TuxCare
We Take Care of Linux

The code is open

… but who is looking?

Open Source

- Open Source is a great:
  - Software development model
  - Business model
  - Work process
  - Sharing model
- Collaboration, team work
  - …and security?

Open Source code security – example 1/3

- curl/libcurl
  - Popular downloader, used everywhere, included everywhere
  - Created and managed by Daniel Stenberg
- Has been around for over 20 years
- CVE-2021-22898*
  - Issue around specific environment variables that could be used to exfiltrate data
  - Had been in the code base for 20 years at the time of disclosure
  - After disclosed, other vulnerabilities were found in the same part of the code
- "Ironically, the only way we know people are looking really hard at curl security is when someone reports security problems. If there was no reports at all, can we really be sure that people are scrutinizing the code the way we want?“ **

- * https://tuxcare.com/curls-20-year-old-bug-is-resilient-back-for-another-fix-cve-2021-22925/
- ** quote from Daniel Stenberg, at his blog https://daniel.haxx.se/blog/, "Increased CVE Activity in Curl?"

Open Source code security – example 2/3

- Log4j/Log4shell (CVE-2021-44228)
  - Logging facility in java, found to be remotely exploitable in December 2021
- Cloudflare disclosed some data on this security issue and identified exploit attempts **8 days prior** to public disclosure
- After public disclosure, the first exploit attempts happened within **9 minutes**

- https://blog.cloudflare.com/exploitation-of-cve-2021-44228-before-public-disclosure-and-evolution-of-waf-evasion-patterns/

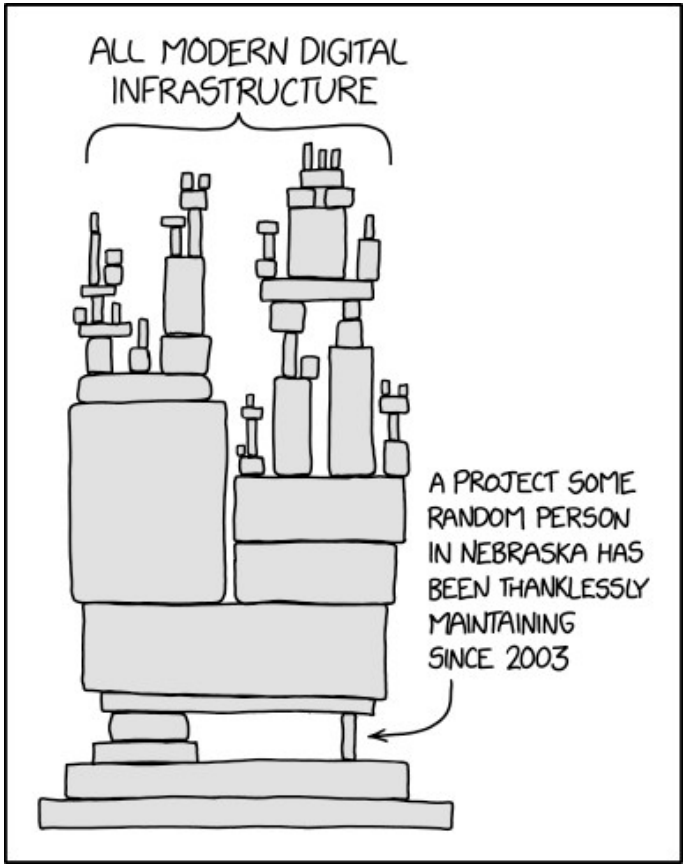
Open Source code security – example 3/3

- Python's tarfile package (CVE-2007-4559)
  - Specially crafted .tar files allow for arbitrary file overwrite
  - Reported in August 2007
  - "Fix" was the addition of a mention in the documentation about the dangers of extracting untrusted archives
  - "Rediscovered" in September 2022
- Surely, everyone read the documentation and avoided its use
  - …impacts over 60%of all github repositories that "import tarfile" (350 000 repositories)
  - Even github's copilot recommends its use
- Who was looking at the code in those 15 years?

- https://www.bleepingcomputer.com/news/security/unpatched-15-year-old-python-bug-allows-code-execution-in-350k-projects/

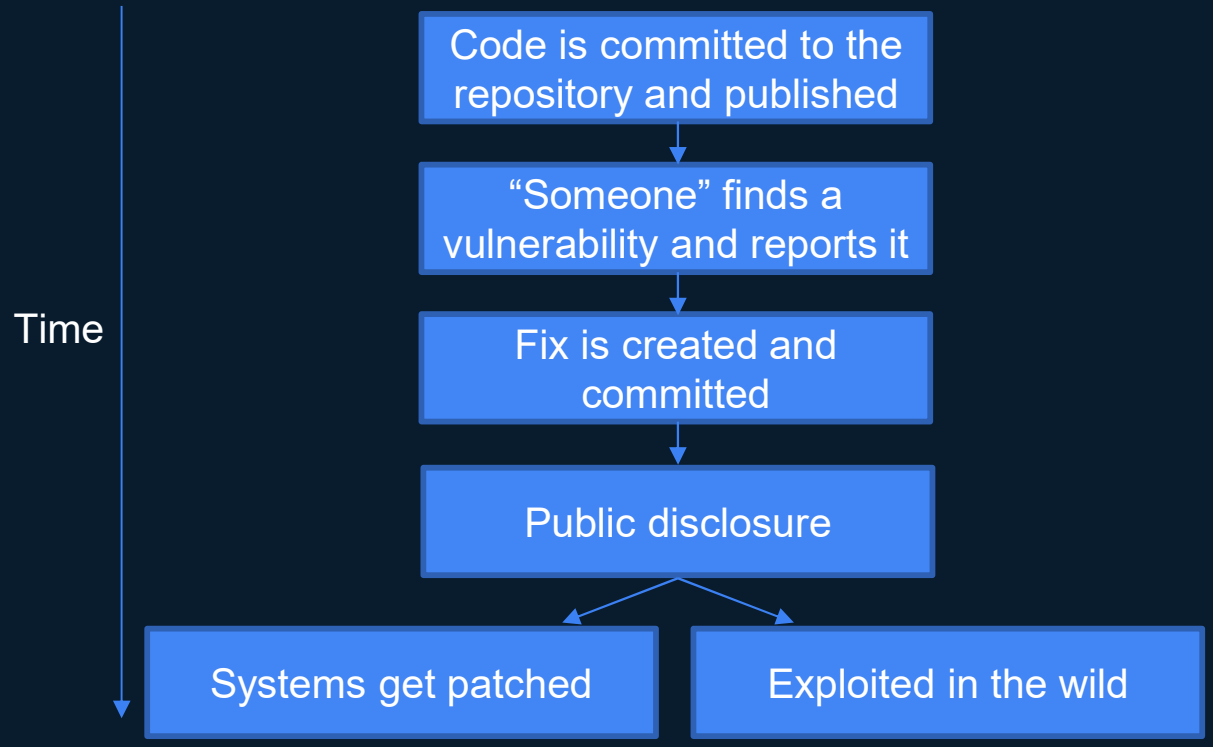
Open Source ecosystem

- This is the most accurate representation of the open source ecosystem
- Many projects have a “bus factor” of <=2 (1 or 2 persons responsible for the majority of the code commits) and/or very low activity
- But those are included in countless other projects as dependencies
- For the majority, there is hardly any code review, security audit or even peer review
  - The committer reviewing his own code does not add the desired value
- When you pull in a dependency into your project, or add a new tool into your workflow, you’re not getting just the functionality you want – you also inherit all the bugs too



- For example, http://oss.x-lab.info/github-insight-report-2020-en.pdf
- https://xkcd.com/2347/

Timeline of a vulnerability



Timeline of a vulnerability – the reality

Time

Code is committed to the repository and published

Threat actors checks the code for vulnerabilities

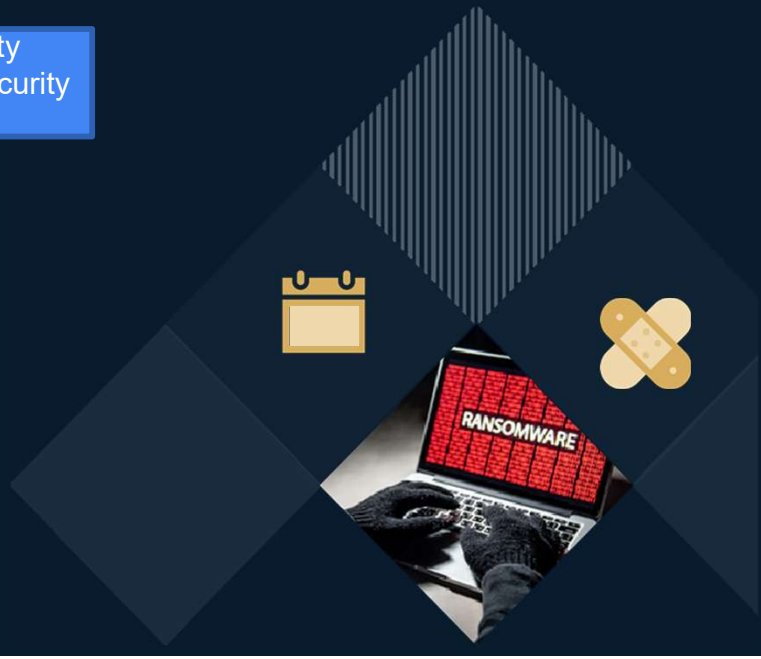
Do not report it

Hoard/use the vulnerability
Because it's unknown, no security tool will detect it

"Someone" finds a vulnerability and reports it

Fix is created and committed

Public disclosure



Open Source code security

- The Linux Kernel security team fixes security issues every week*
  - Most will never get a CVE
  - The main reason why you should always take new updates for the Kernel
- But the code is open
  - Someone with the right know-how and resource could check every commit and identify the code that fixes security issues
  - Only a very small subset of Kernel users will ever do this

- * https://www.youtube.com/watch?v=HeeoTE9jLjM “Kernel recipes 2019 – CVEs are dead, long live the CVE!” – Greg KH

The Threat Actor ecosystem

- Reporting a new issue or exploiting it for gain
  - Official bug bounties will often haggle the price (difficulty to exploit, scope, impact, take too long to acknowledge the issue)
  - Other threat actors will simply pay for the vulnerability
  - Unscrupulous companies pay millions for new vulnerabilities (for example, to break into locked phones)
- Complete economic model
  - People who find vulnerabilities
  - Exploit creators
  - Sellers/Resellers/Bundlers
  - Hack-as-a-service
  - “Tech support”
  - Ransomware creator toolkits

The Threat Actor ecosystem

- Much more efficient business model than bug bounty programs
  - To access, you need reputation
  - … or pay to play
  - Many new exploits always available for purchase
  - Threat actors can actually double dip – sell the initial exploit, then a couple of days later turn around and report the issue, earning from both ends
  - Highly incentivizes looking at code looking for new issues

Is Open Source less secure then?

- Not necessarily. But it is much more accessible when trying to identify new vulnerabilities
- Reverse engineering closed source software requires additional expertise
  - Reverse engineered code loses variable names, comments, context
- Open Source is the opposite – by design
- It doesn’t mean there are more or less bugs – it’s just easier to spot if you’re actively looking for them
  - And it’s easier for everyone, threat actors included

So, who is looking?

- Threat actors. Highly incentivized by large payments and very efficient economic model
- PhD students in cybersecurity-related degrees looking to get their names on a new CVE
- Security researchers doing pro-bono work on small projects
- Security researchers paid to work on large open source projects
- Intelligence agencies hoarding vulnerabilities

Key takeaways

- Open Source is not inherently more or less secure than closed source
- The best line of defense is still patching often
  - When a vulnerability is exploited within minutes of disclosure, waiting weeks until you patch is less than ideal
  - You will never know how long a vulnerability has been known to some threat actors before being publicly disclosed – if it ever is
- We need more incentives for code reviewers/code security on the white hat side of the fence. Or the other side wins by default
- Undisclosed vulnerabilities are not detected by your vulnerability management tools. A clean report means exactly that – it couldn't detect anything
- Keep good logs for as long possible. Keeping those logs forever is even better
  - When a new vulnerability is announced, have your vulnerability scanner do a pass on the logs. Discovering that your systems were breached 10 months ago* is a cybersecurity eye opener

- * https://www.upguard.com/blog/cost-of-data-breach

Key takeaways

- Verify the code you're bringing in
  - If you can't, at least validate the project's security posture:
    - Do they do code audits?
    - Is there a bug bounty program?
    - How are vulnerabilities handled?
    - How quickly are they handled?
    - Is the project transparent about security issues?
- If you have the resources and know-how, do a security audit of the code in-house and share the results with the project – it's open source after all, collaboration is a core aspect

TuxCare
We Take Care of Linux

Thank you!

Get in touch:

jcorreia@tuxcare.com

@jcorreiacl