The Art of Vulnerability Disclosure

François Martin

KARAKUN

François Martin

Senior Full Stack Software Engineer

francois.martin@karakun.com

martinfrancois

/in/françoismartin

@fmartin_

CVE-2018-1000529
CVE-2021-36460

Survey

- Who has found a vulnerability before?
- Who has disclosed a vulnerability before?
- Who has disclosed more than 5 vulnerabilities already?

Why disclose vulnerabilities?

- Allows vulnerabilities to be fixed before they are actively exploited
- Prevent malicious actors from using them to harm you and others
- Active contribution to making software more secure
- Increase trust in security by users
  - Users are usually not aware of what vulnerabilities are, but expect there to be none
- Good way of publicly demonstrating your knowledge and efforts in cybersecurity

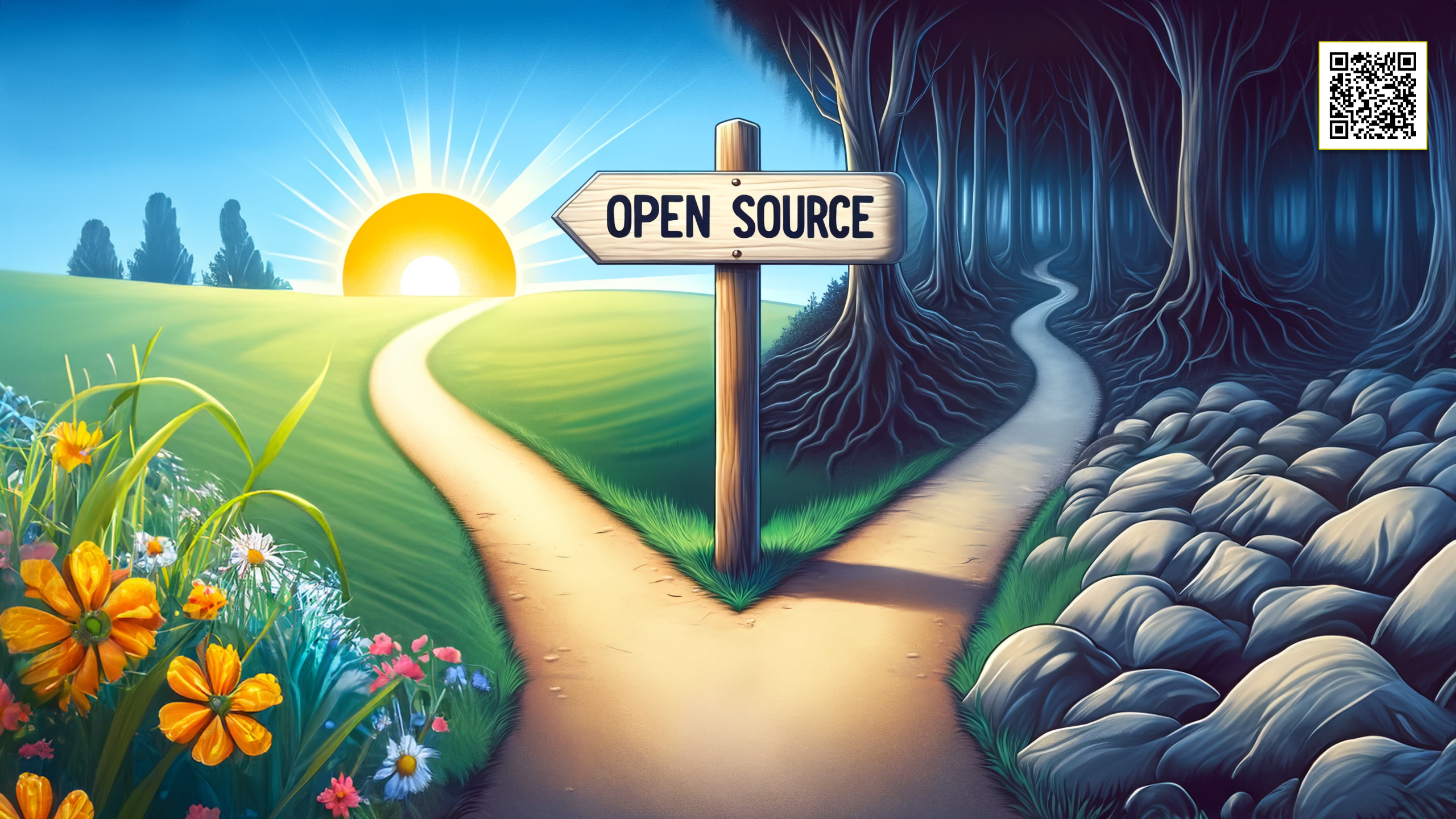
Weakness versus Vulnerability

Vulnerability

- Software code flaw or system misconfiguration, that allows attackers to **directly** gain **unauthorized** access to a system or network
- Example: SQL Injection
- Catalog of public vulnerabilities: **CVE** (**C**ommon **V**ulnerabilities and **E**xposures)

Weakness

- Problem reducing a system's security, even if it can't be exploited
- **Weaknesses can become vulnerabilities**
- Example: Using weak or insecure cryptographic algorithms (MD5, SHA-1)
- List of weaknesses: **CWE** (**C**ommon **W**eakness **E**numeration)


OPEN SOURCE

Disclosing to Open Source Projects

The easy way: using Snyk's vulnerability disclosure program

- Submit the vulnerability report to report@snyk.io or use Snyk's Vulnerability Disclosure form
- Minimum details to be included in the report
  - affected module
  - relevant package manager and ecosystem
  - vulnerability details
  - steps to reproduce
- Report can also be encrypted (recommended)
- Snyk takes care of the rest
  - Snyk responsibly discloses the vulnerability for you
  - You get full credit, including an assigned CVE

Methods of Disclosure

Private Disclosure

- Private report to organization
- Organization decides if and how to publish details
- Used in most bug bounty programs

Full Disclosure

- Full details made public as soon as identified
  - can include exploit code
- Often used when vulnerability is ignored to put pressure on vendor to fix it

Shortcomings

- If vendor does not respond or fix it, details may never be made public
- Companies trying to ignore / hide vulnerabilities often leads to Full Disclosure
- Allows attackers to leverage vulnerability before patch is available
- Controversial, considered irresponsible
  - **Only use as last resort, or when exploits are publicly available**

Methods of Disclosure

Responsible or Coordinated Disclosure

- Middle ground between private and full disclosure
- Initial report is private
- Full details published once patch is available
  - sometimes delayed to allow time for installing the patch
- Set deadline for organization to respond or provide a patch (recommended)
  - Example: Google Project Zero
  - full details published after 90 days, regardless of a patch being published
  - if patch published within 90 days, full details are published 30 days after patch is public
  - 7 days instead of 90 days with evidence of active exploitation against real users (”in the wild”)

Who to disclose to

- If the organization or vendor has a bug bounty program, use it and follow their guidelines
  - Carefully evaluate the guidelines before you publish anything yourself
- Check for a security.txt file at `/.well-known/security.txt` (RFC 9116)
- Check the company's website for links related to security
- Try generic email addresses like **security@**company.com or **abuse@**company.com
- Try contact pages on the website
- Send private messages on social media platforms the company uses
- Call the organization by phone
- Ask the community for known initial contacts at the company

Disclosure

First contact

- Make sure the contact is the correct person to disclose the vulnerability to
  - Example text: I discovered a security vulnerability in your app “SomeVulnerableApp”. Are you the right person to discuss the details? If not, please forward this email to the person with the correct responsibility.
- Preferably, suggest using encryption (using PGP / GPG)
  - Example text: If you wish me to disclose the details in encrypted form, please attach a public PGP key for me to encrypt my message with.
- Clarify your intentions
  - Example text: I do not demand payment, rewards, or other forms of compensation as a condition for responsibly disclosing the details of the discovered vulnerability to you.

Disclosure

Disclosing the initial report

- Provide enough details for the vulnerability to be understood and reproduced
- For projects that can be run locally, provide a minimal reproducible example (proof of concept)
- Include HTTP requests and responses, screenshots or any other supporting evidence
- Avoid using security terminology, explain in clear and simple terms
- Explain the impact of the vulnerability to clarify the priority
  - Example text: Versions <=3.1.2 of `thymeleaf-spring6` where `th:text` is used in an HTML template are affected. JavaScript code can be included in a specific URL which when opened is executed by the browser that could for example allow attackers to steal cookies.
- If possible, offer advice on how the issue could be mitigated or resolved
  - You can also refer to guides, such as an OWASP cheat sheet
- Clearly communicate deadlines (if any) after which you will publish full details

Publishing

- As soon as vulnerability was found, report the vulnerability to a CVE program participant (CNA)
  - Search for the affected (parent) company / product name in the list of partners
  - If you cannot find it, report it to MITRE CNA-LR
- The CNA will request a CVE ID
- CVE ID will be reserved for you (can take a while)
- Coordinate with the vendor when and which details to publish
- Publish the details, for example in GitHub repository
  - Examples: CVE-2018-1000529 and CVE-2021-36460
- When details of the vulnerability are published, request publication of the CVE ID
  - Include links to the published details
- NVD (National Vulnerability Database) by NIST analyzes the vulnerability and assigns CVSS score between 0.0 and 10.0 (the higher, the more severe)
  - Examples: Log4Shell vulnerability and stored XSS vulnerability

Timeline

- Strong recommendation: write down timeline **starting the moment you discovered** a vulnerability
- Include dates for the following:
  - Discovery
  - Communication sent and received
  - CVE request, assigning and publishing of the ID
  - Advisories published by the vendor
  - Fixes / releases by the vendor
- Provides insight into how quickly security issues are resolved
- Provides evidence of your unsuccessful communication attempts in case of an unresponsive vendor

Timeline

Example (from CVE-2018-1000529)

22nd of May 2018: Discovery and responsible disclosure of the vulnerability by @martinfrancois
24th of May 2018: Acknowledgement of the vulnerability and submission of CVE request
24th of May 2018: Pull request with fix for the vulnerability for Grails v3.x merged into grails-fields-plugin
24th of May 2018: Release of Grails Fields Plugin v2.2.8 for Grails v3.x
25th of May 2018: Pull request with fix for the vulnerability for Grails v2.x merged into grails-fields-plugin
25th of May 2018: Release of Grails Fields Plugin v1.6 for Grails v2.x
15th of June 2018: Release of Grails v3.3.6, including the updated dependency of the fixed Grails Fields plugin v2.2.8
22nd of June 2018: CVE-2018-1000529 assigned
26th of June 2018: CVE-2018-1000529 published

When things go wrong

- **Do not take it personally!**
- Vendor does not respond at all
  - Try contacting them again (for example: after 7 days, 30 days, …)
  - Try multiple different communication channels
- Vendor does not agree it is a vulnerability or does not (want to) fix it
  - Elaborate on the impact of the vulnerability, to ensure they understand the risk
  - Send links to examples, news reports, or statistics of similar vulnerabilities being exploited
  - Explain using likely end-to-end scenarios how the vulnerability could harm the company
  - Ask contact to escalate it to (senior) engineers, managers, or the CISO

When things go (really) wrong

- Full public disclosure
  - Could lead to negative reactions or even a lawsuit
- Anonymous disclosure
  - As you made contact already, it is obvious to the vendor who did it
  - Ensure you have taken sufficient operational security measures to protect yourself
- Report to third party
  - If vulnerability is severe, consider reporting to industry regulator or data protection authority
- Move on
  - **Unless the vulnerability is extremely serious, it is not worth risking your career or livelihood over an organization that does not care**

Looking back

Google Vulnerability Reward Program: 2023 Year in Review

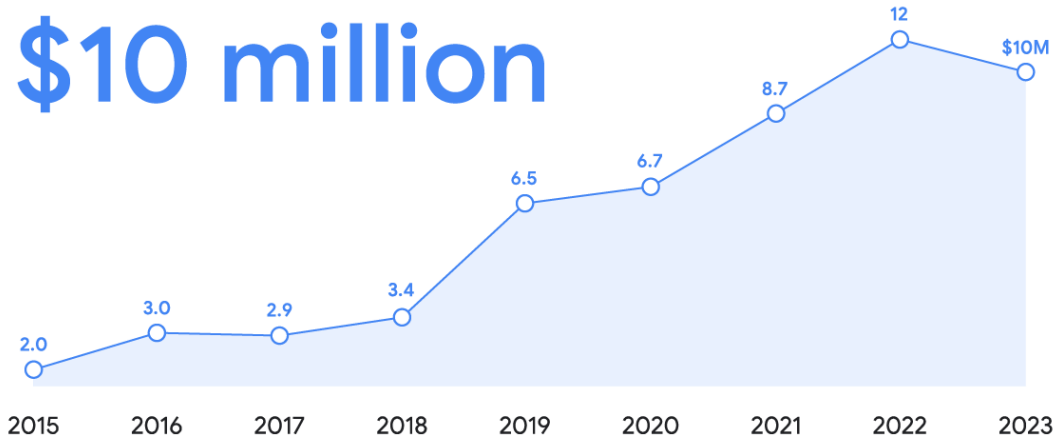
Highest reward
$113,337

Paid researchers
632

Countries represented
68

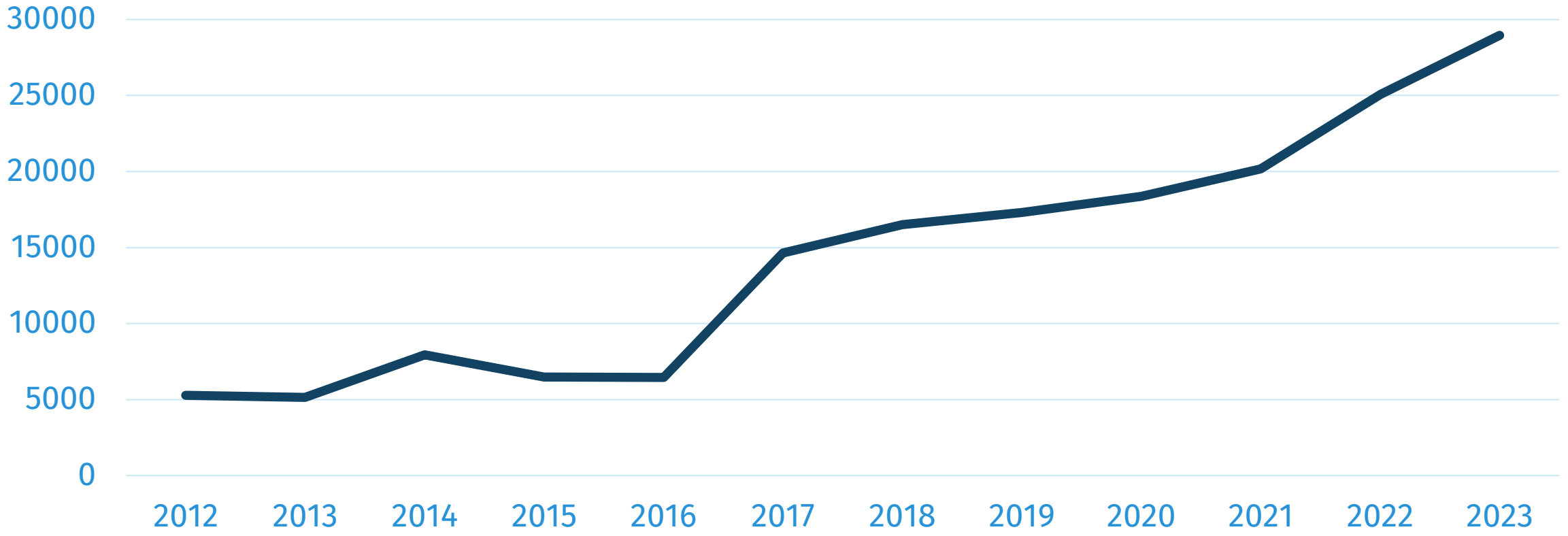
Total rewards since 2010 $59 million

Total rewards in 2023
$10 million



Looking back

Count of new published CVE records per year



Learn more

- OWASP Vulnerability Disclosure Cheat Sheet
- Snyk’s Vulnerability Disclosure Process
- Google Project Zero Vulnerability Disclosure Policy

Slides:

https://bit.ly/devnexus2024