CASE STUDY

Mercado Libre's Journey to a Public Bug Bounty Program

Mercado Libre, host of the largest online commerce and payments ecosystem in Latin America, first engaged HackerOne in 2017 to ensure they could meet critical security objectives such as delivering secure software fast enough to meet growing market demand and improving production security across the software development lifecycle (SDLC). Mercado Libre started with two private programs, HackerOne Bounty and HackerOne Response, and also leveraged HackerOne Triage as they began their journey into the powerful world of ethical hacker engagements.

Six years later, the team at HackerOne is delighted to announce the launch of Mercado Libre's Public Bug Bounty Program. After six years of private VDP and Bug Bounty, Alejandro Federico Iacobelli, Application Security Director at Mercado Libre, offers many practical insights and learnings from his programs.

Leading up to this latest milestone in Mercado Libre's security journey, Iacobelli shared his written reflections on the early days of their crowdsourced security program, how they continue to improve their program and SLAs, the importance of the relationship between his team and the hacker community, and why he's taking his program public now.

We're excited to share Iacobelli's learnings with other current and future bug bounty program leaders, as well as introduce Mercado Libre's public program to the global hacker community.

hackerone | mercado libre

CASE STUDY

Author: Alejandro Federico Iacobelli,
Application Security Director, Mercado Libre

Since the emergence of XP in the mid-'90s, agile development methodologies have rapidly gained popularity. Undoubtedly, this set of prototyping approaches offers numerous advantages, such as reduced time to market, which is essential for most industries today. However, in terms of application security, it has also introduced additional challenges.

One of the main challenges was, and still is, how to scale up all the SSDLC (Secure Software Development Life Cycle) checkpoints to match the speed of these methodologies. "Comprehensive Vulnerability Assessment" is a clear example of an unscalable checkpoint. Outsourcing quality offensive testing in scenarios with hundreds or thousands of deployments per month was not only expensive but also impractical due to the limited qualified supply.

Crowdsourced Security

A decade ago, a complementary and more scalable approach to vulnerability assessment emerged. The basic idea was to tap into the massive talent hidden within the crowds, giving rise to the concept of crowdsourced security. Bug bounty programs have been one of the most successful implementations of this idea, not only because of the massive amount of issues being found [1] or the wide range of viewpoints any company can benefit from but also because of the impact these types of programs have had on many companies' cultures.

Bug bounty programs provide companies a way to connect with a global talent pool of security researchers who serve as an extension of the company’s security team and can be available at all times to find and report vulnerabilities in exchange for bounty payments and reputation. This constructive collaboration allows companies to tap into subject matter experts at any given time, with the end goal of making the internet safer for all of us.

hackerone | mercado libre

Our Journey to a Public Bug Bounty Program

Six years ago, we implemented two private programs in collaboration with HackerOne, to ensure we maintained the highest security standards for our growing digital landscape, including a vulnerability disclosure program and our bug bounty program in collaboration with HackerOne. At that time, we had a small team, a limited budget, and no real experience in handling such a program. As a result, we chose to start with an invitation-only approach. Since then, our three primary annual OKRs have been:

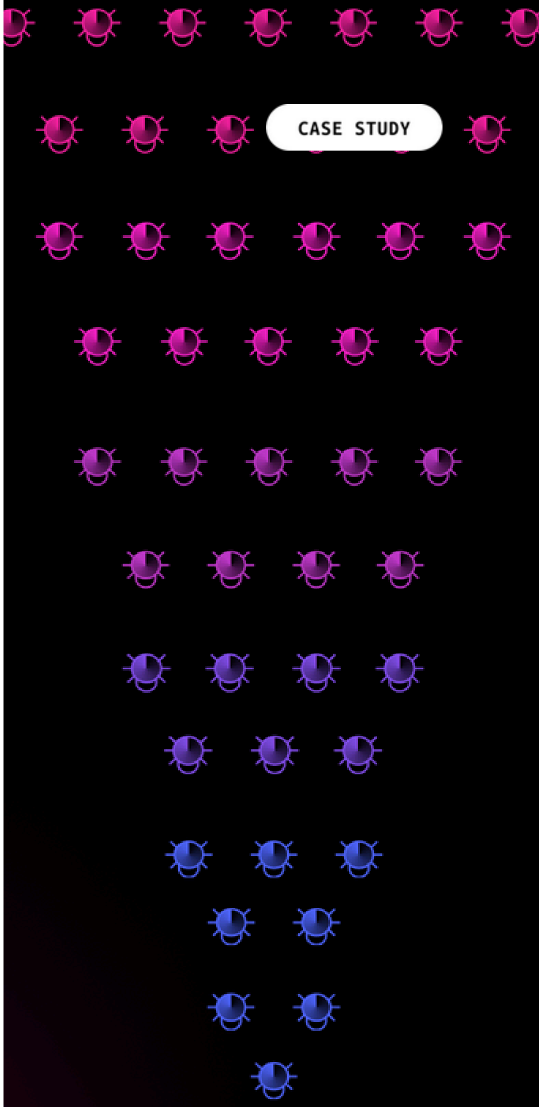
- To double the number of active researchers (those with at least one valid medium/high/critical impact report).
- To continuously expand our eligible scope in a structured and constant manner.
- To have a healthy response efficiency, especially “time to bounty” and “time to fix.”

We also keep a close watch on a secondary OKR that pertains to the number of reported high/critical vulnerabilities. Since 2018, we’ve got some interesting insights.

Community Growth

Growth in Players with Valid Reports: Since 2020, we have observed an average yearly growth of 62.5%. However, there was a marked deceleration in 2023, with the growth rate declining to 8.05%. We have some theories behind this behavior. On one hand, we’ve reached almost all LATAM registered researcher communities, one of the most interested players due to the fact that they are also customers. On the other hand, there is a psychological concept called hedonic adaptation. In a nutshell, people’s happiness tends to fade out as they get accustomed to a specific thing. This is why things like constant scope update is a good retention strategy.

Growth in Players with Accepted Invitations: Concerning researchers who accepted our invitations, our numbers have surged significantly. We have seen an average yearly growth rate of 250%, but so far, this surge has had no real effect on the other OKRs we track.



Loyalty

Year-over-Year Retention: Since 2019, our YoY retention has increased from **20.59%** in 2020 to **34.57%** in 2023.

Multi-Year Retention (3-year retention): Starting in 2021, the MYR has also seen growth, rising from **9.26%** to **16.5%**.

Churn Rate: Since 2020, our churn rate has remained steady at **65%**.

Reports

Monthly Average:
We consistently track a crucial metric: the average number of reports per week. Over the past three years, we've successfully maintained the same average through a combination of dynamic surfaces, custom promotions, and changes in the invitation rate. However, it's noteworthy that during months when we conduct hacking events, there's a notable increase in this average — typically a 1.5-fold rise — before it reverts to the usual level.

Yearly Growth:
The total number of reports per year has doubled in five years (if we compare 2019 with 2023). This makes sense because we've also increased our attack surface and active researcher community by a factor of 10.

SLA

Resolution Times:
Our average SLA accomplishment improvement since 2019 was around 18% YoY. This means that we started with 60% of high and critical vulnerabilities being fixed within SLA at the program launch in 2018, and almost five years later, we are over 87%.

Response efficiency:
Our actual response efficiency (according to HackerOne statistics) is 16 hours to first response, one day to triage, on average, and three days as the average time to bounty. 97% of the reports we receive meet the H1 response standard.

Why Are We Going Public Now?

The straightforward answer is that despite our efforts to increase the invitation rate, payment amounts, bug eligibility criteria, scope, platform documentation, payment processing, and bug fixing times, we have reached the same number of active researchers for two consecutive years.
Our mindset is that if the same group of researchers is constantly looking for bugs over a long period of time, this approach will eventually resemble outsourcing, losing the advantages of crowdsourcing. This is why we needed to find a way to make more people aware of our program, and going public is our natural next step.

Throughout this journey, we've worked hard to become a trusted partner to the researcher community and have gained valuable insights to leverage to constantly make improvements to our program and scope. Here are some of the key learnings that have prepared us for this milestone:

1. Learn how to craft a program policy that targets the specific bugs and products of your interest, all while preserving the program's "playability."

A clear and concise policy is a key aspect that all program managers must learn how to build. Apply the “keep it simple” software design principle. Avoid long and complex policies that no researcher is going to read. Invest time in constant refactoring. Focus only on the information that is useful to researchers. Some key aspects are:

Scope: Whitelist and blacklist approaches are both valid. Choose the approach that is cleaner and more suitable for your needs. Avoid using infinite lists of IP addresses or subdomains and consider the "playability" factor, especially if you have a multi-subdomain/domain application. Researchers do not want to monitor every step of the way if one of their gadgets for a specific finding are out of scope.

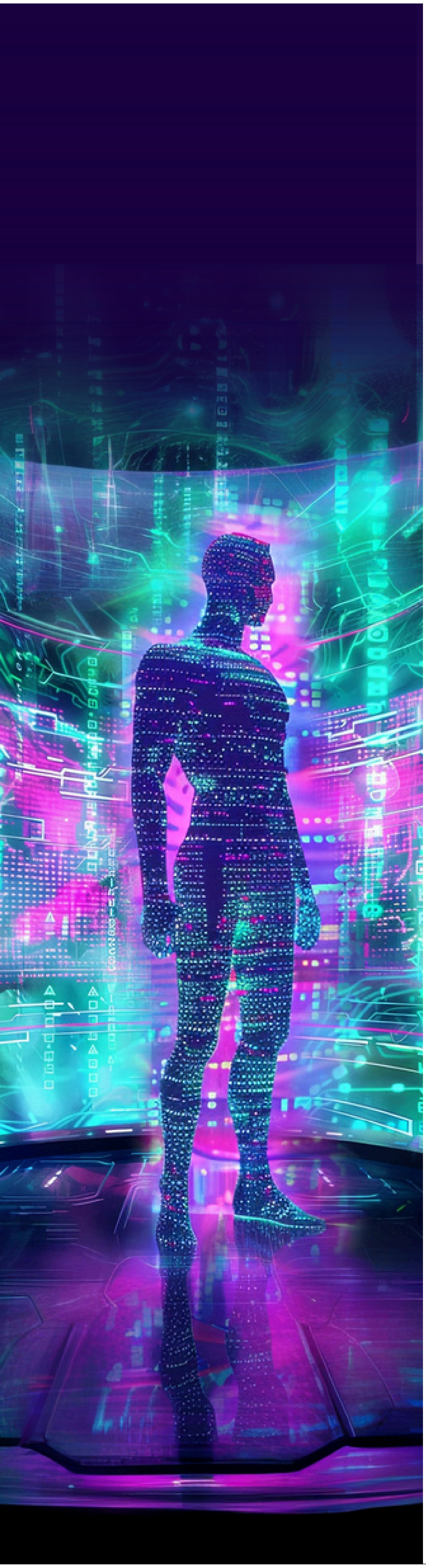
Rewards information: Most bounty hunters participate for monetary rewards, so providing them with more visibility is beneficial. Instead of fixed values, prioritize ranges based on impact and scope. Two vulnerabilities of the same type often differ in their impact, and using ranges allows for more accurate rewards to be assigned.

Response times: Most bounty platforms offer a response efficiency indicator to researchers. It is a good practice to include a self-imposed Service Level Agreement (SLA) in your policy. This allows researchers to compare both indicators and ensures that you prioritize timely responses and take time management seriously.

Qualifying and not qualifying bugs: As a company, being crystal clear about the specific types of bugs you are interested in is crucial for researchers to understand what to focus on. The longer the list of non-qualifying bugs (NQL), the fewer researchers will be interested or able to assist you. A lengthy NQL is also indicative of significant technical debt on your end. Furthermore, if your NQL continues to increase over time, it may be perceived as a sign of low maturity.

Testing tooling: Test users, test credit cards, and test endpoints (for scenarios like SSRF post-exploitation) are some tools that help researchers avoid unnecessary onboarding frictions.

If needed, provide links to product and technical documentation: Understanding what the product is about is one of the first things any researcher does in the passive information-gathering step. The faster this stage is completed, the faster they can find bugs.



2. Implement internal vulnerability mitigation SLAs and test your team's response capacity under different conditions.

A straightforward idea that you should follow is to pay for risk reduction, not just risk identification. Spending 1MM USD on 1,000 critical bugs but taking one year to fix them is not an efficient use of the security budget and leads to many negative externalities, such as bad actors exploiting known vulnerabilities or an increase in duplicate reports, one of the most common reasons hunters drop out. To honor this principle, vulnerability management is the most important process to polish. This process has some main ideas worth mentioning:

Real-time event-driven integrations: Accessing a non-everyday tool for triaging, like the HackerOne frontend, could have a negative operational impact. This is why they offer webhooks as a mechanism to build event-driven integrations. The most important integration is probably your issue tracker, followed by any other alerting tool, like Slack or Opsgenie.

Metadata enhancing: Issue metadata is essential if you want to optimize your vulnerability management process. For instance, you can utilize specific metadata to aid in the reporting and routing process. Automatically assigning the triage and development teams based on criteria like the domain where the vulnerability was found is an effective method for reducing manual tasks. Institutionalized SLAs: Bug-fixing SLAs must be enforced by company policy. Penalizing teams during quarterly evaluations if SLAs are not met has proven to be an effective way of communicating that security is a priority.

Duplicate findings feature: When you are triaging dozens or hundreds of reports at the same time and with a decentralized team, the probability of receiving the same report tends to be high. Implementing a duplicate report finder is something that you want to have in your arsenal.

Standardized templates: Elements such as a criticality calculator, vulnerability description, technical mitigation, or type of vulnerability are fields that, if not standardized and enforced by proper issue tracker features, can result in a significant waste of manpower. This is aside from the fact that it will aid in leveling the team's knowledge.

On-demand triage reinforcement: It's typical to experience occasional spikes in report submissions. In such instances, the optimal approach is to dynamically allocate additional triggers (while managing other projects concurrently) to uphold the essential program metrics.

3. Set a budget according to your company's Application Security maturity and pay as soon as possible.

Another essential choice to make is how much are you going to pay per valid report. In this matter, overpayment is as bad as underpayment. Some studies [4] have shown that most researchers are price inelastic. There are some variables that you should consider in your payment equation:

Report impact: It’s all about incentives. If you want your worst bugs to be found first, you should put the incentive on impact.

Scope being affected: Not all business units or products are equally important for the company's main strategy. Incentives should go to the more important products in terms of revenue.

Report flow per time unit: One statistic that you should constantly monitor is how many valid reports you have per unit time. Having low vulnerability rates is a clear indicator that you should increase your incentive for researchers to play.

Application security program maturity: Paying big amounts of money for bugs that you can find with an open-source tool is not the best way to spend your resources. If you are in a scenario where you need to pay more, but your maturity is not enough, a good trade-off is to only pay for critical findings. This will leave the trivial detected reports out-of-scope.

4. Invest extra effort in making your crowd engaged, remembering that not all researchers are attracted to the same incentives.

Each day, an expanding number of companies are joining the bug bounty arena. As a result, researchers have an increasingly diverse range of programs to select from. This underscores the importance of brand loyalty. Here are some ideas that could be helpful:

Hacking Events: The definition of a community is “the condition of sharing or having certain attitudes and interests in common,” and hacking events are great examples of places where the community can meet and share knowledge under your brand’s flag.

Custom Swag: Custom-made swag is a great way to personalize the researcher's experience. Creating custom stamps for your top researchers is an excellent motivational gesture that goes the extra mile.

Feedback surveys: You can't improve what you don't measure. Quarterly surveys are a good way to understand what can you do better to make your researcher’s life easier. Even in the worst-case scenario, where your questions remain unanswered, it serves as a clear sign that you need to focus on strengthening your loyalty.

Out-of-scope concessions: Just because a vulnerability is considered out of scope doesn't imply it has no impact on your organization. Therefore, if it's the first time a specific researcher has reported a bug to you, and the vulnerability is classified as P1 or P2, offering a concession and paying it at a lower rate could serve as a good incentive for the researcher to continue participating in your program.

Ambassadors: Each country has different dynamics, levels of maturity, and references. Offering incentives to individuals who bring more researchers to your program is a good idea.

Dynamic promotions: Older programs tend to receive fewer bug reports than newer ones, even if you pay higher bounties. One reason for this is that testing the same functionality repeatedly can become monotonous. Sending promotions for new functionalities helps researchers shift their focus toward fresh areas where low-hanging fruit vulnerabilities could still be available.

Read the research: There are excellent studies that have been conducted on what motivates people to play a program [2][3]. It's a good idea to read as many as you can and draw your own conclusions.

5. Learn from your mistakes. Remember that the SDLC is a dynamic process.

One valuable insight that a bug bounty program offers is untainted statistical information about which vulnerabilities are more prevalent than others over time. When a pattern emerges, it's crucial to identify the root cause and implement a strategy to address that pattern company-wide.

You should use your bounty program as a thermometer to gauge how effective you are in developing mature capabilities to address entire classes of vulnerabilities over time. A clear indicator that you're not performing well is if you encounter the same types of vulnerabilities year after year.

6. Bug bounty is not a replacement for penetration testing exercises.

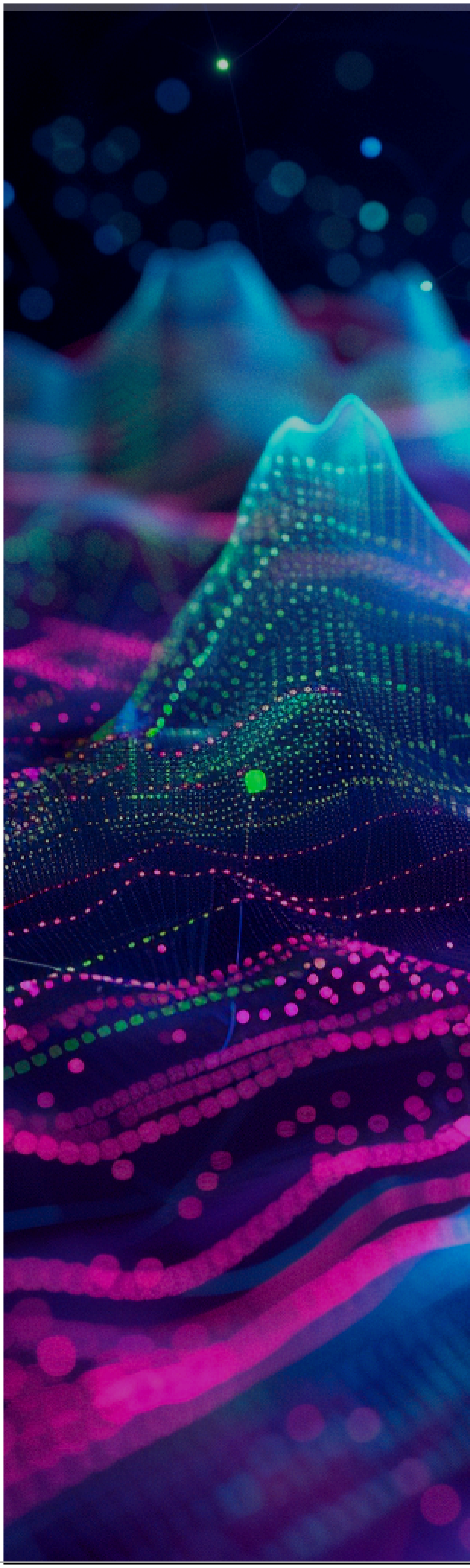
You should never stop doing penetration testing exercises at the expense of a bug bounty program.

There are numerous reasons that support the idea that these two activities are not contrasting but complementary. Let's mention some of the most important:

Crowdsourcing vs. Outsourcing: The primary distinction lies in the philosophy behind both approaches. When you opt for outsourcing, you seek highly skilled professionals. Crowdsourcing, on the other hand, is based on a different set of theories, such as the "diversity trumps ability" theorem, and concepts like Joy's law, which states: "No matter who you are, most of the smartest people work for someone else."

Trust: On one hand, when you commission a penetration testing exercise, you usually know the researchers involved. You've perused their resumes and have had firsthand interviews with them, making trust a significant factor. Conversely, bug bounty researchers are predominantly anonymous. While major platforms like H1 are striving to encourage researchers to undergo Know Your Customer (KYC) and background checks, there's still a long way to go.

Goal: When you commission a penetration testing exercise, you typically set a clear objective. The researchers might exploit one or fifty vulnerabilities to reach that objective, but the primary focus is on achieving the set goal, not on the number of vulnerabilities they uncover along the way. Conversely, a bug bounty is more akin to a vulnerability assessment. Researchers are encouraged to identify as many vulnerabilities as they can without a strict direction or purpose.



7. Researchers who are also customers tend to find better vulnerabilities.

A statistic we've found is that users of our ecosystems tend to find better and more critical vulnerabilities than people who are just passing through. One possible reason for this is the fact that, as long-term users, they are very familiar with all the functionalities that are being offered. This gives them an advantage in terms of information gathering and business understanding. Another important reason for this fact is that is always easy to create regional testing kits. This means test users, valid KYC, and working credit cards to test all the flows. Being a custom user helps to avoid that tedious onboarding, too.

What's Next?

Our expectation with this strategy is to capture casual players who possess valuable insights, to pique the curiosity of some long-term players, and to challenge a community of over 500k with monthly custom challenges and promotions. Additionally, we aim to build a market so attractive that it deters users from turning to black markets. We're excited to welcome new researchers to our bug bounty program.

Learn more about our bounty program and scope.

References
1. Marten Mickos, LinkedIn
2. Bug Hunters' Perspectives on the Challenges and Benefits of the Bug Bounty Ecosystem
3. Hackers' self-selection in crowdsourced bug bounty programs
4. Hacking for good: Leveraging HackerOne data to develop an economic model of Bug Bounties

hackerone | mercado libre

With over 2,000 customer programs, more companies trust HackerOne than any other vendor

HackerOne has vetted hackers for organizations including:

gm · Lufthansa · ZEBRA · zoom

Spotify · citrix · PayPal · Uber · HYATT

U.S. Department of Defense · Google · reddit · Nintendo · Adobe

A.S. Watson Group · sumo logic · yahoo! · priceline

shopify · slack · yelp · salesforce · TOYOTA

Contact us