Bug Bounty Hunting

Mgr. Jan Kvapil

EurOpen 2022

Abstract

Looking for vulnerabilities in software—previously, often practices resulting in illegal activities, now, an (extra)ordinary occupation. How to look for those vulnerabilities legally and help to secure the software for others as well? Where and how to submit the vulnerability disclosure report and corresponding proof of concept? Apart from the answers, we will take a look at some publicly disclosed reports. Next, we will see some of the available tools and also wonder, how vulnerability disclosures fit into the open-source world and whether you should consider starting a bug bounty program.

Disclaimer

This text is based on my (admittedly short) personal experience in bug bounty hunting and vulnerability research. The area of interest is quite broad and would benefit from more formal research. Maybe I will come to that some day in the future. As such, take this paper as light and introductory material into the interesting world of bug bounty hunting.

The following text is educational and the author does not take any responsibility for any acts the reader might decide to partake in.

1 Introduction: an ethical hacker

Following precisely how the word *hacker* has been used and perceived by various groups and the general public would probably suffice as a topic for small research on its own. In my dictionary, this word carries less and less of a negative connotation as time progresses. While I have yet to finish the book Hackers: Heroes of the Computer Revolution by Steven Levy[1] I can already recommend it in order to help the reader with rebuilding the intuition behind the mindset of *a hacker*. A hacker is strongly driven to understand and explore all the *nooks and crannies* of a given system—even, or especially, if the authors of the system, its documentation and other sources assure that everything works perfectly. Adding *ethical* in front of the word hacker makes it transparent to

[1] ISBN-13: 978-1449388393

the public (and the hacker as well) that the hacker has an inner moral compass that he or she follows.

The phrase *bug bounty hunter* might also have its connotations. Namely, that the motif of a bug bounty hunter is solely a financial gain. While such motifs can be present, there are others as well. And if seen as an occupation, bug bounty hunting does not differ from the others.

Now, I would like to walk the reader through the topics and processes related to bug bounty hunting and ethical hacking.

1.1 Platforms and programs

A hacker spends his time exploring, testing and analysing various pieces of software. However, such practice can lead to a significant crossing of legal boundaries. Consider for example the Computer Fraud and Abuse Act—*Whoever having knowingly accessed a computer without authorization or exceeding authorized access...* has quite probably committed a criminal offence under this act. Interestingly, the evolution in the security field has lead to the creation of *platforms* that help to legally connect hackers with companies. Those platforms brought also a certain notion of gamification to the process. While the details of the security vulnerabilities are disclosed only under certain conditions (e.g. after mutual agreement between the hacker and the company) some information can be public, such as the reputation and impact of the hacker.

Both hackers and companies can register on the platforms. The company creates a Bug Bounty Program (a *program* in the following text). The programs can be *public*, i.e. anyone can see them. For example, this is the U.S. Department of Defense's program. Another option is a *private* program, whose visibility is limited only to certain hackers. The incentives to have a private program can vary, one can imagine that making the program public could result in too many reports that a small company won't be able to handle. Access to a private program can be given after gaining a certain amount of reputation.

In general, a company can host Bug Bounty Program directly. However, there could be a non-trivial amount of legal and financial paperwork (payouts for bounties going all over the world). Similarly to other business processes, bounties are subject to international laws and also sanctions, which is quite relevant at the time of writing this text due to the ongoing Russian invasion to Ukraine. The response of one of the platforms is worth reading.

Some of the known platforms are HackerOne, Bugcrowd, Intigriti (however, there many more). Understandably, the tech *giants* are capable of hosting their own programs, such as Microsoft, Google. One might ask: Can the platforms themselves be a target of an attack? I'll answer by quoting the last platform's website:

> After something like 2 years of virtual blood, sweat, and tears to build this site, it took a bug hunter only a few hours to discover our private APIs.

It is not a surprise that the well-known software companies have their own program, however, there are also other programs worth noting. Namely, The Internet Bug Bounty program (sponsored by multiple companies), whose scope covers more and more open-source projects (e.g. The Ruby Programming Language, Ruby on Rails, OpensSSL and others) as time progresses. Another interesting program is the one ran by the European Commission (hosted on Intigriti). Finally, it might be of interest that Google's program also spreads outside of Google's own projects to some open-source projecs (e.g. OSS-Fuzz projects among many other) and also to security research in general.

2 Bug hunting

I will now show the steps leading to submitting a vulnerability report in order to provide a better insight into it. I am most familiar with HackerOne, therefore the following scenario is based around this platform.

2.1 Where to start

There is a multitude of tools (most of which I haven't tried) that can probably help with finding a vulnerability. That said, relying on tools and skipping over fundamental understanding is not something I would recommend. For example, a report that consists of the output from a static analysis tool won't be accepted—a clear impact has to be shown and discussed at a minimum. Information security is broad, therefore I would suggest starting small and in an area, one is familiar with. However, hacking is a constant process of refining one's skills and learning about new technologies, practices, etc. One of the first programs that I have been reviewing was a web application written in PHP. I have dived into PHP, and its documentation and started looking for the weak spots. The ability to learn quickly and navigate through lots of materials at once is beneficial—and soon interesting things can pop up (such as "types juggling" in PHP).

To start safely, HackerOne provides a nice sandbox consisting of multiple vulnerable applications of various levels of difficulty. In case the reader starts finding the first issues in a matter of hours or days he can try starting hunting on a real program. It is worth reiterating that bug bounty hunting is *a delicate practice* from the legal perspective and my suggestion is to **read** the platform's Terms and Conditions (ToC) carefully this time.

2.2 Finding a program

In general, I see two approaches, on one hand, seasoned hackers are able to work on multiple projects at once, especially if they prefer automation over a manual review. On the other hand, some stick to a single program for months or years. The first group benefits from automation and the ability to catch the low-hanging fruits, the second one builds a much deeper understanding of a

given service—and the understanding pays off because it is easier to showcase the impact of a vulnerability. Finding vulnerabilities is not an easy task and starting with software or app that one is familiar with gives a head start. There are multiple well-known public programs on HackerOne, for example GitLab. The downside of public programs (from the point of view of a hacker) is that many other hackers are working on them already, which lowers the chances of finding an original bug (because most programs do not pay off bounties for duplicate reports).

Once a program is chosen *things* start to get exciting—looking for a security vulnerability can bring an adrenaline rush. However, similarly to reading ToC, it is **a necessity** to carefully go through the program's policy and scope. The assets in scope can be targeted and hacked by the attacker. The programs can whitelist the assets (i.e. the "In scope" section) in multiple ways. For example, Curl only mentions a single asset in scope and that is its repository on GitHub. Others provide a much broader scope—consisting of raw IP ranges, links to binaries, etc. Programs can tag the scopes with the technologies (at least the main ones) that the asset is built upon, presumably to help the hacker direct the attention. Similarly, some assets are explicitly out of scope and hacking them is prohibited.

Following the policy and staying within the scope is essential to remaining an ethical hacker. For example, this is an excerpt from GitLab's policy:

> Any activities conducted in a manner consistent with this policy will be considered authorized conduct and we will not initiate legal action against you. If legal action is initiated by a third party against you in connection with activities conducted under this policy, *we will take steps to make it known that your actions were conducted in compliance with this policy.*[2]

A policy can also contain slightly surprising clauses, such as this one from PayPal's program:

> As a condition of participation in the PayPal Bug Bounty Program, you hereby grant PayPal, its subsidiaries, affiliates and customers a perpetual, irrevocable, worldwide, royalty-free, transferrable, sublicensable (through multiple tiers) and non-exclusive license to use, reproduce, adapt, modify, publish, distribute, publicly perform, create derivative work from, make, use, sell, offer for sale and import the Submission, as well as any materials submitted to PayPal in connection therewith, for any purpose. You should not send us any Submission that you do not wish to license to us.

While understandable when viewed from the program's side, similar statements made me question, whether I would submit a report to such a program.

[2] Emphasis mine

2.3 Looking for a bug

After agreeing to the policy and understanding the scope one can finally start hacking. The first thing is to get familiar with the program—is it a web application? Then go ahead, register and start using it. The platforms often maintain `<username>@<platforms-domain>` e-mail that is meant to be used for such registration. The program can then easily monitor and filter those accounts from the ones of the real users. Some programs provide test account credentials at request or as a part of their policy. It is important to understand that a bug in one program can fall into the *won't fix* category, but be a security vulnerability in a different context. The goal of a hacker is to find a vulnerability and show its **impact** if exploited. Of course, there is no single path how to find a bug. There are many tools and resources that can help out and the pentesting, hacking and security community comes with new ones often. For example:

- BurpSuite: well-known tool for web security and penetration testing,
- Payload All the Things: a list of various payloads such as Cross Site Scripting payloads,
- Amass: a tool for network mapping of attack surfaces,
- sqlmap: tool for detecting SQL injections,
- HackerOne resources: a list of other tools categorized by their area of interest.

While such tools can be of great help, one needs to be careful to understand first what the tools do and how (again, complying with the policy and scope is crucial). However, there are more basic tools that are enough for starting out. Fluency in some kind of a shell program is at the base level. Often, the first tool I use is the command-line `curl`—it is great at testing out APIs. Next, the ability to script in languages like Python goes well alongside that (combined with libraries such as pwntools). Most of my proof of concepts take advantage of a simple Python script. The necessary tool set depends on the target, but basic DevOps tactics, for example spinning up multiple Docker containers, do come in handy. Testing locally denial of service attacks is much more doable if a successful attack crashes only a container and not the host system.

In general, the hacker does not need to be a coder in order to find issues (e.g. information disclosure vulnerabilities can sometimes be found by simply using the website and *clicking* around), but it empowers the hacker greatly. That said, the current browsers already provide enough information through developer tools that can be used to start investigating the web traffic and look for weaknesses.

At this point, it is good to articulate the common phrase that *the attacker only needs to find a one way in.* I want to stress this point because while there are rules, one can still find quite unorthodox ways of gathering meaningful information. Once I needed to learn, whether and how a target uses a particular

piece of software. Its own assets and domains did not tell me much, but a quick Open Source Intelligence (OSINT) provided me with an hour-long podcast from one of the lead developers of the target. The podcast was meant to be interesting for other developers, but at the same time, it has given me valuable information on how that particular software is being utilized. A hacker needs to be creative and unorthodox in his ways and the application of his skills. A piece of information is often not harmful on its own but becomes when put into a mosaic of others.

2.4 Submitting a report

Once a bug is found the vulnerability disclosure report can be created. Depending on the platform it has to contain the vulnerable asset (e.g. a domain), type of weakness (e.g. buffer overflow) and its severity—which can either be set directly (e.g. High) or calculated using the CVSS calculator. Then the most valuable parts come, the proof of concept (PoC) and the impact.

Depending on the program the reports are first read and triaged by someone from the platform (in order to avoid spam) and only then handed over to someone from the program or processed by the program directly. Writing a report and communicating clearly with the program is another skill set worth having for a hacker. It is not unheard of that the communication within a report does not go smoothly. Also, the triager might lack the particular skill set required for successfully exploiting the vulnerability. Keeping the proof of concept succinct and clear is wise. Ideally, I try to provide a single `curl` command that showcases the vulnerability or a simple Python script. If possible (e.g. for some OSS projects), I add a container with the required provisioning to minimize the work for the triager. Some reporters create a screencast to give proof of the vulnerability (and also provide the commands)—the triager then does not need to necessarily recreate the issue but can triage it anyway. Giving the right amount of details is non-trivial and the communication is lacking, because waiting for a response several days or weeks is normal.

At this point, it is apparent where open-source projects benefit. Since the hacker has access to the source code he can reference it directly. Or even better, actually provide a working fix through the means of a patch. The disclosure process can therefore be quicker. Worth mentioning are also GitHub's Security Advisories that allow collaboration of the maintainers and the reporter on the fix together in a private branch.

A hacker has to be careful about his expectations. The program owners can update the severity (e.g. lowering it from High to Low) or discard the report completely by marking it as Informative. Such reports do not result in any bounty, but at least do not harm the hacker's reputation. Also, the triage results vary across the programs. One of the vulnerabilities I have reported was triaged as High, Low or disregarded as Informative when reported to three different programs—one has to avoid frustration and understand that the program owner is in charge. Sometimes, the frustration pushes the hacker to try harder and escalate the previous issue into something more severe, consequently, forcing the

program owner to reconsider the previous decision.

If the report is triaged, all communication goes well and the bug is fixed one can expect to receive a bounty (after filling in the required paperwork on the platform). The time between submitting the report and the payout can be several weeks (though some programs pay an initial bounty right after the triage). The overall experience heavily depends on the program and can be a reason to stick to only a few programs or to move on to another one.

2.5 Is bug bounty hunting worth it?

There are several positives to bug bounty hunting. The hacker can choose her target. If there is a new technology that interests her, she can find a program that uses it and start learning. Looking for vulnerabilities gives a new perspective—quite different from the point of view of an ordinary developer. The goal is to find a way in, misuse anything that is available, and break things. There is no need to worry about the future deployment of the tools, and scripts that are created along the way. It is a technical and creative work that provides unbounded learning opportunities. Depending on a program the bounties can be of a significant amount. Also, some programs reward hackers with *a swag* or a Pro version of their services.

There are negatives as well. Staying *on toes*, and noting down any suspicious behaviour all the time is tiring after a while. Finding a good program that one enjoys exploring is not straightforward. Spending weeks diving into a piece of software and not finding anything is not an experience one can handle over and over again. A hacker is paid based on the value she brings to the program (higher severity bugs are rewarded a bigger bounty) not by hours—so, it is in the hands of the hacker, whether bug bounty hunting is sustainable long-term.

Looking from the other side, is it worth it for a company to have a bug bounty program? I can't comment on the experience of a program owner, especially concerning the time, effort and investment it takes to run it. I can comment as someone with an experience of trying to build something and also trying to break something. The inner incentives and motifs are quite different. The developer is driven to make *things work*. Finding an issue is a burden, not an achievement. Contrary, anything out of the ordinary fuels the hacker's curiosity. I can imagine a company simulating the bug bounty environment internally—reward tickets that report a security vulnerability, train developers in security and give them dedicated time to go and break the company's apps (preferably built by a different team). Also, a company can organize a hackathon, where the sole goal is to break its systems or create a team and participate in Capture the Flag style of events. That said, internal employees are easily in conflict of interests and therefore having a program hosted on a platform is a viable option.

However, the least a company can do is to clearly specify on the homepage or in the project's `README`, how is a hacker supposed to contact the owners in case they discover a security vulnerability. Surprisingly, even big projects are lacking security contact for reporting security vulnerabilities. Failing to provide such contact can either result in an unwanted public disclosure or no disclosure

at all.

Resources

The hacking community produces a lot of materials—reports, videos, podcasts or blog posts. The amount of content can be quite overwhelming, but reading and understanding the published reports and techniques is a a great way how to hone one's hacking skills.

- HackerOne hactivity: a list of all the disclosed reports on the platform, to narrow it down take a look at @vakzz's work on GitLab, which I might revisit during the talk,
- Capture the Flag writeups: another interesting list of techniques used to circumvent the security and exploit a vulnerability.
- John Hammond, LiveOverflow and STÖK: hackers, educators and content creators on various things related to hacking and bug hunting,
- Bug Bounty Reports explained: a YouTube channel concerned with explaining the disclosed reports,
- The Bug Hunter's Methodology and here: to keep up with growing number of programs one benefits from a better methodology,
- Bugcrowd's LevelUp series: online conference targeting hackers and security researchers,
- dayzerosec: a podcast discussing the latest disclosed reports from all the platforms.