BUG BOUNTY FIELD MANUAL

How to Plan, Launch, and Operate
a Successful Bug Bounty Program

BY ADAM BACCHUS

hackerone

Table of Contents

Welcome, Friend, to The Bug Bounty Field Manual!

You are here because you want to learn all about this bug bounty stuff. You're ready to get ramped up immediately, but you have questions, uncertainties — maybe even trepidations. Well, you've come to the right place. This manual was created to teach everything you need to know to plan, launch, and operate a successful bug bounty program.

The illustrious bug bounty field manual is composed of five chapters:

1. **Assessment:** See if you're ready for a bug bounty program

2. **Preparation:** Tips and tools for planning your bug bounty success

3. **Champion Internally:** Getting everyone excited about your program

4. **Launch:** How to navigate a seamless program kickoff

5. **The Post Bounty Era:** Operating a world-class bug bounty program

Spinning up and executing a successful bug bounty initiative is no small undertaking! Thankfully, you're not alone in this journey.

Hi! My name's Adam Bacchus, and we're going to get to know each other over the next few minutes, so allow me to tell you a bit about myself.

I'm currently the Chief Bounty Officer at HackerOne, and before that, I helped run bug bounty programs at Snapchat and Google, and before that, I did some hacking myself as a security consultant. I'm passionate about helping organizations start and run successful bug bounty programs, helping hackers succeed, and generally trying to help make the Internet a little bit safer.

In my spare time, I enjoy fire breathing, playing music, and mixing drinks.

This elephant-sized guide covers what you need to know about how to plan, launch, and operate a successful bug bounty program.

Don't worry, we'll take it one bite at a time. You ready? Let's do this!

BUG BOUNTY BENEFITS

Sometimes it's helpful to start with the end in mind. Imagine yourself enjoying a well-deserved holiday in Hawaii.

You're sitting on the beach, listening to the gentle roll of the ocean waves, and you feel the fine sand in between your toes. Right as the sun sets, you sip your Mai Tai thinking about how *amazing* your experience has been since launching your Bug Bounty Program: after a seamless launch, you are receiving very useful bug submissions that your prior efforts *never* found, your engineering team is now able to secure your systems faster and cheaper than ever before, and you just launched a competitive bounty challenge for your top hackers.

Ahhhh, now doesn't that sound nice?

Cheekiness aside, all of this can be achieved by a well-executed bug bounty program. (As far as the Mai Tai goes, you're on your own.)

In fact, the median time to receiving a vulnerability report from launching a program is less than 9 hours! You will get a great volume of actionable, unique reports at a fraction of the pen testing budgets of yesteryear.

I'm about to dive into the really good stuff, but before we get into the mechanics of bug bounty programs, we first need to do a self-check to see if you are *truly* ready for one.
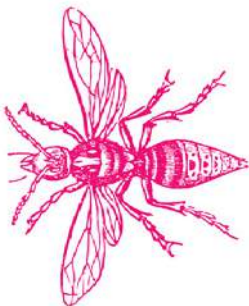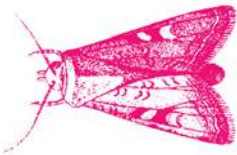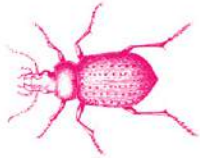
ASSESSMENT

After having run or been a part of dozens of bug bounty programs, I can tell you that the experience and value derived from them heavily depends on taking a moment to assess where you're at today.

An initial self-assessment is critical to ensure you don't jump off the deep end too early. Launching a program without this assessment can actually make things worse! Not to worry, though; to help you determine where you're at and what type of program best suits your needs, we've created an assessment questionnaire. It only takes a few minutes to fill out, and trust me - it's worth it.

AM I READY FOR BUG BOUNTIES? $\longrightarrow$

When it comes to working with hackers, there are a few different options. The assessment questionnaire will help you gauge which is best for you. What are these options, you ask?

One of the biggest distinctions is whether or not you offer monetary rewards, or "bounties," to hackers outside of your organization that report valid security flaws to you. Many organizations start off without bounties, with what's called a **vulnerability disclosure program (VDP)**. In a **bug bounty program (BBP)**, the stakes are a bit higher, as you offer varying monetary rewards for issues identified and reported to you. Another factor is time; some choose to start with a pilot program to test the waters, which can last anywhere from a month to a year.

This guide is about running a bug bounty program, but the HackerOne platform can also be used for vulnerability disclosure programs and crowdsourced pen testing. You can even start off with buying just a single vulnerability!

Some HackerOne customers leverage time-bound pilot programs in lieu of a pen test. This is another great way to dip your toe into bug bounties.

We encourage our customers to run the bug bounty program that works best for them. Whether private or public, time-bound versus ongoing, a program offering cash bounties or sticking to swag only - we work together to create the best blend for your organization and will grow with you.

Alright... now let's begin our journey into bountyland!

Want to learn how you can get the leading vulnerability disclosure platform for FREE? Click here, give us a few bits of info and we'll be in touch ASAP!

$\longrightarrow$

PREPARATION

This is the most important step in any bug bounty initiative. As mentioned earlier, it's tempting to jump straight off the deep end, but whether or not you do a bit of prep work can make or break your experience. Organizations that hit all the right notes get amazing ongoing value out of their program. Those who don't can end up getting burned.

Chapter 2.1: Vulnerability Management

As we alluded to in the assessment questionnaire, you likely already have some vulnerability management (VM) processes in place (i.e. ensuring vulnerabilities are identified and fixed in a timely manner). In any VM process, you're going to have streams of vulnerabilities coming in from different sources, such as: automated scanners; issues uncovered by security engineers, developers, or external consultants; or even someone publicly posting a zero-day!

You'll then need to prioritize these issues, typically based on severity. Once you've prioritized them, you have to hunt down owners to ensure the bugs get fixed in a timely manner. When starting a bug bounty program, you're essentially adding a new stream of bugs into your existing VM processes.

It's important to ensure you have a solid process in place for communicating vulnerability information to the right owner, including severity of the issue, and expected remediation timelines.

When you launch your bug bounty program on HackerOne, you have the ability to assign a severity to each report, as well as integrate with common bug tracking systems (e.g. JIRA, Assembla). This streamlines bug reporting and triage efforts. If you'd like, you can even invite the developer responsible for the fix directly to the report on HackerOne!

Some organizations shy away from bug bounty programs because they feel they aren't ready for a new stream of bugs. I often hear sentiments such as:

"Our security team is already swamped, how can we find time to run a bounty program?"
"We have a hard enough time getting developers to fix security bugs in a timely manner today, and you want me to pile more security bugs on top of that? Are you *@3$ Crazy!?"

Fortunately, a bug bounty program can actually help improve the security culture in your organization!

This is a coordinated and purposeful dialogue that you spearhead with a key internal stakeholder: the engineering team.

As soon as you can, give your developers a heads up that bugs are coming and that it's a good thing. Explain that any bug identified through your program is live in production, and fortunately a friendly hacker has taken the initiative to report it to you. This means a malicious attacker could also find the same bug, which creates real world motivation for more timely remediation. It also helps to prioritize and improve security culture throughout your organization, and provides valuable data on where your current security processes have failed (more on this in Chapter 5, The Post Bounty Era).

Chapter 2.2: Allocate resources

Before you do anything else, lock down the resources necessary to get up and running. This includes allocating time for yourself as well as securing resources from your security team, and your immediate organization.

Chapter 2.2.1: Choose a leader, build your team

First, you'll need a **bug bounty leader (BBL)**. This person is the ultimate owner and champion for your bug bounty initiative. It might even be you! The BBL is responsible for the success of your program, but they can't do it alone. Once the BBL is established, they need to form a **bug bounty team (BBT)** that will support the ongoing program. The BBT is typically composed of members of the security team, but there's no reason you can't include other security-minded individuals from your organization.

Chapter 2.2.2: Share the load

Running a bug bounty program requires a commitment from your BBT to perform a large variety of tasks, such as triaging bug reports, communicating with hackers, defining and dishing out bounty amounts, vulnerability management, and more!

Many teams already have a full plate as part of their regular job duties, so time management and batched work is the name of the game.

The best way to share the load is to set up a weekly on-duty rotation.

BUG BOUNTY DUTY

	S	M	T	W	T	F	S
Dade	1	2	3	4	5	6	7
Kate	8	9	10	11	12	13	14
Joey	15	16	17	18	19	20	21
Ray	22	23	24	25	26	27	28

Whoever is on bug bounty duty is responsible for all operational work that week, as well as continuing progress on any strategic improvements to your program.

Chapter 2.2.3: Brace yourself, bugs are coming

In addition to setting up an on-going rotation, you'll want to clear out the calendars of your BBT for the first week when you launch. There's often a large spike in activity at the launch of a program (like Yelp experienced), and this is also the time when you'll quickly identify and course correct for any hiccups in your program.

Chapter 2.2.4: Choose your platform

Fortune 500 companies like Qualcomm, and hot tech companies like Uber, deploy bug bounty programs with the help of a bug bounty platform. It's possible to run a program in-house, but things like tracking status of reports, processing payments, and attracting and maintaining relationships with top hacking talent is a lot easier with a bug bounty platform.

I may be a little biased, but I've heard this "HackerOne" thing is pretty great; I mean, you could try to recruit 85,000 hackers on your own... or you could sign up for HackerOne to quickly and easily access the largest pool of top hacking talent in the world. But don't take my word for it - see what our customers have to say.

Before real bug reports start flowing in, familiarize yourself and the rest of your BBT on how to use the HackerOne platform. If you've signed up for Professional or Enterprise, a HackerOne specialist will help train your team.

Chapter 2.3: Bounty Process

One of the first questions most people have when starting a bug bounty program is, "how much is this going to cost?" Not surprisingly, one of the first answers you'll usually get is "it depends!"

There a wide variety of factors to take into consideration, such as your initial scope, how you assess severity, and whether you are running a pilot vs. an ongoing program.

We have a great post on our blog: Anatomy of a Bug Bounty Budget for a deep dive on this topic, but feel free to continue reading for high-level guidance.

Chapter 2.3.1: Bounty Tables

The simplest way to approach this is to set up a bounty table. A bounty table illustrates how much you are willing to pay for various bugs, helps set expectations for hackers, and gives you and your team a guideline to ensure fair and consistent reward amounts.

Typically you want to pay out based on the severity of the issue identified. HackerOne provides CVSS (Common Vulnerability Scoring System) scoring in-platform to assist with this. Both hackers and teams can use CVSS to calculate a severity, or simple pick one from Low, Medium, High, or Critical.

Available in every product tier, we show you the average award amounts across the hundreds of programs on our platform. Integrated with the recently launched CVSS severity setting on reports, our Bounty Statistics feature automatically shows you the median bounty across our platform for that severity, as well as what programs at a competitive and top level are paying out. Here's an example of bounty statistics for a Medium severity report:

At the time of this writing, here are the statistics for bounties across all severities:

SEVERITY	MEDIAN BOUNTY	COMPETITIVE BOUNTY	TOP BOUNTY
Critical	$1,400	$9,000	$15,000
High	$500	$2,500	$4,000
Medium	$150	$600	$1,500
Low	$100	$250	$500

I'd recommend starting with around (or slightly above) the median values. It's best to ramp up your bounties as bug scarcity increases, as opposed to going too big initially. Once you have a feel for your flow of bugs, you can increase your bounty ranges. Higher bounties results in more attention, especially from higher ranked hackers with polished skill sets.

In December, 2016, Shopify paid out $368,800 in ONE DAY to hackers. Was this a failure? Unexpected? No. Quite the opposite.

READ SHOPIFY CEO'S RESPONSE ON HACKER NEWS →

This is a great starting point, but it's always great to refine further over time based on the issues that matter most to you. For example, perhaps you would pay little to nothing for XSS on a sandboxed domain with no sensitive content or functionality, but would pay top dollar for an XSS on your flagship website.

For example, Twitter splits their scopes into "core" properties and "all other," then outlines what typical bounties look like for different classes of bugs based on their impact. At the time of this writing, an XSRF on a "core" property on a "critical action" will

net a $2,500 bounty, but any other XSRF on a non-core property results in a much lower bounty of $140. Money talks! This is a great example of using bounty structure to incentivize hackers to target and report the issues you care the most about.

Bounty amounts are certainly a hot topic, but HackerOne arms you with data and access to experts in order to come to the best conclusion.

For those who want that extra insight and consultative approach, our Professional, Enterprise, and Fully Managed customers have access to a HackerOne representative who will walk you through the entire process.

Chapter 2.3.2: Define your bounty awarding process
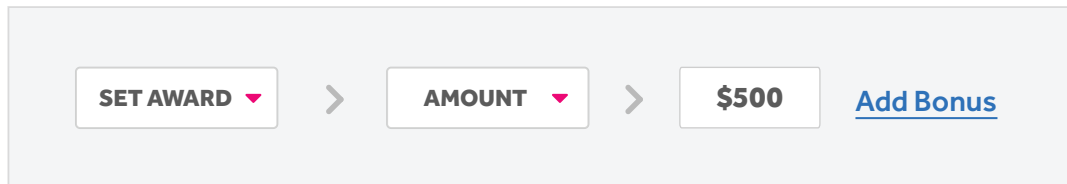
So you've settled on how much to pay for various types of bugs. But how do you actually pay out?

Without HackerOne, you and your finance / ops team will have to manage the ENTIRE process. This includes compliance (like OFAC and AML laws), taxation (getting W9s from hackers, issuing 1099s at the end of the year), covering processing fees, dealing with currency exchange rates, and other messy logistics.

All this to pay a bounty to a Hacker. Imagine dealing with that 100x over, with different countries around the world!

Thankfully, our no-hassle bounty platform handles all of this for you!

HackerOne's payment process is painless - you can either set up a prepaid balance to pay bounties out from, or add a credit card to your account for payments. After you've done that, it's as simple as defining a bounty amount on the report:

| SET AWARD ▾ | > | AMOUNT ▾ | > | $500 | Add Bonus |

You can even add a "bonus" on top for particularly clever or well-written reports.

Chapter 2.4: Determine your Service Level Agreements

It may seem counterintuitive, but running a bug bounty program is akin to providing a service to hackers who participate in your program. The most successful bug bounty programs have well-defined service level agreements (SLAs) that they share with hackers on their rules/policy page. The three big SLAs to consider are:

1. Time to triage / response on a new report
2. Time to bounty (after validation)
3. Time to remediation

Let's spend a minute to look at each of these categories a little closer:

Time to triage / response on a new report depends on the resources available to you, your team's expertise level, the sophistication or esoteric nature of the bug, and your report volume. HackerOne recommends an SLA of two business days for time to triage.

In his *Hacker Herding* blog post, Dan Adams, Web Application Security Specialist at Sky Betting and Gaming suggests to timebox triage. He says, "It's essential therefore in order to keep to pre-agreed SLAs (and deliver on those hacker expectations) to clearly delineate and set aside time dedicated to triaging and progressing vulnerability reports."

Time to bounty (after validation) depends on which payout model you choose. Some of our top programs pay bounties within a few days of report receipt! At a minimum, I recommend determining and issuing a bounty within 1-3 weeks from time of triage.

Time to remediation depends greatly on the severity of the issue and the complexity of the fix. Of these three SLAs, time to remediation probably has the most wiggle room, but it's still very important! Security only improves when bugs are fixed, not when they are found, and many hackers participate in bug bounties because they want to see security improve as a result of their work. HackerOne recommends the following SLAs for remediation timelines based on severity:

- Critical = 1 day
- High = 1-2 weeks
- Medium = 4-8 weeks
- Low = 3 months

Chapter 2.5: Craft your policy/rules page

Transparency between hackers and security teams is vital to a successful bug bounty program. The "front door" for hackers to any bug bounty program is the security page, which commonly contains your disclosure policy, rules of engagement, scope, and other important information. Twitter, Uber, Snapchat, Coinbase, and Mail.ru (among many others!) have great security pages that incorporate elements outlined in this guide.

Three benefits of having a rock solid security page include:

- It steers hackers towards the scope and vulnerabilities you care most about
- It sets clear expectations around bounty pricing and timelines
- You garner more trust (and participation!) from top hackers

At a high level, you should ensure your rules page includes the following elements:

1. A well-defined scope
2. Bounty structure
3. Qualifying vs. non-qualifying vulnerabilities
4. Service level agreements
5. Eligibility/participation requirements

For a further deep dive on this subject, check out our in-depth guide on how to craft your rules page.

→

James Kettle, Head of Research at Portswigger (makers of the popular Burp Suite tool) says the security page is something that should be carefully crafted.

"

I spent a lot of time on it. I think people especially like the 'known issues' section as it saves them from wasting time reporting something that we already know about.

- JAMES KETTLE, HEAD OF RESEARCH AT PORTSWIGGER

And for all you overachievers out there, the NTIA Safety Working Group also has some good advice on this topic.

CHAMPION INTERNALLY

Thus far we've gone through a lot of the blocking and tackling basics for the planning part of your bug bounty program. Chapters 1 and 2 describe many of the planning details for your bug bounty program.

You've determined you're ready, primed your vulnerability management processes, defined bug bounty roles and responsibilities, gone through the details of your bounty process, set up a budget, created your policy, and now - now you're ready to sprint ahead to bug bounty glory!

Well, not quite. Have you gotten management approval yet for the budget? Did you talk to all pertinent internal team members and discuss the bug bounty process with them? As you can see, and are now becoming more aware, championing a bug bounty initiative requires a lot of people to be on-board!

It all starts with you, though. Your passion and commitment to launching a program can (and will be!) infectious.

Think of it like you're Luke Skywalker and your engineering and development teams are Han, Chewbacca, C3PO, R2D2, and Leia - you can't win the battle yourself, but without you, the whole galaxy will fall apart.

It's your job to get the supporting cast on board. Part of how you do that is to make sure everyone is aware of how much value will be derived from these efforts.

Beyond the core cast members, you have PR, legal, and executive leadership. Taking our Star Wars analogy further, let's say they're Lando Calrissian, Obi-wan Kenobe, and Yoda (Lando is obviously the PR guy).

Involving everyone, gathering feedback, and showing them the potential of this initiative will go a long way towards launching and running a successful program, as well as result in greater feedback loops and overarching improvements to your security posture (more on this in chapter 4 and 5).

So this sounds great and all, but how do you actually go about doing it?

Your approach will vary depending on the audience. Here's some high level advice on factors to consider when approaching common teams at most organizations.

3.1 Security Team

Depending on the size of your security team, maybe everyone's already on board! If not, you'll want to consider this the center of your "onion," with additional layers around it: engineering, finance, legal, and PR. If the core crew isn't convinced, you're going to have trouble taking off.

A bug bounty program is usually most relevant to whoever works on product security. Since your program will identify every bug that fell through the cracks of your product security team's processes, there can be an inkling of defensiveness. You may hear sentiments like "Why do we need this? Our processes are fairly robust."

Identifying these bugs and tracing them back to their root causes will provide invaluable data for you and your team to discover where things went wrong, and course correct as needed.

The thing is — no matter how amazing your security processes are — there will always be bugs that slip through the cracks. Augmenting your security team with the help of friendly hackers serves as an excellent safety net, as well as helps identify areas where your security processes have failed.

In addition to your product security, if you have a team that works on detection and incident response, they'll need to be in the loop as well. Imagine it's Friday night, you're about to go home and relax, and all of a sudden your intrusion detection system (IDS) starts blowing up in your face. It looks like you're being hit by automated scans from 10 different countries. What's going on!? It's critical you inform your team ahead of time that additional activity may be generated due to normal testing so they aren't caught by surprise.

3.2 Engineering

Whatever your engineering team is building (web apps? mobile apps? IoT?), nobody likes unexpected work. When your bug bounty program starts picking up steam, there'll be plenty of security bugs to fix on a regular basis. Some security teams take on the herculean task of fixing security bugs across the entire organization, **but the best way to scale security is to empower all teams to fix the bugs themselves**. This is a huge topic that could warrant its own manual, but in short, you'll want to work with leaders across all departments of engineering to ensure they are aware of your bug bounty initiative, and that they understand the benefits their teams can derive from it. Your bug bounty program *will* uncover flaws in engineers' work. As mentioned previously, this can cause some defensiveness, but leveraging this data to identify root causes of systemic security issues in your environment is an immensely powerful asset. In addition to improving your security team's capabilities to identify these flaws, you'll be able to use this data to help show engineers the real world impact of not embedding security into development processes.

This sounds great in theory, but what happens when the bugs start hitting the fan? Having spent a few years working on vulnerability management at Google, I can tell you from experience that the more of a heads up individuals and teams have around incoming work on security, the more likely it is you'll be successful in convincing them to allocate adequate resources towards fixing bugs.

It's impossible to predict exactly how many bugs will flow into each area, or how long each of them will take, but it is within your power to ensure your bug bounty machine is operating smoothly enough that you can identify and notify owners of affected products of vulnerabilities as quickly as possible. As mentioned previously in the vulnerability management section, you'll want to have a common language with engineers to communicate the severity of various bugs, as well as expected remediation timeline.

Any bug found via your program is live in production; if a hacker was able to find it and was nice enough to report it to you, a malicious actor is probably out there sitting on the same bug. Even in organizations that don't normally have the best remediation timelines, this perspective can help improve security culture throughout your organization.

3.3 Finance

Fortunately, HackerOne makes it easy to pay hackers, so the process with your financial team should be straightforward! When it comes to payments, you have two options:

1. Add a credit card to your HackerOne account that gets charged for any bounties paid out

2. Make a deposit in advance which bounties can be paid out from

Most organizations go for option #2. Whenever your team decides on a bounty, you simply set the amount you'd like to pay on the report, and it is withdrawn from your balance. Easy as pie! We recommend stashing away three months' worth of bounty budget at a time. With HackerOne taking care of payments, you don't have to worry about compliance, taxation or figuring out how to pay a hacker in, say, Siberia who doesn't have a mailing address. HackerOne also collects the appropriate tax forms, so you don't have to worry about it. For more information, check out this support article on the topic of payments.

3.4 Legal

Taking a look back in time, we can see the way that organizations interact with hackers has changed immensely (and for the better!). One great example is Samy Kamkar, and the XSS worm he built that exponentially propagated through MySpace overnight, affecting one million users, and forcing MySpace to temporarily shutdown. Now - admittedly - he probably shouldn't have done something quite so destructive to a production environment. That said, the repercussions were pretty big!

The secret service came to search his place, and he ended up pleading guilty to a felony charge of computer hacking, and was prohibited from using a computer for three years. Oh, and he also had to do 720 hours of community service! There's a great video where Samy talks about this experience.

In any case — we've come a long way! These days, instead of being threatened with legal repercussions, friendly hackers are actually rewarded for responsibly disclosing vulnerabilities. We've already talked about crafting your policy/rules page - setting rules of engagement and communicating with hackers about what is (and isn't!) okay. Having clear guidelines like this in place greatly reduces the risk of a "Samy situation," where a hacker inadvertently impacts your production environment. Lastly — when working with legal, you'll want to be sure they have a chance to review HackerOne's terms and conditions.

3.5 PR/Comms

While most organizations start with a private, unannounced bug bounty program, it's still very important to keep your PR and communication teams in the loop. Especially for PR teams that haven't dealt much with information security, there may be a hint of cautiousness when they hear about your initiative. I've sometimes heard sentiments like…

- Isn't having a bug bounty program the same as admitting we can't handle security ourselves? It makes us look weak!

- What if a hacker exposes our data?

PRO TIP:

In reality, *tons* of big name organizations are running public bug bounty programs, and are reaping the benefits of the positive PR associated with it. To get started with your program, click here, give us a few bits of info and we'll be in touch ASAP!
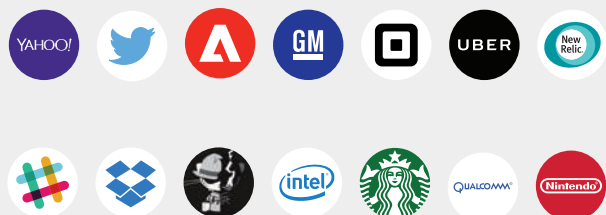
The U.S. Department of Defense's "Hack The Pentagon"
program is another great example. When communicating
with your PR team, you'll want to convey:

- It's impossible to catch everything yourself

- Bug bounty programs let friendly hackers work with
  you to help identify issues before the bad guys do

- Big name companies and organizations (even the Federal
  Government) have bug bounty programs - they are a best practice

- These days, not having a program actually
  puts you behind the curve

Overall, you'll want to ensure all internal stakeholders are aware of
your timeline to launch and are ready to rock. It can be helpful to set
up a high-level presentation advertising the benefits of a bug bounty
initiative at your organization; you can use this as a reference to
demonstrate value and become a bug bounty hero within your org.
Here are a few resources to check out, and perhaps share internally:

- Riot Games' David Rook's talk on Running a Bug Bounty Program
  (~21 minutes)
  David Rook provides some great advice based on his
  experience running Riot Games' bug bounty program.

- 5 reasons not to run a bug bounty program (~25 minutes)
  This is good ammo for addressing common concerns
  that come up around running a bug bounty program.

- Bug Bounty 5 years in
  Great post by Collin Greene based on his experience
  running bug bounty programs at Facebook and Uber

If you need any assistance in convincing internal stakeholders
that a bug bounty program is a good idea, feel free to reach out
to sales@hackerone.com for hands-on advice and guidance.

LAUNCH

The foundation has been set. You have your well-crafted policy, agreed upon SLAs, locked down your budget, and you're ready to push the big red button!

If you're a Professional or Enterprise customer, we also recommend you do one last spot check with your customer success manager at HackerOne. We can help review your policy/rules page, bounty amounts, and ask a few questions to make sure you haven't forgotten anything before launching.

Now the biggest questions to consider as you prepare to launch your program are:

1. Should my program be public or private?

2. How many hackers should I invite?

3. What's my scope?

Chapter 4.1: Start small and work your way up

I can't stress this enough! To quote some wise rhyme-sayers of days past, "low and slow, that is the tempo." I've seen some companies get burned hard by either going public too soon, inviting too many hackers, or having a huge scope straight out the gate. Unless you want to end up like this poor puppy -

it's important to take it easy, at least at first. Here are the key factors you want to consider, as well as our recommended starting points:

- Program type (public or private): Recommendation = Private*

- Number of invited hackers: Recommendation = 5

- Scope: Recommendation = 1 - 2 scopes (e.g. websites, IoT devices, mobile apps)

A note on private programs; there are two major types of private programs. Many organizations start with a private program that is completely unannounced, operating under the radar until they are ready for prime time. Some organizations, like (at the time of this writing) LinkedIn, publicly advertise they have a private program which is invite-only. Operating in this method lets you benefit from the positive PR associated with running a bug bounty program, but still lets you scale your efforts at a reasonable pace.

This might seem small, but launching your program is like cooking a great dish - it's a lot easier to add more spice later than trying to balance things out after dumping a whole bottle in at the start.

In the first day, expect four serious, non-duplicate vulnerability reports. The average customer targets finding ten bugs in the first two weeks — you can target more if you like. Ask about HackerOne's Managed Program if you need help with the validation, reproduction and triage of inbound reports.

Beginning with a much more manageable amount of hackers contributing to your program means you can test your processes.

I liken it to Tony Stark when he's first trying out the Iron Man suit. Remember? He jumped straight to 10% thrust capacity and blew himself into a concrete wall.

Not fun.

He had to learn how to handle the power of the suit before he became a superhero. So he took baby steps, starting from 1%, then 2.5%, and so forth.

The next scene - he's flying Mach 3 over the city of Los Angeles.

Similarly, you can get there too, by throttling the inflow. During your first week, you'll start triaging reports, filing bugs internally, communicating with hackers, the whole nine yards.

Invariably, you will hit a few snags - which is okay! It's best to get the bumps and bruises with just a trickle of reports. We all know things happen and nothing goes perfect. Here are a few possibilities:

- Your only developer who knows how to fix a critical vulnerability is on vacation

- Half of your security team catches the flu and calls out sick

- A hacker reports a vulnerability in a scope you didn't even know you had! (yikes)

Whatever happens, just remember this is to be expected, and it's important to roll with the punches. You'll learn from these experiences and course correct as needed.

PRO TIP:

If you are a Professional or Enterprise customer, remember, your customer success manager is here for you throughout this whole process. If you're uncertain about anything, our team of bug bounty experts can help advise - trust us, we've seen it all.

Chapter 4.2: It's time to start triaging

Alright, reports are coming in hot - are they what you expected? Are hackers hitting the right targets, and finding the types of bugs you want to see? If not, you might need to tweak your policy page, or adjust bounty amounts. Now is the time to ask for and listen to any feedback from hackers participating in your program. Keep in mind that many top tier hackers have been doing this for awhile, and direct feedback on your program policy and how you triage reports in these first few weeks is gold.

Conversely, it's important for you to provide feedback to hackers as well; transparency is paramount! Every security team (yours included) has their own preferences for how bugs are reported, as well as which properties and bug types they care about most. While you've already done a fantastic job writing a crystal clear policy, sometimes hackers make mistakes, and it's important to provide feedback when this happens.

Developing a solid relationship with hackers will keep top talent coming back again and again for more, helping provide ongoing value for your program. For more information on why first impressions are so important, check out our detailed [post on bug bounty first impressions] (https://hackerone.com/blog/bug-bounty-first-impressions).

Did you know that HackerOne offers triage services?

If your team doesn't have the bandwidth to take on the flow of incoming bug reports yourself, HackerOne can help handle all of this for you. Our triage team is composed of technical experts that live and breathe bug bounties... in fact, many are bug bounty hunters themselves! We'll handle all communication with hackers, reproduce reported bugs, and only escalate valid bugs to your team. If you want a fully managed program, we can even help with determining and paying out appropriate bounty amounts. For more information check out https://hackerone.com/product/fully-managed/

Chapter 4.3: Flex and iterate

You'll inevitably run into some outlier cases that might not have been addressed in your policy page. For example, hackers might find and report bugs on properties you didn't even know existed! Maybe a dev spun up an externally-facing instance on Amazon for a pet project, then forgot about it and hadn't updated anything on it for a few years; or perhaps, someone engaged a third party web dev to spin up a website for a company event.

Whatever happens, it's important to be on your toes, and realize you'll need to iterate and update your policy page and processes based on feedback from hackers. As I'm sure you can tell, there's a strong need for fluidity in your policy and processes. If you've done a good job of championing internally, you should have a fair amount of leeway in terms of leading decisions around changes in your scope, policy, and processes.

Chapter 4.4: Pump it up

Once you're in the groove of triaging reports, filing bugs internally, etc., and it seems like volume is steady (or even starting to tamper down), now is the time to turn up the volume.

As you get a handle on things, there are three primary ways to incentivize more participation and increase the volume of reports:

1. Increase the scope of your program

2. Add more hackers to your program

3. Juice the bounty amounts

Now, you may be wondering, which of these should I do first? How much scope should I add? How many hackers? Which bounty amounts do I increase, and by how much?

All great questions, and the answer (you guessed it) is… it depends!

If you want a ton of eyeballs on one or two properties, adding more hackers while your scope is small is a good way to start out. If you're getting lot of lower severity reports, but not enough juicy big bugs (RCE, SQLi, etc.), you might want to increase bounty amounts for those vulnerabilities specifically. If it seems like you've received a ton of reports for your first one or two scopes and improved security quite a bit on them, perhaps the well is dry, and it's time to add a fresh scope to your program.

One strategy is what we at HackerOne call "The Blitz". It was pioneered by Shopify, where they paid out over $350K in one day. Their CEO, Tobi Lutke [explains in a Hacker News thread]:

> So the basic idea is that when we launch something new, we 10x the payouts to bootstrap the process of familiarization. We also provide a very convenient local environment for doing the work in. It should be more fun and more lucrative to make Shopify related discoveries then other companies. After this initial period we then reduce the payouts somewhere slightly above community standards. It's all just business 101.

- TOBI LUTKE, CEO, SHOPIFY

This is a basic marketing investment approach to up the level of awareness and participation by top talent on your program.

Now "The Blitz" strategy may not be right for you, as there are admittedly a ton of factors at play.

When considering what's the best road to take, in general, you'll want to consider the following:

- How many reports are you getting in each of your scopes?

- What types of reports are you getting in each of your scopes?

- Hacker breadth (do you have a lot of hackers actively participating or just a few?)

- Hacker depth (of the hackers that are participating, are they continuously submitting a lot of reports, or is each hacker submitting 1-2 reports then disappearing?)

- How's your budget looking? Do you have flexibility to increase bounty amounts in areas you want more attention on?

- How's your bandwidth looking? Is your team underwater from triaging, or is your bug bounty on duty looking pretty light?

- How's vulnerability management going? Do you have a ton of work to do to fix the bugs that have come in?

The answers to these questions will help guide you towards which knobs to tweak to pump up the volume of reports.

At HackerOne, we have what we call the Hacker Success Index: a series of inputs you can analyze to benchmark against other programs and your past behavior. See the initial post on our blog: Measuring Success in Vulnerability Disclosure.

THE POST-BOUNTY ERA

You've been running your program for a bit, you've ironed out a lot of the obvious kinks in your bug bounty processes... so now what?

It's important not to stagnate. It's easy to fall into the trap of "yeah, the first month or two was awesome, I guess we'll wait and see what else rolls in…" Trust me, I've been there; once report volume tampers down, there's less work to do. Bug bounty on-duty is a breeze! Why wreck a good thing? This is the phase of your program where the BBL (bug bounty leader) can really shine.

Chapter 5.1: Scale your program

More hackers + more scope + increased bounties = bigger, badder bugs which means more value. It might seem counterintuitive, but you want vulnerabilities.

The fact is - any vulnerabilities that are live in production are already there, you just might not know about them yet.

Over time, you'll want to increase your scope to cover just about anything. The key things to consider as you expand your scope are:

- What types of bugs do I want?

- Can I handle the increased load?

As you add sites to your in-scope attack surface (one at a time), you can iterate through the previous processes you went through when first launching your program.

Your bug bounty program should be treated as a dynamic program that will need adjustments and tweaks over time. As you introduce a new scope, illustrate in your security page what sorts of issues you're hoping to find. Consider what you can do to give hackers more information to better aim their efforts at uncovering juicy bugs.

SCOPE PRO TIP:

Uber does an excellent job of this with their Treasure Map. Yelp also has a great page illustrating various scopes and what they're hoping hackers will find.

As you invite more and more hackers, eventually you might even take your program public! When you get to this point, you probably have a crew of great hackers with whom you've developed solid relationships and work with on a regular basis, and you're going to see some fresh talent pour in as well. You might also get a little more noise when you go public - you can tamper this down with signal requirements, and playing with bounty amounts (e.g. lower bounties on low severity issues, increase bounties on top tier bugs).

To whatever extent you haven't already, you'll want to automate as much as possible.

This is tip #7 from Dan Adams at *Sky Betting & Gaming* in his *Hacker Herding* article. Dan says,

"

Templates. Just templates. You're going to need to manage large volumes of tickets and find yourself making the same statements over and over. Save your sanity and as soon as the trends become clear write a few, high quality boilerplate responses for different scenarios. Run them past your teammates, hone them, and use them.

- DAN ADAMS
 WEB APPLICATION SECURITY SPECIALIST AT SKY BETTING & GAMING

The more efficient your bug bounty processes are, the easier it will be to scale and handle a larger load of reports.

Another way you can spice up your program is to take it live. HackerOne has helped companies such as Snapchat, Zenefits, Panasonic Avionics, AirBnB, and Shopify run live hacking events where HackerOne's top brass hackers have flown out to hack these companies on the spot. Live hacking events like this are a great way to develop solid relationships with the bug bounty community and get tons of awesome bugs in a short period of time. Check out our recap video of our live hacking event in Vegas (H1-702)!

Many programs also run time-boxed monetary promotions to incentivize research in a particular area. For example, Github celebrated their 3-year bug bounty anniversary by offering bigger bounties for the most severe bugs found in a 60-day period.

Github ended up getting the **exact** type of bug reports they were looking for. They paid out over $30K in bounties to some amazing hackers and tracked some very detailed stats for their program that were only possible through the HackerOne platform according to Github Security Program Manager, Neil Matatall. Read their full post: Bug Bounty third anniversary wrap-up

And it was a BIG success.
Their Bug Bounty Leader even tweeted:
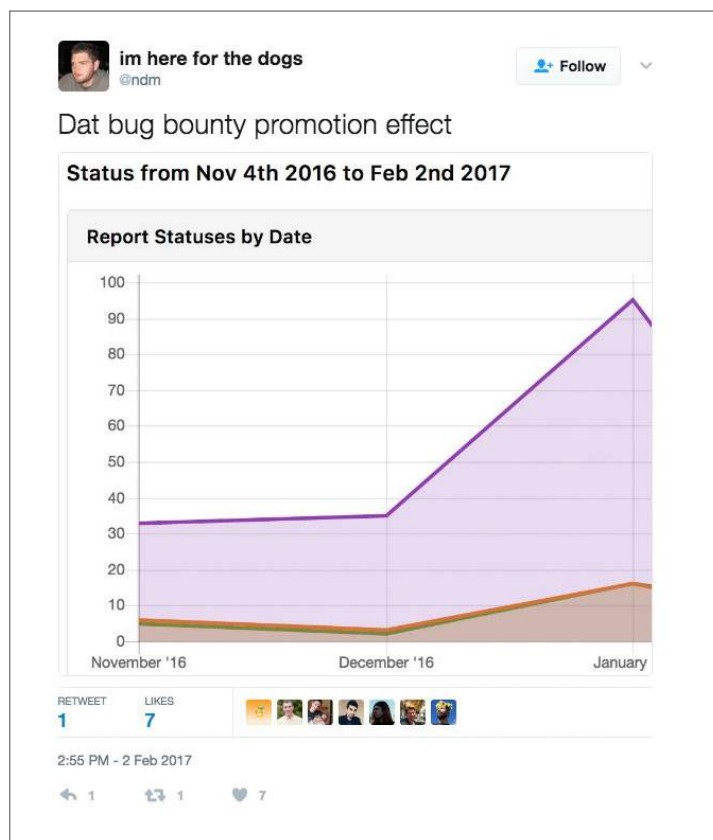
Github is just one example, and there are others. Spend some time as a team to brainstorm fun activities for hackers and that will maximize the value for your program. You can always talk to us at HackerOne as well. We love putting our thinking caps on and cooking up some rad ideas!

Chapter 5.2: Vulnerability Management - Reloaded

Security only improves when bugs are fixed, not when they are found.
As your program scales, so will the influx of bugs that will require solid vulnerability management (VM) practices to ensure you can derive as much value as possible from your program. Ensuring you've ironed out any inefficient kinks in your VM processes will be vital. Even though we've already talked a bit about vulnerability management in terms of:

- Finding owners

- Communicating remediation timelines

- Improving security culture

As your program grows, bugs will start bursting the seams of your VM processes. At this point in your bug bounty life cycle, it's critical more than ever to have rock solid processes in place to handle the ever-increasing number of valuable bugs pouring in. You're not alone in this challenge! Some of the biggest companies in the world struggle with how to do this well.

While there's no silver bullet to vulnerability management, here are some best practices I'd recommend based on my experience:

FIND AN OWNER.

- One of the most common pain points I've felt and heard from others is identifying owners that will take responsibility for fixing security bugs. This can result in huge delays in your remediation timelines.

- Remember that Issues uncovered from your bug bounty program have a hacker on the other end waiting for news as to when the bug will be squashed!

- You'll quickly realize you'll need to identify efficient means of figuring out who owns what. This will vary from organization to organization, but start early on identifying any optimizations or automation to track down owners of various in-scope assets.

AUTOMATE!

- If you've communicated internal SLAs around remediation timelines of bugs of various severities, you can leverage this to automatically ping individuals that are close to (or have already fallen out of) SLA on the bugs they are responsible for fixing.

- You might even want to set more granular SLAs for your bug bounty team, for example on finding an owner:

  - Critical severity bugs should have a defined owner within three to four hours.

  - High = three to four days

  - Medium = one to two weeks

  - Low = one month

- You can then automatically ping yourselves on bugs that don't have owners and are out of SLA to reactively pour in more manual effort to get them moving.

PRIORITIZE YOUR MANUAL EFFORTS BASED ON THE SEVERITY OF THE ISSUE.

- Higher severity bugs will require more frequent, and perhaps more in-depth updates.

- Lower severity bugs may not require as many updates.

SLUGGISH VULNERABILITY MANAGEMENT IS OFTEN A RESULT OF PEOPLE AND PROCESS PROBLEMS.

- Security bugs are often unexpected work. Put yourself in the developer's shoes and try to understand their perspective. What's the first thing you'd do when presented with such a task? You'd probably want to understand what it is, why it matters, and how to prioritize it.

- For this reason, it's crucial that you continue to push your bug bounty hunters to provide exploitability and impact information in their reports, as well as fill in those gaps yourselves as needed. When negotiating with a developer to fix an issue, "why is this important enough for me to set aside time for it?" is a key question that must be answered.

- You'll also want to consider everything else on the affected developer/team's plate. Constant blind pushing for bug bounty issues to be fixed ASAP may result in a "boy who called wolf" situation; if everything's urgent, nothing's urgent! Don't be afraid to ask what else they are working on and where fixing this bug fits in their stack. Maybe you can even help take some other things off their plate!

- Consider each team's processes for accepting work inputs. Are they SCRUM fans? How do you get your security bug into their product backlog? What sprint will this issue land in?

When negotiating with a developer to fix an issue, "why is this important enough for me to set aside time for it?" is a key question that must be answered.

Vulnerability management is a complex topic with multiple books written on the topic, but the key things to remember in the context of a bug bounty program are:

1. There's a friendly hacker waiting for news on remediation of the bug. Don't be afraid to give them occasional updates, especially if you think you're at risk of slipping out of your remediation timeline SLA.

2. Someone outside of your organization discovered this bug. Luckily a friendly hacker brought it to your attention, but this means it's possible that criminals are also aware of (and perhaps already exploiting!) this issue.

In regards to the second point, you can leverage this to negotiate for the prioritization of a fix. You may also want to look into what logs can supplement detection and response mechanisms to determine if the bug is being actively exploited (depending on remediation timeline), or if it's been exploited in the past. Root cause analysis can also help you uncover systemic issues in your software development life cycle (SDLC).

Don't be afraid to give your hackers occasional updates, especially if you're at risk of slipping out of your remediation timeline SLA.

Chapter 5.3: Leverage your bug bounty data - root cause analysis

Bug bounty programs are awesome for sussing out and squashing boatloads of individual vulnerabilities, but some of the greatest value you'll derive from your program is the data that comes out of it.

Bugs that are identified by your program signal towards cracks in your existing security processes and SDLC. This might seem dark and gloomy, but actually, it's pretty amazing! Aggregating this data over time will help you advance from playing whack-a-mole with individual bugs towards proactively identifying and eliminating root causes of systemic issues, resulting in vast, overarching improvements to your security program.

If you're seeing XSS pop up far more often in one of your scopes vs. all others, it's time to sit down with the development team responsible for that area and figure out what's going on. Do they just need more training around security development? Are they not taking advantage of libraries or templates that can help automatically prevent large swathes of XSS?

In addition to chasing down root causes of systemic issues, the data from your program can help you tweak the program itself to achieve better results. How many bugs are you finding, of which vuln types, on which properties? Where is your budget being spent? Perhaps you've spent half your budget (and tons of time) on numerous low severity issues in a handful of properties that don't contain much sensitive data or functionality. Maybe you had already set the reward amount to be fairly low in those areas, but maybe it needs to go lower; this will free up budget to increase reward amounts for the juicier bugs that aren't being found as often.

HackerOne also provides quite a bit of data around your program in your dashboard, such as:

- # of bugs resolved

- # of hackers thanked

- # of reports rewarded

- total bounties paid

- bounty average

- payout % of resolved reports

- response time

- resolution time

You can see these metrics for a set period of time, as well as across the entire duration of your program. Graphs are provided illustrating report statuses by date, response and resolution time by date, bounty payouts by date, and a breakdown of bug closure state. There's also data around the top hackers that have contributed to your program, and your top bounties by amount.

Some of HackerOne's most advanced customers leverage the HackerOne API to connect the dots between their bug bounty program data and any existing internal analytics tooling (e.g. Tableau, Looker, homegrown analytics, SOC dashboards, etc.). How to leverage this data will vary depending on your organizational goals, but the sky is the limit.

HACKERONE CURRENT METRICS SAMPLE VIEW

Current Metrics

Bugs Resolved	10
Hackers Thanked	8
Reports Rewarded	6
Total Bounties	$9,000
Bounty Average	$1,500
Payout % of Resolved Reports	60.0%
Response Time	22 hrs
Resolution Time	15 days

Chapter 5.4 - Diplomacy

In an ideal world, every report will be valid, with clear reproduction steps and a well-defined security impact, and every bounty paid will be perfectly aligned with both your and the hacker's expectations, bugs will be resolved in a timely manner, and you'll respond to every report within a reasonable timeframe.

But we live in the real world. Nothing ever goes *that* swimmingly.

In reality, so many things can go wrong: there isn't enough info to reproduce the report; a hacker feels your bounty amount was too low; your bug bounty on-duty person calls out sick, and no one is triaging reports… you get the idea.

When things go south, it's easy to get frustrated and want to quickly churn through reports; it's also easy to get angry and engage in virtual fisticuffs when someone is being difficult. All of that said, it's important to remember that everything you say is representing your company, and behind every report is an individual who's invested time in trying to help you improve your security.

You can avoid blowups by proactively crafting an awesome security page, and following these tips for a successful launch, but when all else fails, you'll need help from your two best buddies: tact and patience. Here are some common examples of misunderstandings that can crop up, and suggested approaches on how to respond.

The Ransom Note

Let's say you get a report that looks like this:

TIMELINE

somehacker submitted a report to **CompanyA**   Dec 20th

I've found RCE, but I'll only give you the vuln details if you pay me $20k up front, otherwise I'm releasing the details to the media in 24 hours.

Wait what? At first, this can seem pretty scary, but it's often a bluff. The whole point of running a bug bounty program is to provide a way to work together with friendly hackers in a structured manner - this report is the antithesis of that. So how do you respond to this?

A **CompanyA** posted a comment   Dec 20th

Hello **@somehacker,**

Thanks for writing in, and thanks for taking the time to hunt for issues as part our bug bounty program; we appreciate it! We'd love to work with you on how to approach this situation; if you have a valid RCE, we are definitely interesting in knowing how to reproduce it. Our typical process is to accept details of the vulnerability first to see if the issue is reproducible. Once we have this information, we can perform a deeper assessment on the impact of the issue, how exploitable it is, etc. This provides us with data to accurately gauge what the bounty amount should be. For RCE bugs, we typically pay $xx,xxxx - you can see this on our rules page here: <link to your policy/rules page>. Hopefully you can understand our situation - without this information, it's not possible to accurately assess the value of your report. If you could provide info on how to reproduce the issue, we'd really appreciate it. If you have any questions or concerns with this approach though, please let us know.

This response is kind, but firm, explaining why you need the vulnerability details first. If they are bluffing they may come back and huff and puff some more, but it's important to stick to your guns. If you end up paying out for every "threat" you receive, you'll dive into a rabbit hole of unwarranted bounties.

This invalid issue is totally valid!

A lot of misunderstandings can occur around lower severity issues that you may not consider to meet the bar for a bounty, or even miscommunications around what is and isn't in scope. Let's say you get a report like this:

TIMELINE

somehacker submitted a report to CompanyA   Dec 20th

This page - <affected URL> - is susceptible to clickjacking, as you're missing the
X-Frame-Options header. Bounty please!

However, it turns out the page in question doesn't house any sensitive or state-changing functionality, e.g. it's a marketing website with only static content. Where's the security impact? A good response would be:

A   CompanyA posted a comment   Dec 20th

Hello @somehacker,

Thanks for the head's up! We took a look at this page and determined that it only houses static content, so there doesn't seem to be much of a security impact here. As such, we're going to close this report as Informative. If you think we've missed something and can demonstrate a security impact in this scenario, please let us know!

If the hacker still pushes on it, e.g. "But XFO is a best practice! You need to fix this and pay me!" don't be afraid to be kind yet firm:

A   CompanyA posted a comment   Dec 20th

Yes, we agree that the header is a best practice, and we may consider adding it here, but the impact of doing so is negligible, as there isn't anything sensitive to clickjack. Since there is no security impact, this does not meet the bar for a bounty. When looking for clickjacking bugs, please try to identify the exploit scenario and assess security impact - pages that house state-changing functionality are more likely to qualify. Thanks, and good luck on your future bug hunting!

Overall, it's important to keep your cool and navigate through the situation as calmly as possible. If all else fails and you can't see eye to eye, you can always request mediation from HackerOne to help mediate the situation.

Riot Games celebrated $1,000,000 in bounties with this awesome cake!

Chapter 5.5: Celebrate the milestones

One last piece of advice is to celebrate your program's key hurdles. Highlight the analysis of your results like Yelp did in their blog "First 100 Days of Yelp's Public Bug Bounty Program".

Showcase your team's process and journey like Joakim Tauren did for Visma Oy in a series of blog posts, or wait for your HackerOne customer success representative to email you a congrats on crossing 500 vulnerabilities fixed. Software development is a journey, and often a long and arduous one. Make sure to pause and recognize your efforts in security and the great work your team is doing to keep your customers safe.

Wrapping it up

Phew, what a read! Thanks for joining me - as time goes on, we'll add more and more resources to help you on your bug bounty journey. We'd love your feedback - do you have any tips or tricks for running your program that aren't listed here? Let us know! Feel free to hit me up on Twitter with any of your thoughts, comments, or questions.

— *Adam Bacchus*

APPENDIX

1. Bug Bounty Readiness Assessment Questionnaire

2. Bug Bounty Leader Job Description

3. Links and Resources by Chapter

4. Glossary

Bug Bounty Readiness Assessment Questionnaire

Engaging with the global community of independent security researchers can result in significant risk-reduction for product and information security teams, at dramatically lower costs compared to traditional penetration testing.

When it comes to working with hackers, there are a few different options. This assessment questionnaire will help you gauge which is best for you. What are these options, you ask?

One of the biggest distinctions is whether or not you offer monetary rewards, or "bounties," to hackers outside of your organization that report valid security flaws to you. Many organizations start off without bounties, with what's called a **Vulnerability Disclosure Program**. In a **Bug Bounty Program**, the stakes are a bit higher, as you offer varying monetary rewards for issues identified and reported to you. Another factor is time; some choose to start with a pilot program to test the waters, which can last anywhere from a month to a year.

This assessment questionnaire is designed to help security stakeholders assess their particular program's needs and begin formulating a plan for inviting ethical hackers to test their attack surfaces.

Note: *there is no one-size-fits-all solution, so beyond this document, HackerOne provides expert consulting with our seasoned Customer Success team - during initial setup, frequent check-ins during initial post-launch phase, and (at minimum) ongoing quarterly reviews.*

3 Core Elements to a Successful Bug Bounty Program

PLATFORM:
Technology tools to receive, organize, analyze, understand and offer rewards for vulnerability reports from third-party researchers.

COMMUNITY:
The group (private or public) of security researchers (aka hackers) who are able to submit reports and are eligible for rewards.

ENGAGEMENT TEAM:
The people communicating with security researchers, validating issues, determining appropriate rewards, and escalating to engineers for remediation.

COMMUNITY OF HACKERS

PLATFORM

Central Repository for Vulnerabilities

Deep Analytics

Best Practices Automation

Best Practices Workflow

Seamless Bounty Payments

Native Integrations

Advanced Hackers' Reputation Management

Intelligent Duplicates Detection

Convenient Triage and Bounty Management Services

ENGAGEMENT TEAM
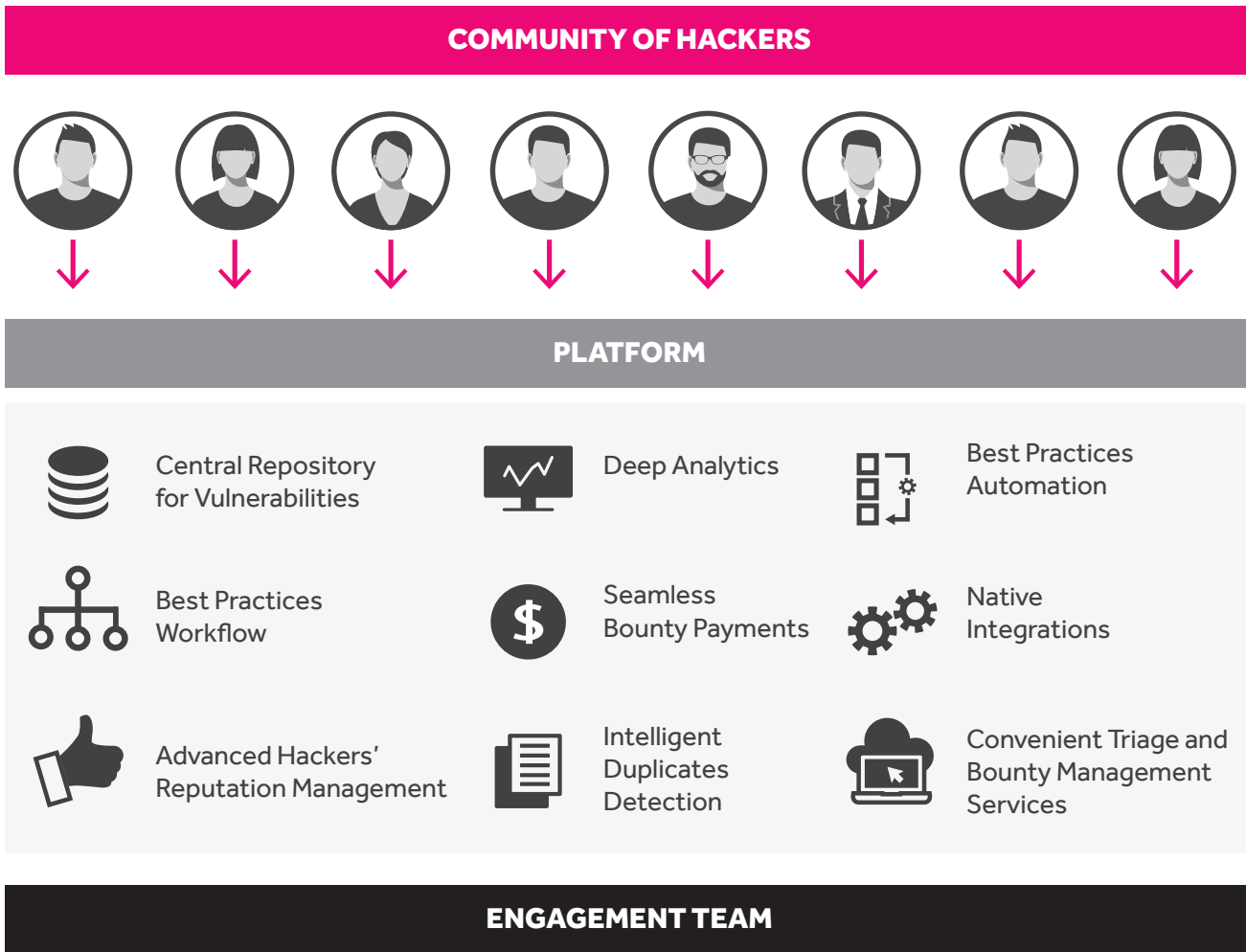
Questions to Assess Program Requirements

Answer the following questions and take note of which answers you picked. Then proceed to the next chapter to draw conclusions. We'll be happy to talk through the results with you and build a custom plan for your organization.

What kind of testing have you performed on your systems in the past? How often?

1. Very little or only internal audits

2. Automated vulnerability scanners checking for known issues

3. Occasional contracted penetration test

4. Regular contracted penetration test (quarterly or more)

5. None of the above

What does your security team look like today?

1. No dedicated security staff, 1–2 engineers focus on security issues.

2. 1–2 dedicated security staff, wear all the security hats

3. Structured team of security engineers, 1–2 security-focused executives

4. Several security teams across multiple product areas or acquisitions, each with independent leadership

5. None of the above

What do your attack surfaces look like?

1. Single web app or mobile app. Relatively simple with no advance credentials needed for testing.

2. A few web applications, light API connectivity with third-party vendors or services. Basic account credentials needed for testing.

3. Complex array of overlapping applications, services, and users. User credentials can be difficult to provision.

4. Combination of software and hardware, web of API structures, several third-party vendors integrated into solution. Physical devices need to be tested.

5. None of the above

Have you received vulnerability reports from the outside before?

1. No, or just 1 or 2 and we aren't sure how to respond.

2. Yes, we've received a few over the years, with mixed quality and frequency.

3. Yes, we receive a steady flow of reports from researchers and have a process for handling them in place.

4. Yes, we have a global response team that receives all reports and routes them to different product areas.

5. None of the above

What kind of internal communication/productivity tools does your team use today?

1. Email, spreadsheets

2. Common issue/ticket tracking software
   (JIRA, ServiceNow, ZenDesk, FreshDesk, etc)

3. Custom-built issue/ticket tracking and internal comms software

4. We have multiple teams who use different
   tools, depending on the team

5. None of the above

Answered ALL 1's: Email Forwarding

Security@ (Vulnerability Disclosure Process)

Your first step is to create a robust and agreed upon process for receiving and fixing bugs from researchers. At this stage, you're not ready to solicit hackers for additional bugs through a paid bounty program.

COMMUNITY:

White hat hackers have likely already submitted some vulnerabilities to you - did you know about them and have a process to respond and address their report? Responsiveness is consistently ranked as the top ask by ethical hackers - having a process is step 1 in addressing inbound reports and building relationships with the hacker community.

PLATFORM:

The out of the box platform functionality with email forwarding from your company's security@ inbox will address your immediate needs. Once the basic workflow to receive reports and communicate with researchers in a timely fashion is established, leverage HackerOne to supercharge your email-powered security@ with efficient handling of incoming vulnerability reports at scale.

Benefits of Security@ Email Forwarding / VDP:

- Email forwarding allows you to see the reputation, signal, and impact of submitters from the largest hacker community in the world (more signal, less noise). If the hacker submitting a vulnerability to you is on HackerOne, you'll get immediate signal and information on any prior submissions they may have had.
- Compliance with the ISO 29147 vulnerability reporting standard ensures your team follows best practices.
- Tap into the power of the platform. HackerOne's advanced features such as triggers, common responses, duplicate detection, report classification, signal requirements, a bi-directional API and much more allow you to work far more effectively compared to a legacy email solution.
- Less email spam. A registration process to complete the submission stops unsolicited bulk email spam in its tracks.

SERVICE TEAM:

Our customer success and support team are available to address your questions at all times: support@hackerone.com.

Answered mostly with 1's:

Security@ With Bug Bounties (Basic Do It Yourself Inbox)

With very little testing, there is likely to be significant "low-hanging-fruit" early in a bug bounty program phase. In many cases, it is recommended to perform automated vulnerability scanning at a minimum before offering a financially incentivized bounty program to a wide group of researchers.

COMMUNITY:

If you are in this phase and have already received unsolicited reports from external researchers, it can be effective to only invite those researchers to your private program, as you'll want to eliminate as much low-hanging-fruit as possible with automated scanning before inviting additional participants.

PLATFORM:

Without incentivizing researchers and keeping the pool of invitees very low, basic platform functionality should do the trick. Establish a basic workflow to receive reports and communicate with researchers in a timely fashion. Keep track of each report's state: New, Triaged, Pending Bounty, Closed. Reports can be closed as Resolved, Duplicate, Informative, Not Applicable, or Spam. With low volume, advanced data analysis will not be especially valuable, so whatever your program data comes with "out of the box" should suffice.

SERVICE TEAM:

Internal Security team members (or DevOps/QA-focused engineers) should monitor HackerOne for new reports and respond in a timely fashion. There will likely be a single bounty decision-maker on your team, should financial rewards be offered.

Answered mostly with 2's:

Professional (Built-in Best Practices)

With scanners eliminating some of the low-hanging-fruit, a financially incentivized program is more cost-effective. Adding external testing via a Bug Bounty Program can apply human creativity that automation cannot simulate effectively.

COMMUNITY:

With a small group of researchers (10-50), invited privately to participate in your program, you can begin building relationships with the researcher community. There will likely still be low-hanging-fruit, so researchers may be excited to have a "first crack" at a new program to earn bounties.

PLATFORM:

Inviting additional researchers to your program will increase the volume of reports. Advanced tools like automatic duplicate detection become important, as multiple researchers may uncover the same issue. Tying into existing organization workflows requires out-of-the-box integrations with popular issue tracking and team collaboration systems. Accessing program data via API enables flexibility for programs to scale.

SERVICE TEAM:

Your security-focused team likely has significant amount of competing responsibilities. Discuss with your team who can take be in charge of triage duty, in a similar way you would assign performance/operations/infrastructure duty. Establish SLAs that you can commit to, and know that program launch can bring the highest volume of reports. It's good to schedule launch during a time when security-focused team can devote significant time to validating and escalating issues. At this level, teams may choose to outsource triage management to gain efficiency of scale and maintain response time SLAs while growing. HackerOne Managed Services may be the right choice when internal security team time is best spent fixing pre-validated issues.

Answered mostly with 3's:

Enterprise (Customization That Scales)

Organizations with established security practices, a security-conscious SDLC, and critical customer data to protect use their Bug Bounty Program as a community-sourced safety net. Many companies at this stage share the goal of leveraging their bug bounty program as a competitive differentiator.

COMMUNITY:

Starting with a small group of researchers, often security teams of this size look to launch a public version of their program over time. Organizations with an existing security disclosure policy look to identify top performers to incentivize and coordinate with private incentivized programs. Specialized researchers will provide the best results when attack surfaces are varied or complex. Invite hackers with demonstrated success in areas like Web Apps, Mobile Apps, API, and IoT (connected devices). Some organizations will also create customized reward structures to keep top researchers engaged over time.

PLATFORM:

With enterprise-level report volume, building a seamless workflow that fits your organization is key for long-term success. Key stakeholders within the organization may require specific filtering and reporting capabilities. Take the data you need using API connectivity, and work with HackerOne's business intelligence team to further customize reporting dashboards. Create an on-brand experience for your researcher community by branding your security page.

SERVICE TEAM:

With a high-volume program, building a team of internal triage experts can have significant challenges. Work to create a duty schedule that will meet SLA requirements. Customize your workflow to include views that show reports violating SLAs. Create customized groups to ensure reports are making it to the right product owners. At this level, teams may choose to outsource triage management to gain efficiency of scale and maintain response time SLAs while growing. HackerOne Managed Services may be the right choice when internal security team time is best spent fixing pre-validated issues.

Answered mostly with 4s: Multiple Programs

Large organizations with multiple product areas, acquisitions, and distinct security teams may require separate bug bounty or vulnerability disclosure programs. When business units function separately, keeping vulnerability report data separate can reduce overhead, keep security teams focused, and minimize risk of sensitive data leaks.

COMMUNITY:

Curate a separate pool of researchers for each program. Organization leaders may look to individual program managers to help identify top performers and ensure those elite researchers are invited to additional programs. Consider each program's scaling strategy separate, but work with a bug bounty expert to look at organization-wide goals and help build a community engagement plan.

PLATFORM:

Global organization leaders should be invited to each program, with appropriate permissions configured. Ensure you can easily switch between program inboxes without logging in and out. View program-level reporting capabilities and ensure there is support for organization-wide analytics as well. Consider managing a global organization-wide inbox that can be the front door for any issues with unclear owners.

SERVICE TEAM:

Treat each program's triage process independently, but consider establishing global SLAs for the metrics easiest to control. For example, "time to initial response" could be an organization-wide SLA, but "time to resolution" may vary widely across different programs. Consider implementing overflow procedures, in which one triage team gets flooded with reports and can "borrow" resources from another. This kind of flexibility and advance planning can ensure an organization meets goals without ballooning headcount. At this level, teams may choose to outsource triage management to gain efficiency of scale and maintain response time SLAs while growing. HackerOne Managed Services may be the right choice when internal security team time is best spent fixing pre-validated issues.

Answered mostly with 5s:

Talk to us! Please contact sales@hackerone.com. We like to think that we've encountered just about every bug bounty situation in the book, but we are also hungry to help unique organizations solve unique challenges.
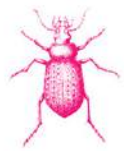
Bug Bounty Leader Job Description

Every great bug bounty program has one person who is ultimately responsible for its success. Even if you're already running a program and haven't consciously assigned this role, chances are there's one person on your team that (officially or otherwise) "owns" responsibility for your program. If not, you definitely need one! Having a single person who leads your program ensures smooth operation of your program by identifying and delivering on measurable indicators of success.

That said, finding the right person for this job can be difficult! You need someone with that perfect blend of technical aptitude, as well as solid communication and project management skills. You need someone who's just as passionate about working with friendly hackers as they are at working with internal teams to on vulnerability management and improving your security program. This could either be someone already working at your organization, or you might want to hire someone specifically for this role. Alternatively, HackerOne's Fully Managed offering can help provide staff to meet these needs for you and your team.

To make it easier to find such an individual, please check out the job description on the next page - feel free to use and adjust this template to your hiring needs!

<ORGANIZATION> is looking for a **<JOBTITLE; e.g. "Technical Program Manager">** to lead our bug bounty efforts.

<ORGANIZATION DESCRIPTION>

The ideal candidate will be a self-starter, a problem solver, organized, a great communicator, empathetic, and have a strong desire to drive for success.

Responsibilities

As **<ORGANIZATION>'s** bug bounty leader, you will be responsible for ensuring the success of our program. You will leverage deep and practical knowledge in security and project management to ensure efficient and effective operations of **<ORGANIZATION>'s** bug bounty program.

Your responsibilities would include:

- Identifying, measuring, and executing on indicators of success for **<ORGANIZATION>'s** bug bounty program

- Ensuring operational success of the program, including leading a team to execute on key deliverables

- Ensure **<ORGANIZATION>** consistently meets SLAs such as…

- response time to hackers on reported vulnerabilities

- time to bounty after confirmation of the issue

- time to remediation of reported vulnerabilities

- Identifying and expanding the scope of the program

- Accepting and incorporating feedback from (and developing a healthy relationship with) the bug bounty community

- Leading vulnerability management efforts on issues identified via the program

- Analyzing data from program to identify and improve upon issues in **<ORGANIZATION>'s** SDLC

- Identifying and calibrating budget for the program

- Working with executive leadership and other parts of **<ORGANIZATION>** to report on results of program and ensure continued buy-in

Requirements

- 2 - 5 years of application security experience, understand security fundamentals and common vulnerabilities (e.g. OWASP Top Ten)

- 2+ years of security consulting experience

- Outstanding communicator with empathy for hackers to strike the right balance. You need to be an advocate for friendly hackers, but also appropriately influence and push back when needed to help hackers be successful.

- Ability to take feedback from hackers and translate to action items for our bug bounty team

- Extremely organized with strong project management experience

- Detail oriented, results driven, fast learner

- A strong sense of urgency and bias for action

- A passion for solving problems, both for hackers and internal teams at **<ORGANIZATION>**

- A great team player

Ideal candidate will meet several of the following:

- Vulnerability assessment experience

- Penetration testing and code review

- Additional experience in IT, security engineering, system and network security, authentication and security protocols, and applied cryptography

- Scripting/programming skills (e.g., Python, Ruby, Java, JS, etc.)

- Network and web-related protocol knowledge (e.g., TCP/IP, UDP, IPSEC, HTTP, HTTPS, routing protocols)

- Federal and industry regulations understanding (e.g., PCI, SOX, GLBA, ISO 17799, HIPAA, CA1386)

- CISSP, OSCP/E, GWAPT, GPEN, GXPN certifications are helpful, but not a necessity

How to Apply

To apply, **<instructions on how to apply at ORGANIZATION>**.

Links and Resources by Chapter

CHAPTER 1

Assessment questionnaire
Answer these questions to determine your bug bounty readiness level.

CHAPTER 2

[CRITICAL!!] Introducing Severity (CVSS)
HackerOne blog post describing Severity and integration
with common bug tracking systems

Bug Bounty Leader Job Description
Template resource for a job description of the BBL.

See what HackerOne customers have to say
Link to HackerOne product page with testimonial information

Anatomy Of A Bug Bounty Budget
HackerOne CFO Ning Wang dives into bug bounty budgeting

Shopify CEO's response to why they paid $368,800 in one day
Hacker News thread

Celebrating Alongside Yelp: Reaching The 100 Day Milestone of Their Public Bug
Bounty Program HackerOne blog post on Yelp's public bug bounty milestone (good
example of mature bug bounty program leveraging private then going public)

How much is a bug worth? Introducing Bounty Statistics
HackerOne blog about how our product automatically shows you
the median bounty across our platform for that severity, as well as
what programs at a competitive and top level are paying out.

Twitter's HackerOne Security Page
See how Twitter splits their scopes into "core" properties and "all other," then outlines
what typical bounties look like for different classes of bugs based on their impact.

Ten Lessons Learned from Launching a Bug Bounty Program

Dan Adams, Web Application Security Specialist at Sky Betting & Gaming based in the UK, goes through his 10 tips on taking your first steps towards operating your own successful bug bounty program.

Bug Bounty or Bust! Crafting Your Security Page

Deep dive into how to craft your bug bounty security page on HackerOne.

Coordinated Vulnerability Disclosure, "Early Stage" Template and Discussion

NTIA working group white paper and research data

CHAPTER 3

How does HackerOne pay the hacker?

Description of how we pay hackers.

Greatest Moments in Hacking History: Samy Kamkar Takes Down Myspace

Hacker Samy Kamkar talks about finding the infamous bug that changed his life.

HackerOne Terms and Conditions

T&C's. What else is there to say? :)

Hack the Pentagon

Blog reviewing the ground-breaking #hackthepentagon program

Riot Games' David Rook's talk on Running a Bug Bounty Program

David Rook provides some great advice based on his experience running Riot Games' bug bounty program.

5 reasons not to run a bug bounty program

This is good ammo for addressing common concerns that come up around running a bug bounty program.

Bug Bounty 5 years in

Great post by Collin Greene based on his experience running bug bounty programs at Facebook and Uber

CHAPTER 4

Measuring Success in Vulnerability Disclosure

Description of the Hacker Success Index: a series of inputs you
can analyze to benchmark against other programs and your
past behavior only available through HackerOne.

CHAPTER 5

Uber Engineering Bug Bounty

The Treasure Map: Uber's innovative treasure map program description

Improving Public Bug Bounty Programs with Signal Requirements

One of the HackerOne features that has huge impact for programs and hackers
alike: signal requirements. This article explains what it is and why it's important.

h1-702 Las Vegas Hackathon

Recap video of our live hacking event in Vegas

Bug Bounty anniversary promotion

bigger bounties in January and February: The GitHub Bug Bounty Program
celebrate their 3-year anniversary by offering bigger bounties for the most
severe bugs found in January and February. Was a big success.

Bug Bounty First Impressions

 Tips for a successful launch of your bug bounty program.

Hello World - Hack us!

Blog by HackerOne customer Visma Oy about their program launch.
Great example of how to practice disclosure and transparency
and maximize value of your bug bounty program.

Glossary

Bounty

A financial reward offered in exchange for a vulnerability report.

Bounty Table

A Bounty Table illustrates how much an organization is willing to pay for various bugs, helps set expectations for hackers, and gives the bug bounty team a guideline to ensure fair and consistent reward amounts.

Bug Bounty Program

A Bug Bounty offers monetary incentives for vulnerabilities and invites submissions from hackers.

Bug Bounty Duty (BBD)

Bug Bounty Duty is a defined period of time (usually one week) where a member of the bug bounty team is responsible for all operational and strategic work on their bug bounty program.

Bug Bounty Lead (BBL)

The bug bounty lead (BBL) is the ultimate owner and champion for a bug bounty program. The BBL is responsible for the success of the program, and helps lead their bug bounty team.

Bug Bounty Team (BBT)

The bug bounty team (BBT) is the group responsible for supporting a bug bounty program. The BBT is responsible for all operational and strategic duties to ensure their bug bounty program is run successfully. These duties can include responding to bug reports, determining bounty amounts, vulnerability management, and building relationships with the hacking community.

Common Response

A saved response, or template, that can be applied repeatedly to Reports.

Criminal

An individual that breaks the law or maliciously exploits a vulnerability.

CVSS	Common Vulnerability Scoring System (CVSS) is the framework we utilize to assign a Severity rating to a vulnerability. Currently leveraging v3.0: https://www.first.org/cvss/calculator/3.0
CWE	Common Weakness Enumeration is the framework we utilize to assign a Weakness to a vulnerability
Directory	The HackerOne Directory is a community-curated resource for contacting an organization regarding a security vulnerability.
Finder	Individual or organization that identifies a potential vulnerability in a product or online service
Hacker	One who enjoys the intellectual challenge of creatively overcoming limitations.
HackerOne	HackerOne is the world's leading vulnerability disclosure platform.
Hacktivity	Hacktivity displays public vulnerability coordination activity occuring on HackerOne.
Impact	Average reputation gained per bounty
Integration	An external integration to HackerOne, such as Slack, JIRA, and Bugzilla.
ISO 29147	An international standard describing vulnerability coordination
ISO 30111	An international standard describing vulnerability handling processes

Report	A Report is a submission from a Hacker that describes a potential security vulnerability.
Reputation	A Hacker's Reputation reflects their historic performance on the platform.
Scope	Structured data representing the attack surface which is included (or explicity excluded) as part of an organization's vulnerability disclosure or bug bounty program.
Security Page	Your Security Page communicates your vulnerability coordination policy to hackers. Also known as a policy or rules page.
Severity	A qualitative rating measuring the severity of a security vulnerability. Can be: None, Low, Medium, High, Critical
Signal	Average reputation gained per report
Vulnerability	Weakness of software, hardware, or online service that can be exploited.
Vulnerability Disclosure	The process by which an organization receives and disseminates information about vulnerabilities in their products or online services. ISO 29147 Definition: 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. It encompasses actions such as reporting, coordinating, and publishing information about a vulnerability and its resolution.
Vulnerability Disclosure Program	A program supporting an organization's Vulnerability Disclosure process.

hackerone

slack

GM

Twitter

Starbucks

Dropbox

Nintendo

Adobe

UBER

salesforce

Snapchat

Lufthansa

intel

Square

New Relic.

airbnb

lyst

GitHub

yelp

ZENEFITS

Booking.com

LendingClub

LOCALTAPIOLA

QUALCOMM

HACK THE ARMY

HARVEST

coinbase

YAHOO!

shopify

imgur

instacart

okcupid

udemy

vimeo

Quora

dr chrono

coursera

CodeIgniter

AUTOMATTIC

Trello

Gratipay