170_ISO_IEC_29147_2014.zip

Krebs, B. (2011). *Bugs Money.* Krebs on Security. Retrieved October 10, 2016, from http://krebsonsecurity.com/2011/12/bugs-money/

Larson, S. (2017). *Why the Pentagon wants people to hack it.* CNN. Retrieved September 24, 2017, from http://money.cnn.com/2017/04/11/technology/hack-the-pentagon-synack-bug-bounty/index.html

Nicolett, M. (2005, March). *How to develop an effective vulnerability management process*. Retrieved November 23, 2016, from http://www85.homepage.villanova.edu/timothy.ay/DIT2160/IdMgt/how_to_develop_.pdf

Payment Card Industry Security Standards Council. (2016, April). *Payment Card Industry (PCI) Data Security Standard.* Retrieved September 16, 2017, from https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf

McGraw, G. (2006) *Software Security.* Upper Saddle River, N.J. Addison-Wesley.

McGraw, G., Migues, S., & West, J. (2015). B*uilding security in maturity model*. Retrieved October 10, 2016, from http://bsimm.com/download/dl.php

Moussouris, K. (2016). *Lessons learned from running big bug bounty programs*. Retrieved May 23, 2017, from https://www.oreilly.com/ideas/lessons-learned-from-running-big-bounty-programs

Moussouris, K. (2015). *A maturity model for vulnerability coordination*. Retrieved September 29, 2017, from https://www.hackerone.com/blog/vulnerability-coordination-maturity-model

Jason Pubal jpubal@mastersprogram.sans.edu