hackerone

The world's leading Vulnerability Coordination and Bug Bounty Platform

Bug Bounty 5 years in

This post by Collin Greene originally appeared on Medium on June 15, 2016.

The following is advice from Uber security team member, Collin Greene, on how to successfully launch and run a high-quality bug bounty program. He includes tips on how to prepare, what to expect in the first few months, things that will go wrong and why ultimately he believes in bug bounty programs, from experience launching and running Facebook and Uber's programs.

That said a bounty program doesn't replace anything you are or should be doing security-wise, it augments it.

In my first few months at Facebook I found 24 security bugs which was a respectable haul. The following quarter we launched a bounty program which promptly found 71 bugs. The pragmatist in me hated, but could not ignore, that my time spent fielding bounty reports resulted in more security than spelunking through the codebase.

So began my professional relationship with bug bounties.this paper, we share 3 best practices for doing just that, honed through our experience managing bug bounty programs with hundreds of security teams using HackerOne.

Why I like bounty programs

Security is hard in a large part because it is unfalsifiable. I cannot say with conviction that I've found all the bugs in any given program. So I try to stack the odds in my favor by layering systems that surface bugs: design reviews, code reviews, type systems, program analysis, external security audits, blackbox scanning, etc. Bounty programs sit at the end of this process which means every bug found is one that slipped past everything thrown at it.

I witnessed my first real security effort at Microsoft in early 2000s. After slammer and code red Microsoft made the unprecedented move of investing a billion dollars into security. Turning this billion dollars into "security" manifested itself as vacuuming up all the security talent they could find, sticking us in a room together with the code and basically crossing their fingers that good bugs shook out.

In contrast bounty programs are cheaper, produce a roughly equivalent quantity (if not quality) of bugs and companies don't have the headache of managing a horde of consultants.

An operating system is not the most amenable target but if done today I bet lots of those same bugs could be found via a bounty program (of which Microsoft has a few)

Having a large pool of people with different perspectives and knowledge attack your software brings a diversity of bugs that has surprised me. You get this in spades with a bounty program as each researcher has their own areas of expertise.

Bounty programs are a good fit with modern development practices. In the old days you planned in months of software development, you hit your deadlines, it went to QA, then it went in a box. Today everyone does agile, ships faster and QA teams are near extinct[1]. Bounty programs fit this environment like a glove as fixes can be pushed quickly and act as informal QA.

That said a bounty program doesn't replace anything you are or should be doing security-wise, it augments it. It is the icing on the Secure Development Lifecycle cake. A bounty program does not replace consultants, they are different tools to achieve different things. If you want a whole lot of bugs though bounty programs are a good bet however.

Bug-elimination research, like other user-interface research, is highly nonmathematical. The goal is to have users, in this case programmers, make as few mistakes as possible in achieving their desired effects. We don't have any way to model this - to model human psychology - except by experiment. We can't even recognize mistakes without a human's help.

- DJB

How to prepare

You are onboard and are eager to launch. Awesome. Here is what I would do.

- Gut Check—A poorly run bounty program is worse for everyone than no bounty program at all. Make sure your security house is in order and have the basics before launching. Don't succumb to pressure to launch before you feel ready.
- Define your scope really really really well. Default deny is what you want. Have a paper trail of your programs scope changes.
- Make sure you can route/triage issues effectively in your company. Look for gaps in ownership. Large swaths of unowned old code mean that you will be the one writing the diffs to fix the inevitable flaws found.
- Meet your PR person, customer support person, legal person, and whoever watches your logs. You will need all these contacts at some point.
- Launch in private mode first.
- Do a sweep for the low hanging fruit being and sure to read this first. There is probably a lot of cruft hanging around like forgotten wordpress instances, secrets in github gists, unused .swfs, outdated libraries and forgotten assets in your IP space.
- Read up on what issues might be found, how researchers will get ramped up and what researchers themselves want in a bounty program
- Define your SLA with researchers even if you never publicly commit to it. For example: Responding to all fresh reports within a week.
- Get a sense of what is being found in other bounty programs.

- Ensure you have a good internal taxonomy of security flaws. You will need to explain the same security issues many times. One really good explanation goes a long way. Be sure that it includes impact that explains the worst case scenario.

- Figure out your bright line for demonstrating an issue but not introducing risk or breaking the rules. Many of the negative incidents have been a researcher going too far to demonstrate an issue.

- Prepare for success. Pre-bug bounty a good month for the Uber product security team was finding ~25 bugs. In our first month we had 157 legitimate issues dropped on our head. Finding these 157 legit issues required reading 1809 submissions. To have any chance of handling this requires a big allocation of time to handle the initial burst of reports and an oncall rotation or process to handle the ongoing work. Expect report volume to die off starting in the 3rd or 4th week.

- Consider if you want to use a bug bounty platform like HackerOne versus doing it all yourself. Having done both I heartily recommend using a robust platform -- especially as you look to scale your program. Even figuring out how to pay researchers across the world is a headache you probably want to avoid. A platform can help provide hacker transparency so you can begin to invite them based on their reputations or skills to save time and eliminates some noise.

- Pricing of issues is the biggest lever you have to effect quantity of reports. Starting with lower payouts and ratcheting up once things are running well is advisable.

Bug bounty skillset

The people on your team signed up to be security engineers and their job just became part customer service. The skills required are:

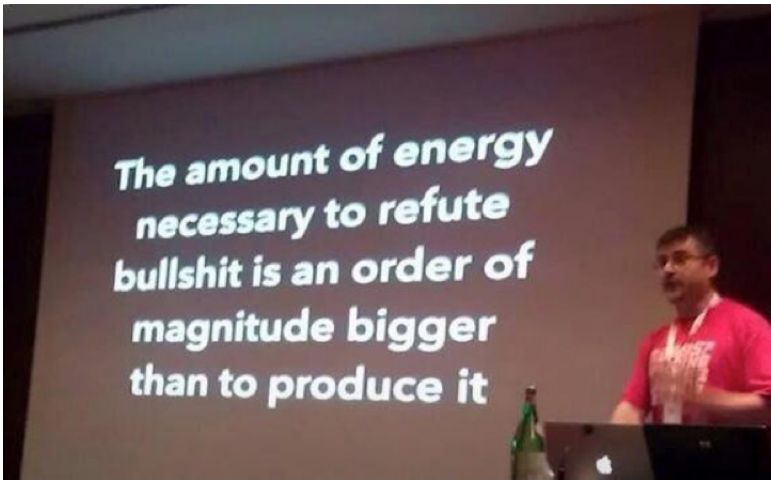
- Empathy—The submitter has less information than you, be kind if they are wrong. This could be their first attempt at what could be a blossoming career in security. Praise cool bugs.

- Wordsmithing—Asking precise questions to coax out the heart of an issue is hard. English is not everyone's native tongue and technical communication is hard. Technical communication in an inherently adversarial situation with some emotional charge ("you are wrong, no you are wrong!") is especially hard.

- Patience—Reporters can test your patience in a million different ways and it is your job be the bigger person. Some reporters need kid gloves. For tricky issues I like to let my response simmer for a few hours before hitting send.

- PR awareness —Don't hit send on anything you wouldn't want in the New York Times but balance that with not being a total PR robot. Give satisfying technical explanations as often as time allows.

- Awareness—Recognize how much effort to spend per report. Every researcher deserves to be heard, but they're not entitled a permanent audience. The joke: "sometimes the real vulnerability is the denial of service against the teams time" is appropriate.



Its alive!

You are up and running and the bugs are falling like rain. It is chaos just keeping up but it is good chaos. The first few months are all about just keeping up.

- Every bug is a lesson. Learn from it and use it as signal to find similar issues.
- Use the pseudo-public nature of discovered bugs to help encourage engineers to fix them quickly.
- Communicate outward. Lessons to the engineering org around what bug bounty is surfacing are pure gold. I do a monthly newsletter at Uber about the most interesting/representative/serious/funny bugs found and people love it. In the days following its release there is a tangible uptick in stale security bugs getting fixed. The excellent Android Security Bulletin is a great public example of this. Security is sexy and people love to hear about it and be a part of it, play to that.
- Be worthy of trust. Every researcher is deciding to spend their time on you with no guarantee of finding anything. In response you should be as transparent as possible with reasoning, upfront with rule/scope changes and aim to treat all researchers equitably.
- Reward fairly. Reward primarily on the impact of the issue. It is tempting but rarely correct to reward primarily on cleverness or if the vulnerable code was built vs bought. Be consistent with payouts over time. If possible err on the side of generosity. Pay bounties at the time of validation; don't wait for bug remediation.
- Stamp out common false positive submissions.
- Your best reporters could be more productive than your worst co-workers. Find ways to celebrate them with swag, nice emails, promoting blog posts, etc.

- Quickly define areas that are not consistently bringing you the bugs you want and mark them as “out of scope.” There is a balance here but no one wants to spend weeks of effort on low impact microsites.
- Engineer a process to dedupe bugs and be consistent in payouts. Usually this is a weekly meeting accompanied by a long list of previous payout decisions and dupes.
- Be wary of burnout among your team. Ensure the burden is evenly spread.
- Keep an internal running list of things your team can improve on and feedback from hackers.
- Bug bounty will be a hell of a stress test for your engineering organization. Keep track of who stepped up to help fix issues and who was unhelpful or slow. You will quickly learn the good/bad engineering teams inside your company if you don’t already know.

The hope is that over time bugs in your software will become harder to write and you will be compelled to raise payout amounts for your program.

The suck

Things will go wrong. It's ok. It happens.

- Triage takes time. So much time. At both Facebook and Uber we paid people to help us run our bounty programs and it still took up a ton of time. Expect the time commitment to be 2–3 people fulltime. Bug bounty providers like HackerOne also offer triage. If you go the consultant route you will likely pay more in consulting time than you do in payouts. The price you pay for the good issues is the cost of filtering out the junk.

- Accept some researchers will seem ungrateful, some will be irritating. Some will try to subtly and not so subtly blackmail you[2]. Its all in the game, yo.

- External researchers often don't appreciate how long it takes to fix something in the real world and internal teams don't appreciate the severity of issues. Being the middleman between the two can be disempowering. This can be remedied somewhat by allocating time to just fix issues yourself.

- Much like that giant trash island floating in the ocean accept that you will end up with a big blob of lower-priority open security issues. The burden for followup/pushing on this issues falls to you, the security team. This is a drag but don't be discouraged, this means the program is working. For every sweet RCE you will have a handful of "no secure flag on cookie"

- Eventually reports will die down and you will need to invest time in stirring up interest. This is a great paper discussing the dynamics of competing bounty programs, basically more researchers = more bugs and researchers often take time out to investigate a new program given it pays competitively.

- As you get to know reporters and notice patterns in reports you will look for shortcuts to dismiss issues (“looks bogus”). Don’t do this. Good bugs can and do come from anywhere, a nontechnical polish carpenter submitted one of the best Facebook issues and an everyday Uber driver found a way to let him only accept higher dollar fares that went undiscovered for years.

- Be prepared for both good and bad press as both will inevitably happen. Its another cost of running a program. For the most common case of a researcher upset at how you handled a bug the HackerOne public disclosure is a godsend.

Keep it spicy

Run a solid, fair, competent bounty program and researchers will keep coming back

Here are some things I've tried and how they fared

- Double bounties on areas to encourage attention. Didn't work. We got an uptick of less than 1% in that area.

  Reward programs —Like a punchcard at a sandwich shop, offer bonuses for hackers who keep reporting valid issues. This might have some pull to keep researchers in a given program. Too early to tell but I am optimistic.

- Spiffy credit cards—people loved them but they were a pain to setup and run

- T-shirts, swag, gifts. You can never have enough special t-shirts or pens or whatever.

- Hall of fame. People love leaderboards more than I could ever imagine, try to have one of these.

- Paying for researchers to visit your HQ or Defcon/Blackhat. This is a pretty good perk that everyone seemed to like a lot.

Cause for hope

Bounty programs improve security and that is swell. More powerfully I see them as the way to train the next generation. I was profoundly lucky to have wonderful mentors who answered millions of my questions on irc or at 2600 meetings. Even with such benefits when I wanted to Actually Hack Stuff I had to go do it, and take on the commensurate risk.

Growing up today, as someone always interested in hacking, I would have spent all my time on capture the flag and bug bounties. Getting to hack real things with no risk of prosecution and a shot at some money for my efforts is a great thing. Bounty programs are a safe and sanctioned way to get into security. If you run a bounty program look for teachable moments in responding to a researcher who may not fully understand something.

We as a security community are a small tribe and bounty programs are the practical ladder by which much of the next generation will get into security professionally.

In a sea of snakeoil and grand claims a robust bounty program is a statement of confidence in your security —openly inviting scrutiny is admirable.

You will find bugs. You will become less of a faceless corporation to the security community. Tracts of your codebase will be illuminated as needing security attention. You will be surprised by the things you missed and become a better security engineer for it.

I do not speak for employers past or present. I thank Ryan, Katie, Erik, Zac and Nathalie for helping me on this article.

Footnotes

A problems of its own in my opinion.

The story here was someone submitted a claimed XSS, admitted it doesn't fire, starts a reddit thread saying Uber bug bounty is a scam with a big donate here link. Takes it all down out of shame a few hours later when the report is made public.

Thanks to Ryan McGeehan.