hackerone

Next-Gen Application Security

Launch Effective Agile Security for Agile Development

Improve your application security by following these words of advice on how to incorporate bug bounties and crowdsourced pen tests into your DevOps pipeline.

Many companies are trying to keep up with DevOps practices while keeping their applications secure. It's a tall order to say the least.

Bug bounties and crowdsourced penetration tests allow you to continually test your applications for security bugs. It lets you tap into the minds of many expert researchers to reduce the risk of a breach and help you make more secure software. But before we dive into how to get started using this tool, let's discuss why security and agile development can seem to be at odds.

Challenges of Securing DevOps/Continuous Delivery Environments

Speed is the name of the game. In today's fast-paced world, developers are under pressure to deliver new features as quickly as possible. When you look at the popular terms of the day, such as continuous integration, continuous delivery, and continuous deployment, you see the common thread of ready-to-deploy software. There are "pipelines" pumping out code, sometimes on a daily basis.

Continuous delivery environments make use of exciting new tools and types of software. Containers allow applications to be delivered in pieces, with small, lightweight packages of software that are easily deployed. Unfortunately, the speed of innovation can be dangerous. Using new tools before their security implications are fully understood can lead an organization down a dangerous path.

Securing software at the speed of delivery is a challenge. Traditional security practices bog down development, frustrating development teams and causing unnecessary friction. Securing software in the world of DevOps requires proactive, not reactive, measures.

One such proactive security tool is hacker-powered security. Hacker-powered security refers to any technique that utilizes the external hacker community to find unknown security vulnerabilities and reduce cyber risk. Common examples include private bug bounty programs, public bug bounty programs, time-bound bug bounty programs and vulnerability disclosure policies. The benefits of bug bounties are many, but how do you get started with bug bounties and where do they fit in a DevOps workflow?

How to Effectively Use Hacker-Powered Security in Your Secure SDLC

Humans make mistakes. Humans write code. Even our most robust scanning and vulnerability management processes are proven to miss things, even big things. There is no such thing as perfectly secure software, and the juiciest bugs require creative, intellectual humans to uncover them.

This section will review how you can leverage the power of the diverse hacker community to uncover critical vulnerabilities before they can be exploited.

We'll start with vulnerability management best practices, take a look at the Agile and DevOps workflow, and wrap up with how to deploy a bug bounty program, fully integrated into your security program for maximum effectiveness.

Hacker-Powered Security Step One: Good Vulnerability Management Practices

Vulnerabilities will always exist in all but the most basic software. Today's world features complex applications with many moving parts. Your software will have vulnerabilities.

Since we know vulnerabilities will occur, it's essential to have a system in place to properly handle them. Vulnerability management encompasses all of the activities and workflows triggered by the discovery of a vulnerability.

What does a mature vulnerability management workflow look like? Once a vulnerability is reported, a triage takes place where the vulnerability is verified by the security team. Verification by a human gives you confidence that the vulnerability is real and possibly exploitable by an attacker.

Once verified, a prioritization process is essential. You may not be able to immediately remediate every vulnerability, but try your best not to let them hang around too long.

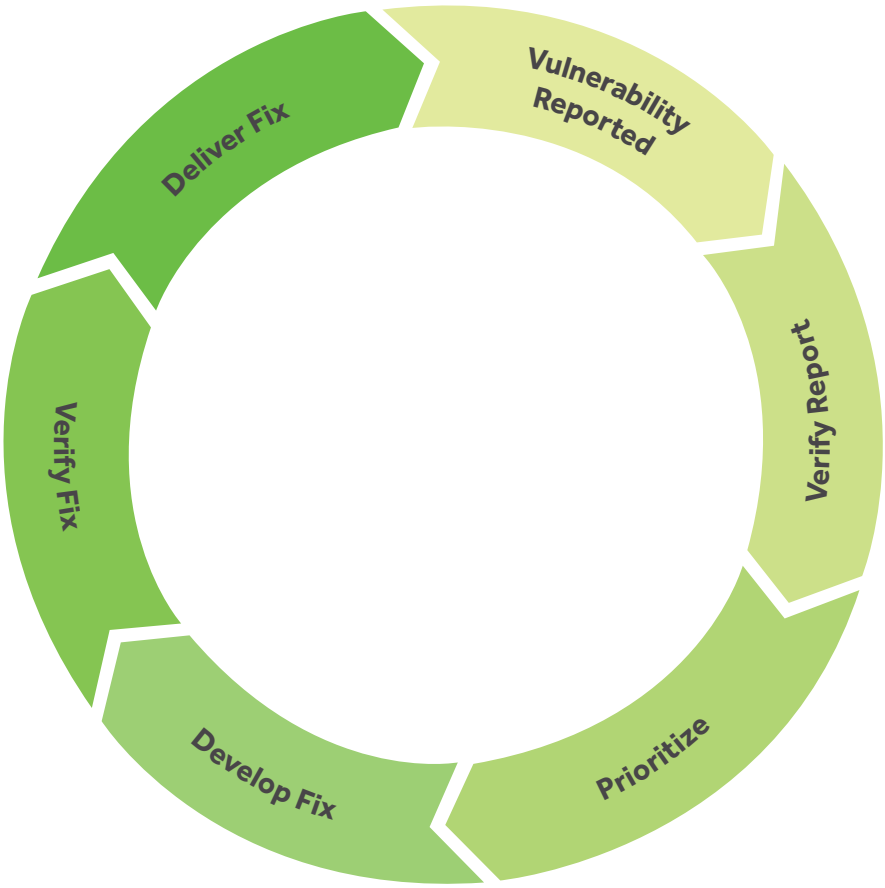
GETTING YOUR DEVELOPMENT TEAMS ON BOARD WITH SLAS

What timelines make sense when deciding levels of priority? Priority can often be in the eye of the beholder and each organization will need to decide what makes sense for it. In general, you should balance your risk tolerance with the ability of the development team to make the required fixes. A legacy application with longer release cycles may not be able to create fixes in only 3 days, so be realistic. If your infrastructure is built on technologies that enable fast delivery, then fix vulnerabilities as soon as possible. Every day a vulnerability goes without a fix is another day an attacker has to find it and exploit it.

Once you decide the timetables for fixing, clearly communicate these Service Level Agreements (SLAs) to the product teams and hold them accountable for them. The time should start once the correct product team has been officially notified of the vulnerability.

Example Fix SLA's

Critical: 1 - 7 Days
Medium: 8 - 30 Days
Low: 31 - 45 Days



A mature vulnerability management process is essential to giving your bug bounty program a good start.

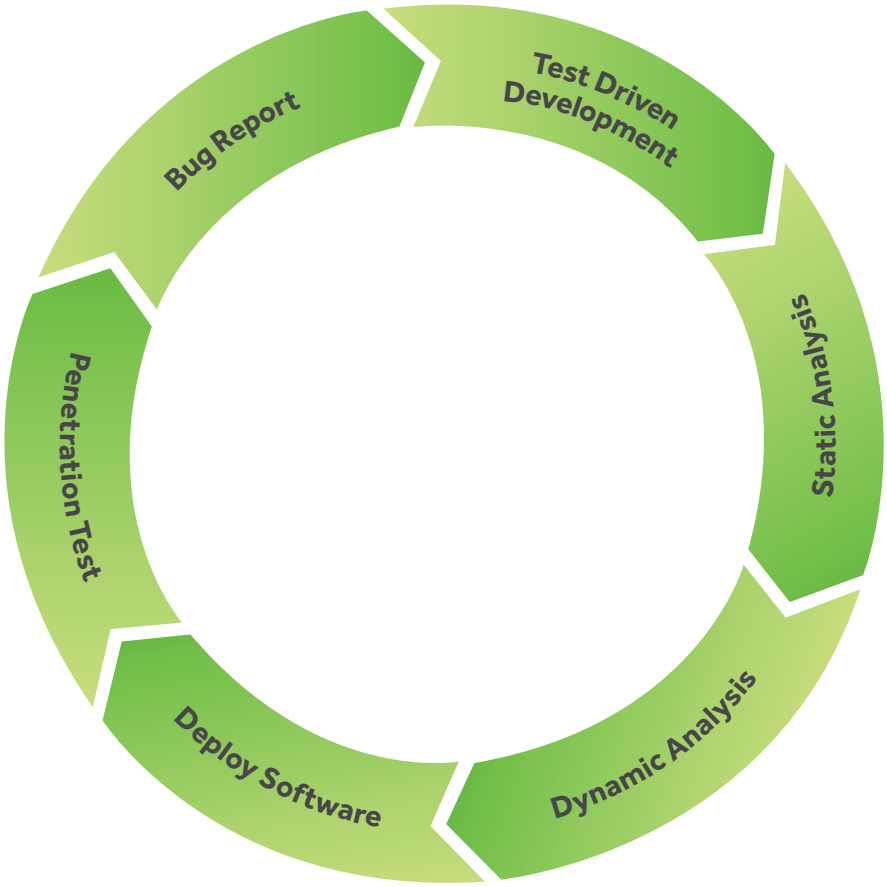
Once a vulnerability is prioritized, the appropriate product team is notified and the time begins to tick on the SLA for the risk level of the vulnerability. The notification usually takes the form of a ticket in a bug tracking system such as Jira. The development team creates a fix under the guidance of a security champion or security SME provided for the development team.

Once the fix is complete, a round of testing ensues to make sure the vulnerability is no longer present. This usually occurs in a testing environment. Then the fix is delivered into production and everyone can feel a little safer.

As we will see, having a vulnerability management process in place will make it much easier to integrate a bug bounty into your overall software development lifecycle. Let's take a look at that piece of the puzzle next.

Hacker-Powered Security Step Two: The Agile and DevOps Workflow

Let's take a look at the typical workflow for an agile or DevOps development team from the lens of security testing. Once we see what tools development teams have used in the past, we can begin to see where bug bounty programs fit into the development lifecycle to create more secure software.



The DevOps workflow depends largely on several passes of automation to help find and fix security vulnerabilities. Can manual tests be incorporated more effectively?

The software development industry has continued to adopt new tools and techniques to help prevent vulnerabilities and find them before they reach production. Developer training is usually the first step taken. However, training developers can be quite expensive, both in money and time. Therefore, it tends to happen infrequently and covers only the basics of software security,

Test Driven Development (TDD) has emerged to help developers test their code as it's being written. The focus is usually on functional testing, with little emphasis on security. Some basic security needs can be tested for using TDD, but the ability to do so is limited.

Static analysis takes over after a developer checks code into a code repository. Scanners comb through the source code looking for patterns that may lead to vulnerable software. Unfortunately, these scanners tend to find many false positives, and time is again taken away to validate what is found.

Dynamic analysis runs a series of complex tests against software running in a test environment. The idea is to try to emulate what a human attacker would do to gain illicit access to a system. Dynamic analysis can find issues that static analysis cannot. However, it can be difficult and time consuming to find the root cause and impact of the problems found.

Don't assume that these analysis programs are useless. They have a place in the DevSecOps workflow. In fact, about 40% of vulnerabilities can be detected using automation. Therefore, it makes sense to eliminate the "low-hanging fruit" of vulnerabilities using these tools. But more is needed.

Once an application is in a usable state, or even deployed to production, a penetration test is run to find more complicated issues that require a human touch. These are important to help find the 60% of vulnerabilities that can't be found using automation. Penetration tests tend to take much longer to perform and require a lot of work to prepare for.

This begs the question: Is there a way to incorporate the human intuition and value of penetration tests with the continuous coverage provided by automated tools?

Hacker-powered Security: How to Incorporate Bug Bounty

Hacker-powered security brings the power of human hackers to the DevSecOps workflow.

Hacker-powered security means having external ethical hackers always testing your applications against the latest vulnerabilities. It takes time for new exploits and techniques to be incorporated into automated tools. Humans can learn the details of a new attack quickly and use it to test your software right away.

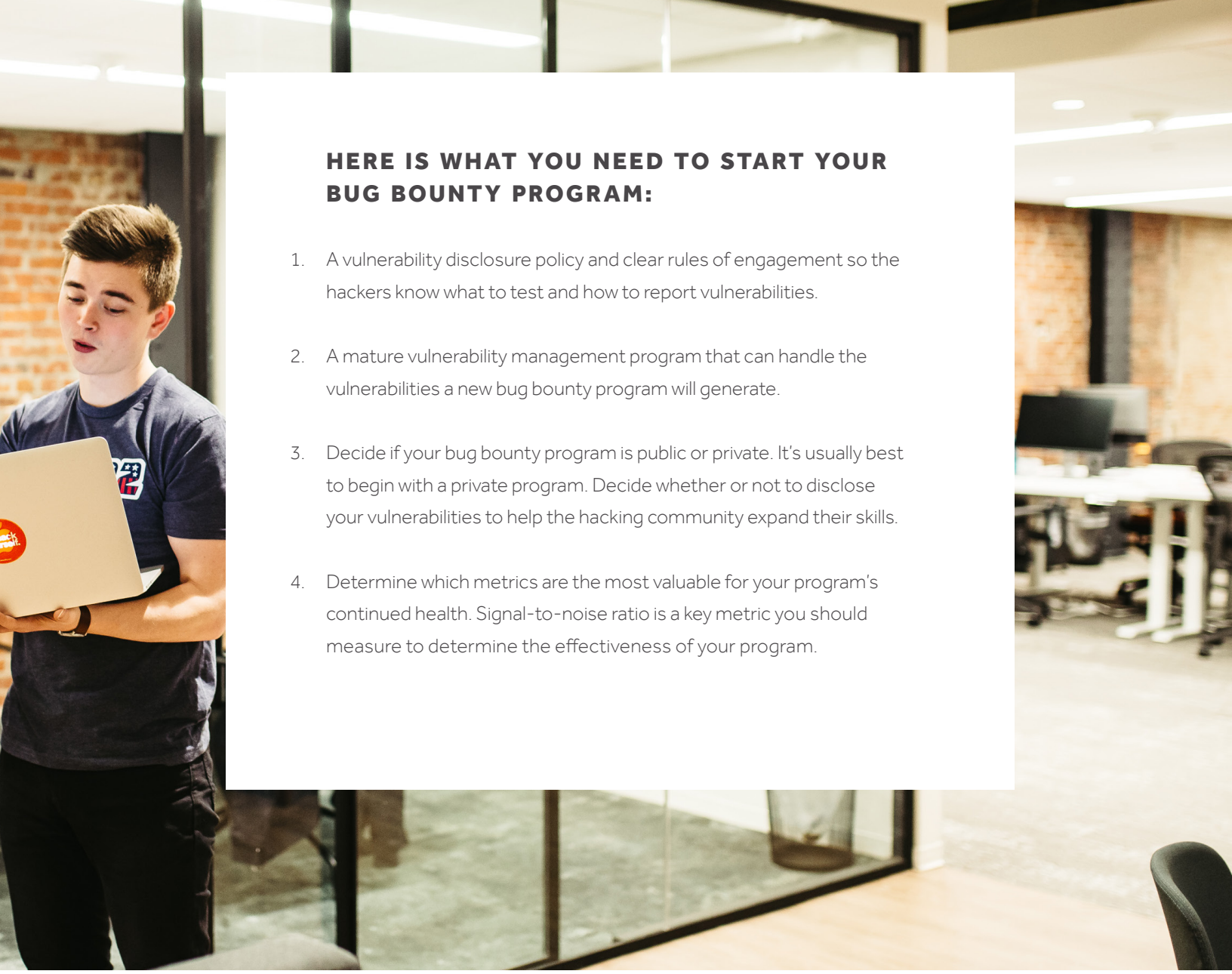
With a mature vulnerability management process in place, you'll be better equipped to add bug bounty into your security workflow. Security in the DevOps world needs to remain "non-blocking," or not hamper your developers' ability to deliver software on time. Bug bounties are a natural fit, as hackers will always be working "behind the scenes" to keep your software safe.

The Roles of People

The increase of administration and management a bug bounty suggests may seem daunting to an already over-extended security team. Let's now take a quick look at what people you'll need and some strategies to help ease bug bounty into the DevOps workflow.

The organizational design of your security team can help or hurt your ability to incorporate bug bounty smoothly. Security teams should be focused on providing services to the development team, not ordering them around as "code cops." Some examples of services provided by the security team include code reviews, design reviews, security testing, vulnerability management, or research on behalf of the development team. A team set up with a service-first mindset will be able to find a place for bug bounty within the vulnerability management service.

The security team also has many decisions to make. What is the scope of your program? How will communication happen with the hackers who submit bugs to you? Will your program be public or private? What metrics do you need to be effective? These questions are more easily answered when a mature vulnerability management program already exists, but there still are items unique to bug bounty programs.



HERE IS WHAT YOU NEED TO START YOUR BUG BOUNTY PROGRAM:

1. A vulnerability disclosure policy and clear rules of engagement so the hackers know what to test and how to report vulnerabilities.

2. A mature vulnerability management program that can handle the vulnerabilities a new bug bounty program will generate.

3. Decide if your bug bounty program is public or private. It's usually best to begin with a private program. Decide whether or not to disclose your vulnerabilities to help the hacking community expand their skills.

4. Determine which metrics are the most valuable for your program's continued health. Signal-to-noise ratio is a key metric you should measure to determine the effectiveness of your program.

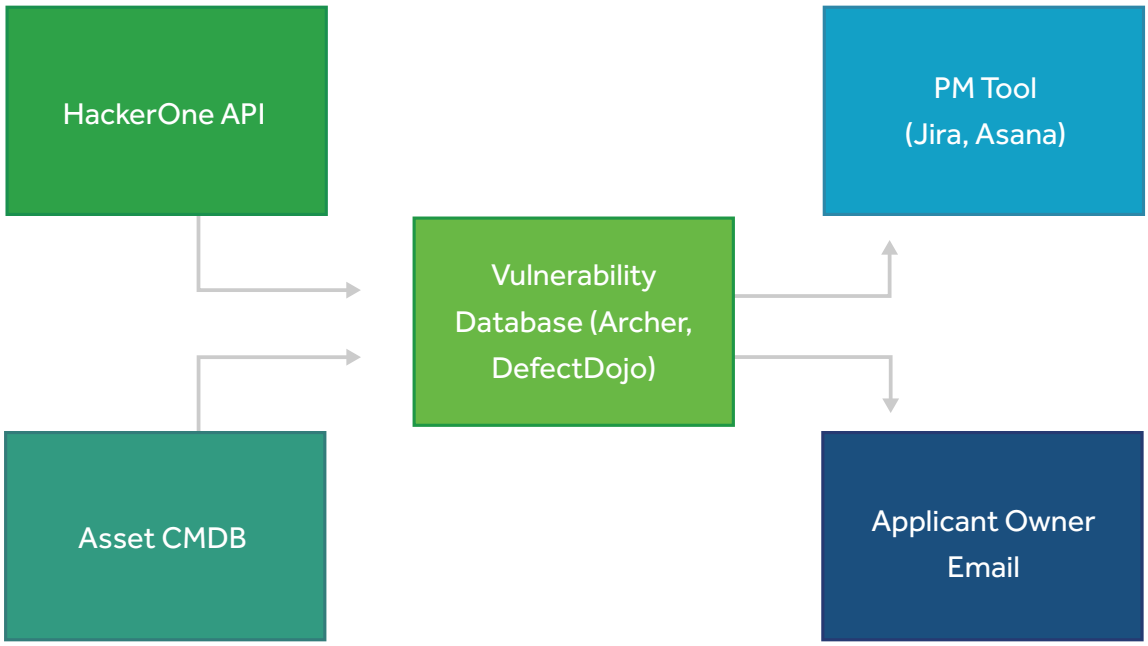
Product teams are the downstream consumers of the vulnerability management process, and therefore shouldn't be in contact with hackers from your bug bounty. The security team should handle triage, verification, and publication of vulnerabilities to the product teams (or you can work with HackerOne to handle everything, read on for more).

In turn, the product team is responsible for provided fixes in a timely manner and reporting these back to the security team. The security team can then provide a formal disclosure of the vulnerability to interested parties or customers of your product.

The Role of Automation

One of the many enablers of DevOps is automation. The ability to do things with computers previously only possible with human intervention has been a game changer. Security scanning tools are an example of the types of automation that can help secure DevOps.

Automation fits well into bug bounty programs as well. HackerOne's platform is built to help hackers report possible vulnerabilities to you. But we know you likely have other systems built to manage vulnerabilities from your existing sources. HackerOne's API make integration and automation easy so you can incorporate bug bounty reports into your vulnerability management workflow.



HackerOne's API allows easy integration with your existing tools. This is what a vulnerability management toolchain may look like with HackerOne included.

It's important to first know what all of your assets are and their configuration. A Configuration Management Database (CMDB) stores all of your assets in one location. You may also have a vulnerability management database such as Archer or DefectDojo to store vulnerabilities tied to those assets.

HackerOne's API allows you to search HackerOne's platform for new vulnerabilities related to your assets on a regular basis. New vulnerabilities found are then placed into the vulnerability database of choice, kicking off a triage workflow for the security team. Once the bug report is verified, emails can be sent to application owners and the vulnerabilities can be sent to a bug tracking tool such as Jira or a project management tool like Asana.

Using automation helps to keep human intervention to a minimum and keeps the product team flowing smoothly. The product team has no huge chances in their process. A new bug report is created and handled like any other, regardless of the source.

Adding the Expertise of HackerOne

Not all organizations have the resources to completely own the bug bounty process. Communicating with hackers and performing triage on new vulnerabilities may be unrealistic at this time.

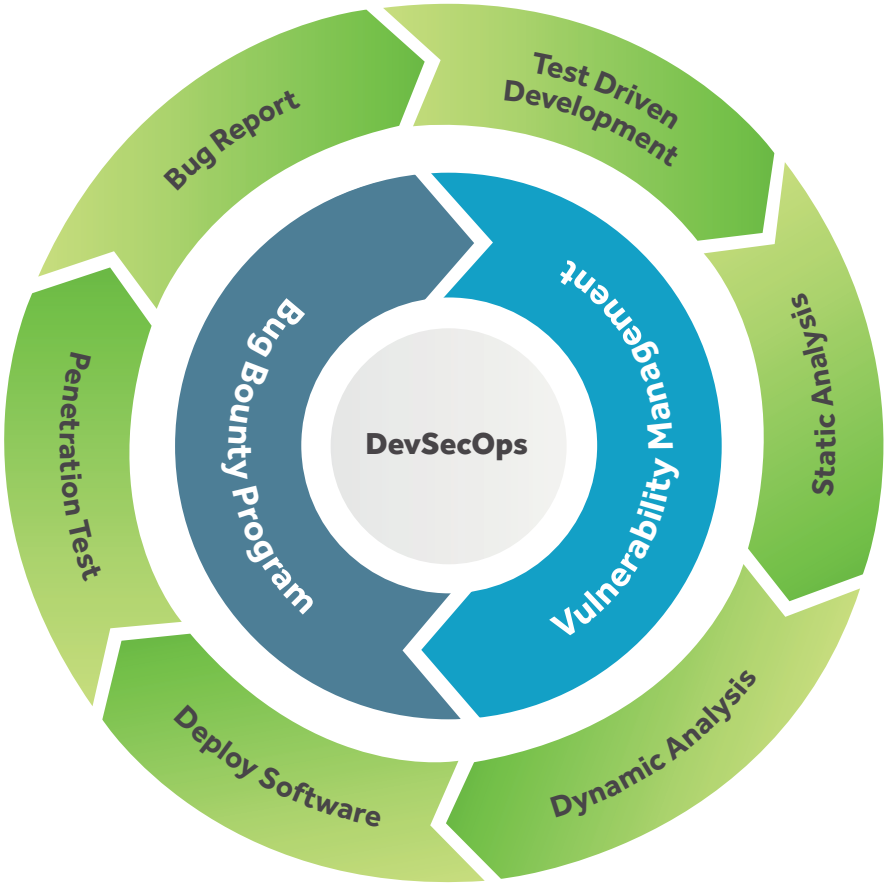
If you're in this situation, HackerOne can help you get your bug bounty off the ground with a fully managed bug bounty program. HackerOne's experienced security analysts will communicate with hackers and validate all submissions made through the bug bounty platform. Your security team will get only valid, well-documented vulnerability reports.

If you feel you're ready to take over the triage duties, you can still use HackerOne's platform and manage the bug bounty yourself. The choice is yours. Whether HackerOne manages it or you do, all organizations can make their software more secure with hacker-powered security.

Let Hacker-Powered Security Augment Your Security Team

Adding security to a DevOps program takes many steps and many tools. But not every tool is the most effective way to secure your applications. Automated scanning and unit testing are great first steps, but they usually aren't enough for companies to have complete confidence in the security of their applications.

Bug bounty programs put many eyes on your application and feature a unique way of testing your applications. **The testing is continuous, ongoing, and mirrors the development itself.** It fits in well with the spirit of DevOps and agile development methodologies.



BUG BOUNTY AND THE SDLC

Hacker-powered security works hand-in-hand with your existing DevOps workflow. The bug bounty program becomes a source for your vulnerability management program. The vulnerabilities are handed to the product teams who build a fix, test it, and deploy.

Why bug bounty fits with the DevOps workflow

- Bug bounty acts as a continuous penetration test
- Hackers understand, and test, the latest exploits faster than automated scanners
- Bug bounty scales your application security efforts to keep up with the growing number of applications security teams are responsible for.

What's needed for hacker-powered security?

- A mature vulnerability management program
- A service-focused security team
- A flexible platform with access to top hackers and service expertise you can rely on

You can achieve maximum security of your applications by completing your secure SDLC with a bug bounty program. HackerOne is here to guide you through the entire process.There's a community of thousands of ready to test your applications and ensure maximum security.

About HackerOne

HackerOne is the #1 hacker-powered security platform, helping organizations find and fix critical vulnerabilities before they can be exploited. More Fortune 500 and Forbes Global 1000 companies trust HackerOne than any other hacker-powered security alternative. The U.S. Department of Defense, General Motors, Google, Twitter, GitHub, Nintendo, Lufthansa, Panasonic Avionics, Qualcomm, Starbucks, Dropbox, Intel, the CERT Coordination Center and over 1,200 other organizations have partnered with HackerOne to resolve over 90,000 vulnerabilities and award over $42M in bug bounties. HackerOne is headquartered in San Francisco with offices in London, New York, the Netherlands, and Singapore.

Learn more by visiting our website or contacting us today.