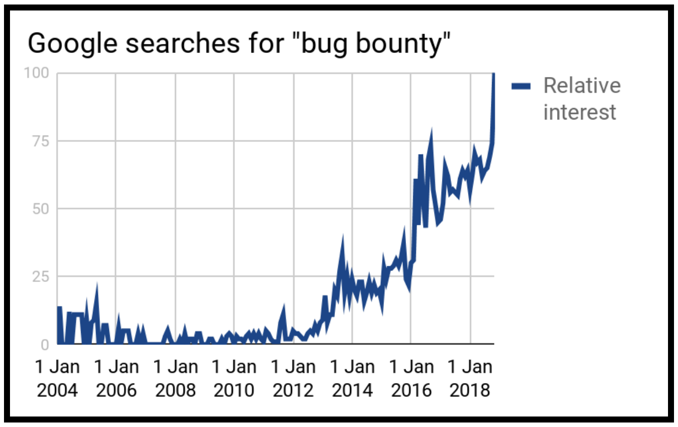
What does it take to run a bug bounty program?

Typical problems and practical solutions



1. Introduction

During the last few years, bug bounty programs have become an increasingly well-known method of vulnerability testing[1]. Companies who have adopted the technique include Google, Reddit, Facebook, Microsoft, Apple, Valve, Fitbit, Mastercard, Netgear, Avast, DigitalOcean, Android, Atlassian, and many others[2].

But what *is* a bug bounty program? It's a formal program where:

1. An organization announces that they'll pay researchers for information about security bugs in their systems;
2. Security researchers find bugs and report them to the organization; and
3. The organization rewards the researchers for their efforts.

So the researchers are "bounty hunters", gunning for security bugs and bringing them back to the organization for a profit.

A few initial observations can already be made:

- Just as the town sheriff has to put up a *WANTED* poster to catch a criminal, the organization must **advertise** their bounty to their target audience (the researchers).
- Organizations only pay researchers for **results**, not effort. This is cost-efficient for the organization, but imposes a freelancer's income schedule on the researchers.

[1] See the Google Trends graph.

[2] See the bounty pages of Google; Reddit; Facebook; Microsoft; Apple; Valve; Fitbit; Mastercard; Netgear; Avast; DigitalOcean; Android; and Atlassian.

- Organizations are competing for the attention of the researchers, who may choose to do other work instead.

These points will be revisited in Section 3.

2. Benefits of a bug bounty

The increasing popularity of such programs can be attributed to their numerous benefits:

2.1 People will attack your software anyway

The world is full of curious people who like breaking things, especially over the internet when it's harder for local law enforcement to catch them.

By running a bounty program, you're adding an economic incentive for this to happen on your own terms. Criminals wanting to find and exploit bugs to harm your organization now have to race against the clock, because **they're competing with the white-hat hackers participating in your bug bounty program**. This increases the chance that you'll be able to fix those bugs before they're ever exploited.

2.2 You can tap into international, specialist talent

If you hire a penetration tester to come into your office and look at your products, you can gain the perspective of one person. But if you run a bug bounty, you can enjoy the expertise of many security researchers around the world.

Once you make your bounty public, you're entering a **global marketplace** of researchers looking for interesting bounty programs to work on. Since researchers tend to be specialists in a particular technology, and only results are rewarded, a bounty program can attract the best specialists in the world to test your software at a relatively low cost.

This benefit can be amplified by your choice of bounty platform, as discussed in Section 3.

2.3 Bounty programs satisfy security standards

Popular security standards like the NIST Cybersecurity Framework and ISO 27001 request regular "vulnerability testing" of your system. Since a bug bounty program puts your system continually under the microscope of security researchers, it goes above and beyond requirements of an annual or twice-yearly penetration test. As a result, a bounty program can help your system meet these standards.

2.4 Ultimately, bounty programs increase the targets' security

If you respond to bounty reports by fixing the reported bugs, the low-hanging fruit of easily-found bugs will gradually disappear from your system. Attackers hoping to compromise your system are forced to spend more and more effort finding bugs, leaving you with products that are more secure and trustworthy for your customers.

3. Challenges and mitigations

Unfortunately, the many benefits of a bug bounty are accompanied by challenges. Some typical problems that may arise, and practical solutions to them, are listed below:

3.1 You need a platform to run your bounty on

Unlike Aphrodite or Athena, bug bounties don't spring into existence fully-formed. They have several requirements to even begin: having a public website where researchers can submit bug reports, storage for all related data, user management and access control to prevent sensitive information from being shared between researchers and the public, a payment system to facilitate bounty rewards, and other features. A platform that provides these administrative functions is called a **bounty platform**.

It's *possible* to write your own bounty platform from scratch. But unless you plan to sell the platform as its own product, it's not worth it. On the other hand, using an established bounty platform comes with multiple benefits:

- You don't have to delay your bounty program until you've finished writing your own platform, which may take months or years;
- You don't have to delay your bounty program until you can find staff who can handle the administrative functions, such as running the website or the payment system;
- Some bounty platforms offer **filtering services** (see Section 3.4) which can save your staff effort and frustration; and
- Bounty platforms come with an existing **researcher base** who can join your bounty program (see Section 3.7).

Some popular options for your choice of bounty platform include:

- Bugcrowd
- HackerOne
- HackenProof
- Synack
- Cobalt

Note that some companies refer to their bug bounty platform as a "vulnerability disclosure platform", especially when offering the option to reward researchers with kudos or points instead of money.

3.2 You don't have experience with bug bounties

If you've never run a bug bounty before, the first-time experience can be daunting. To slow things down and give yourself the opportunity to learn, you can deliberately limit the number of bug reports you receive from researchers:

- Don't list all of your organization's products on the bounty straight away. Start with just one.
- Don't publish the bounty publicly on the internet. Make it a private program accessible to a few select researchers by invitation only.
- Don't offer large bounty rewards at the beginning. Keep the payouts small until you've calibrated your spending.

It's also a good idea to establish a **shared channel** (such as a chatroom or forum) where all of your bounty staff can share information and ask for help, so that the knowledge of one team member can benefit everyone.

3.3 When should we increase the bounty?

Initially, small bounty rewards can be helpful for preventing an overwhelming flood of reports (as discussed in Section 3.2). But eventually the bounty rewards must be increased. Why is that?

Economics gives the answer. Since a successful bounty program gradually eliminates easily-discoverable bugs in your products, the effort that researchers have to spend finding bugs increases. Eventually, an equilibrium is reached: researchers stop submitting reports on a product, because the reward they receive for reporting a bug doesn't measure up to the effort they'll have to spend finding it. At this point, it's time to increase the bounty rewards on the product.

A **common mistake** is to identify this moment based on gut instinct. Such judgements are wrong as often as not, and become increasingly unreliable and inconsistent as your bounty program grows to cover multiple products and bug types. Instead, pull data from your bounty platform to answer questions like:

- In the last 90 days, were **critical bugs** reported in this product? (If so, slow down; researchers are still finding serious problems at the current reward levels.)
- Is the development team who fixes the bugs **able to handle an increase in reports**, or are they already flooded with work? Is their queue of issues to fix growing or shrinking?
- Are you **over or under budget** for the year's bounty rewards? Can you afford to pay out higher rewards?

All of these figures should factor into your decision. It can also help to create a dashboard or calculator which automatically collects and reports on this data.

The corollary of this point is that to pull data from the bounty platform, your criteria for choosing a platform (see Section 3.1) should include a well-documented API, or at least a summary report page that can show you what's going on in your bounty.

3.4 Most incoming bug reports are invalid

Once you make your bounty program public on the internet, receiving invalid bug reports is par for the course. For example, about four out of five bug reports submitted to Atlassian turn out to be invalid,[3] usually by being out of the scope of the bounty. Trawling through these invalid reports in search of gold is a time-consuming task that takes your staff away from the technical work of running a bug bounty.

Fortunately, this problem can be addressed by choosing a bounty platform (Section 3.1) which offers **filtering services**. This means that your organization pays the platform's employees to work on your behalf by filtering out invalid reports, thus saving your staff considerable time and frustration.

You can also take a preventative approach against invalid reports by means of your bounty's **briefing page**, i.e. the public landing page outlining the scope and rules of your bounty. Since this page is your opportunity to tell researchers which reports will be marked invalid, you should use it as your first line of defence against invalid reports. By clearly stating which products, bug types and attack vectors are included in your bounty and which are not, you can prevent invalid reports from being submitted in the first place.

3.5 Communication fatigue

At least for now, scarcity is a fact of life: there's never enough time or enough energy to go around, and the same applies to the social batteries of your bounty staff. Since running a bug bounty demands an immense volume of communication, this presents a challenge.

To prevent issues with communication fatigue, you can provide your bounty staff with a set of standard responses, letting them say what they need to say without wearing themselves down. These responses should be templated, as in these examples:

> ***Message template: Bug resolved***
> Hi <researcher>,
> Thank you for your report to our bug bounty program.
> The issue has been fixed by the development team and should reach production soon.
> If you can still reproduce the issue in 2 weeks from today, please let us know and we can investigate further.
> Thank you for your continued efforts toward our bug bounty program.
>
> ***Message template: Responding soon***
> Hi <researcher>,
> Thank you for your report to our bug bounty program. We are reviewing your report and we will get back to you as soon as possible. <Organization> aims to have reports triaged within <time period> of the reporting date.
> Thank you for your continued efforts, and we'll get back to you soon.

[3] Based on data from Financial Year 2018.

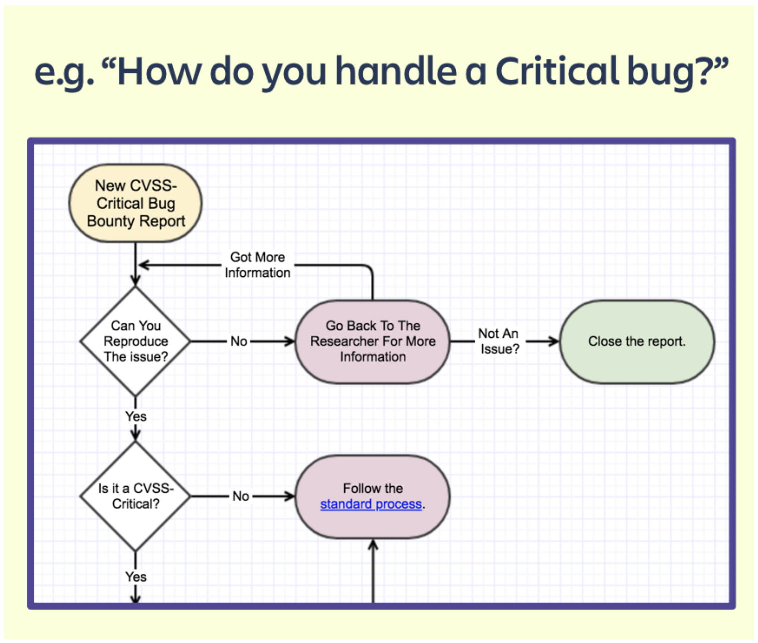
Other scenarios to consider responses for include:

- The bug has been confirmed as reproducible
- The bounty reward has been paid out
- The bug is being tracked on an issue tracker which the researcher can follow
- The bug is already a known issue
- The bug has been reported before in the bounty (it's a duplicate)
- The bug is not a security flaw, but rather a functionality issue
- The report is unclear or hard to understand
- The report is about a bug type which is out of scope
- The report is about a product which is out of scope (e.g. third-party software, or old versions of your products)
- The report describes something considered to be a feature, not a bug
- The researcher has broken the rules of your bug bounty, e.g. by testing user credentials they have found, or by destructive actions
- The bug can't be reproduced on a testing machine

It is worth the effort of preparing these messages for your staff to use, because it makes it easier for them to communicate frequently and promptly with your bounty researchers. This in turn makes a better experience for the researchers, making your bounty program stand out against the rest in the global marketplace.

3.6 Decision fatigue

On a related note, decision fatigue is another challenge for your bounty staff. Every instance of having to make a new judgement call costs them time, effort and consistency. When a bug needs to be handled urgently - for example, in case of a critical vulnerability - then having a clear plan of what to do becomes essential.


e.g. "How do you handle a Critical bug?"
New CVSS-Critical Bug Bounty Report
Got More Information
Can You Reproduce The issue?
No
Go Back To The Researcher For More Information
Not An Issue?
Close the report.
Yes
Is it a CVSS-Critical?
No
Follow the standard process.
Yes

To prevent decision fatigue, establish a shared page of standard procedures and protocols for handling situations that crop up in your bounty program. Any time one of your bounty staff has to make a new decision that isn't covered by the existing documentation, they should update the docs to cover that case in the future.

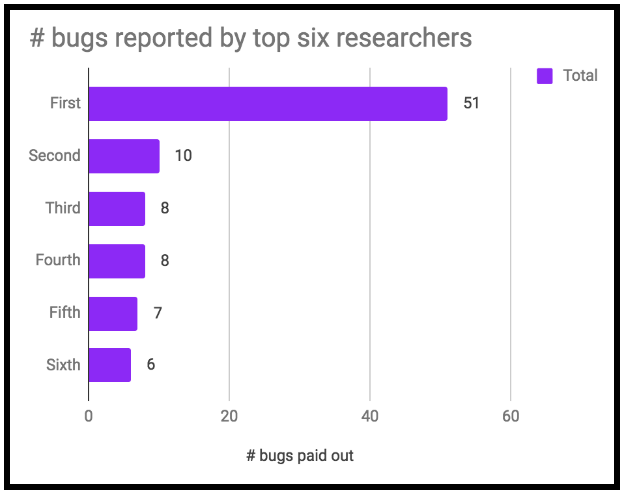
Flowcharts are an excellent way to express procedures! For example, here's an excerpt from a flowchart used at Atlassian to handle critical-severity bugs.

Other situations you may want to establish rules and procedures for are:

- When a researcher asks if they can publish a blog post about finding the bug, what is required in order to say yes?

- When does a reported bug require a CVE ID?
- When does a reported bug require a security advisory to be published?
- When should your country's Computer Emergency Response Team (CERT) or other government agencies be informed of a bug?
- How should you determine the severity of a bug? What scoring system should be used?
- How should you determine the bounty reward on a bug? Is there a tiering system for each product, each bug type, or each severity class?
- When are you allowed to award bonus points, extra money or swag to a researcher?

3.7 You're dependent on a small group of researchers



A bounty program lives or dies by its researchers. But because researchers tend to be specialists, you may find that one or two decide to specialize in your company's products. A skewed distribution of reports (as in this graph) is not unusual.

If you find that your bounty program is carried by a few of these particularly prolific researchers, this puts you in a precarious position: you're dependent on the efforts of just a few people. If these major players leave your program, the bulk of your bug reports are lost.

Part of the trouble is that increasing the bounty rewards may bring you more bug reports. It *doesn't* necessarily bring you more researchers. You might just receive more bug reports from the same group of researchers.

However, you *can* reach more people by advertising your bug bounty and holding public hacking events. Consider options like:

- Putting up posters at universities' IT labs or other security-focused public spaces
- Online advertising targeting penetration testers and security researchers
- Sponsoring the catering or room hire for a local security seminar or workshop, in return for advertising your bug bounty
- Having your security staff mention your bug bounty during their own seminars

It's also a good idea to run your bug bounty on an existing bounty platform (see Section 3.1), since this essentially gives you free advertising to any security researchers already working on that platform.

3.8 Bug bounties involve necessary yet repetitive tasks

Alas, along with every elegant proof-of-concept submitted to your bug bounty comes a suite of necessary but repetitive tasks:

- Your organization's developers need to know the due date for the bug being fixed
- The developers also need to be reminded when the due date gets closer

- Team leads may want to be alerted to any overdue bug fixes
- Bounty staff need to be notified when a bounty researcher responds with more information or requests
- Bounty staff need to be notified if an urgent critical-severity report is made
- The researcher needs to be informed once their bug has been fixed

This ceaseless stream of back-and-forth information is extremely repetitive and boring to manage manually. So *don't* do it manually. Program an automatic service to handle as much of it as possible. Apart from making your bounty staff's jobs more interesting, these automated pings and reminders form the driving force behind ensuring that bugs reported to your bug bounty do actually end up fixed in production.

4. Conclusion

Running a bug bounty program has numerous benefits, including:

1. Reducing the number of bugs exploited in your software;
2. Leveraging international, specialist talent to do vulnerability testing on your software;
3. Helping your products meet security standards; and
4. Ultimately making your software more secure.

Even if you can't run a bounty with monetary rewards, you can still run a "vulnerability disclosure program" where researchers are rewarded with branded swag or public recognition (via a blog post or hacking hall of fame). At the very least, you should provide contact information for researchers to report bugs to you, such as a security@yourcompany.com address.

To avoid problems in your bug bounty, your choice of bounty platform should:

1. Be an **existing** platform;
2. Offer services to **filter out invalid reports**;
3. Provide **reports and statistics**; and
4. Have an **API** thorough enough to pull out data and control the bounty.

Once your bounty has been set up, you should then:

1. **Start small**, limiting the number of initial reports;
2. Use the platform's **filtering services**;
3. **Document** all procedures and rules;
4. **Pull data from the platform** to inform your decisions;
5. **Advertise** your bounty to wide your researcher base; and
6. **Automate** any repetitive tasks.

Best of luck, and happy bug hunting!