## *Open Source License Choice and Vulnerability Disclosure Analysis*

*I choose the* ***Apache License 2.0*** *for my project. Apache-2.0 is a permissive license that imposes minimal obligations: it requires preserving copyright notices and provides an express patent grant to users. This means I can distribute modifications or larger works under any terms (including proprietary) without having to release source code, as long as I include the license notice. In contrast, the* ***GNU General Public License (GPL)*** *is a copyleft license requiring that any distributed derivative works be licensed under the GPL and accompanied by source code. Given the context of a proprietary tool, a strong copyleft license like GPL would force us to open source the entire project if I included GPL code, which is not aligned with the business goals. A permissive license (Apache-2.0 or MIT) avoids such obligations while still allowing open collaboration – I specifically favor Apache-2.0 for its added patent protection, which the simpler MIT License lacks, to safeguard both the project and its users from patent litigation risks.*

## *Initial Vulnerability Report Template*

*When reporting a security vulnerability, the initial report should contain key information to enable the product vendor to understand and act on the issue. The* ***vulnerability disclosure report template*** *includes the following fields:*

* ***Reporter Contact Information:*** *Name, affiliation (if any), and secure contact method (e.g. email with PGP). This allows the receiving organization to reach out for clarification or updates.*
* ***Product & Version:*** *The specific product (and subsystem if applicable) and version/build number affected. This identifies exactly what is vulnerable.*
* ***Environment Details:*** *Relevant environment/setup info (operating system, hardware, network configuration, etc.) where the issue was observed. This helps in reproducing the problem.*
* ***Vulnerability Summary:*** *A clear, reproducible description of the issue. Describe the weakness and how it can be triggered in normal terms, without providing exploit code. Include steps or inputs that reliably reproduce the bug (e.g. test case or URL) and any required conditions.*
* ***Impact:*** *The potential impact or severity of the vulnerability if exploited. For example, what an attacker could achieve – data theft, privilege escalation, denial of service, etc. – and the business/user impact (e.g. breach of privacy, financial loss).*
* ***Mitigation/Workaround (if known):*** *Any temporary fixes or workarounds the reporter is aware of that could reduce risk (for instance, disabling a feature or applying a configuration change). If the reporter has ideas on a proper fix, they can be noted here* ***without*** *providing actual exploit code.*
* ***Coordination and Timeline:*** *A proposed disclosure timeline and any coordination preferences. This includes the date of report, any suggested deadline for a fix before public disclosure (if following responsible disclosure), and availability for follow-up. It sets expectations for how long the reporter will wait for a response or patch before potentially escalating or publishing the issue.*

*(The above template ensures that the vendor receives all necessary details to verify and address the vulnerability, while avoiding inclusion of actual exploit scripts. It mirrors industry best practices for initial vulnerable reports.)*

## *Justification*

*Having a structured vulnerability reporting process is crucial. In fact, the U.S. Cybersecurity and Infrastructure Security Agency (CISA) has* ***mandated*** *federal agencies to implement Vulnerability Disclosure Policies (VDPs) that clearly define how to submit reports and what information to include. The template aligns with these best practices by requesting key details like description of the vulnerability, affected scope, impact, and reproduction steps. Providing such guidance to researchers upfront improves report quality and helps the organization triage and fix issues faster. Likewise, choosing an appropriate open source license (as discussed above) is backed by open source community recommendations – for example, resources like* ***Choose a License*** *emphasize that permissive licenses (MIT/Apache) grant broad reuse rights with minimal conditions, whereas copyleft licenses (GPL) carry the obligation to share source of derivative works. In summary, my decisions are informed by authoritative guidance to balance openness, legal obligations, and security.*

## *Reflection*

*This deep dive highlighted areas for refinement in the processes.* ***Next, I would focus on improving the open source compliance and disclosure practices.*** *For open source usage, implementing a rigorous license audit (e.g. using software composition analysis tools) before release would prevent incidents like inadvertent GPL code inclusion – this change engages developers and legal teams to proactively address licensing issues, ultimately protecting the company’s products from compliance risks. I would also refine the vulnerability disclosure workflow by defining clear internal roles and response timelines (e.g. initial acknowledgement within 3 days, fix within 90 days) to better set expectations for external researchers and internal stakeholders. These refinements benefit all stakeholders: developers gain clear guidelines (reducing legal/ethical uncertainty), the security research community sees a trustworthy partner (encouraging responsible reporting), customers get more secure and compliant products, and the organization avoids legal pitfalls while bolstering its reputation for integrity. In essence, continuously aligning the practices with ethical standards and industry benchmarks will lead to more sustainable and positive outcomes for everyone involved.*

## *GPL Case Brief: Incorporating GPL Code into a Proprietary Tool*

### *Facts*

* ***Inclusion of GPL Code:*** *The company’s proprietary tool (closed-source software) was found to contain a third-party component licensed under the GNU GPL. The GPL-licensed code was incorporated (likely via a library or code snippet) into the tool’s codebase.*
* ***Distribution to Customers:*** *The proprietary tool, including the GPL component, was shipped to multiple customers. This external distribution triggers the GPL’s requirements, which would obligate providing the corresponding source code to recipients under the same license.*
* ***Discovery of Violation:*** *The use of GPL code was initially overlooked. It came to light either through an internal review or an external inquiry. No GPL license notice or source offer was provided with the product, meaning I were (unintentionally) out of compliance with the GPL terms.*
* ***Immediate Risk:*** *Once discovered, the company faced significant legal and ethical risk. Continuing to distribute the tool without addressing the GPL obligations could constitute a license violation. This exposes the company to potential legal action by the copyright holders or enforcement bodies, and reputational damage for not respecting open source licenses.*

### *Issues*

* ***Does distributing the proprietary tool with embedded GPL code require us to release the tool’s source code under GPL?*** *– (At issue is whether the proprietary software is now a derivative work of the GPL component, thus invoking GPL’s copyleft requirement that the entire combined work be licensed under GPL and source made available to users.)*
* ***What are the obligations to users and the open source copyright holder now that GPL-licensed code was used?*** *– (Must I provide attribution and offer the complete corresponding source to all recipients? Have I effectively lost the option to keep the code proprietary unless I remove the GPL parts?)*
* ***How can I become compliant, and what are the consequences if I do not?*** *– (Do I need to cease distribution until fixed, replace or remove the GPL code, or open source the whole product? What legal actions or ethical repercussions might result from non-compliance?)*

### *Ethical Analysis*

*From an ethical standpoint, using GPL code in a proprietary product without honoring the license terms violates principles of honesty and fairness.* ***Deontological ethics (duty-based)*** *would assert that I have a duty to uphold the license agreement – when I chose to use GPL software, I implicitly promised to abide by its terms. Failing to release the source is essentially breaking a promise to the open source community and the original authors. This conflicts with professional codes of ethics: for example, the ACM Software Engineering Code of Ethics calls on engineers to “honor property rights including copyrights and patents” and to give proper credit for others’ intellectual property. By not complying, I also undermine the spirit of open source collaboration, which* ***virtue ethics*** *would view as demonstrating poor character (not acting as a responsible, respectful member of the developer community).* ***Utilitarian analysis*** *likewise shows it’s unethical to ignore GPL obligations – such an action can harm the open source ecosystem (discouraging authors from sharing code if their terms are ignored) and could harm the users or company if legal action or loss of trust occurs. In short, the ethical course is to promptly come into compliance, respecting the license and the intent of the GPL to share improvements with the public.*

### *Legal/Policy Analysis*

*Legally, the GPL is a binding license agreement under copyright law. Incorporating GPL code into the tool and then distributing it means the software is considered a “work based on” the GPL component (a combined or derivative work).* ***GPLv3 (and GPLv2) require that if any part of a program is GPL-licensed, the entire derivative work must be licensed under GPL and the complete corresponding source code offered to recipients****. In practical terms, I were obligated at distribution time to provide the customers with the GPL component’s source and the proprietary tool’s source (since it’s now a single combined work) under GPL. Because I did not do this, I’m in violation of the license. The GPL license enforcement has teeth: the copyright holder can suspend the rights to use/distribute the software and pursue legal remedies. In fact, GPL violations have led to lawsuits, settlements, and injunctions in several jurisdictions. For example, companies have been sued for not releasing source after shipping GPL software, facing significant damages. The company’s failure to comply not only breaks the license agreement but also exposes it to these legal consequences. From a policy perspective, this situation reveals a gap in the software supply chain governance. I lacked an effective open source license compliance policy. Moving forward, I need to implement stricter controls (e.g. inventory of open source components and automated license scanning) as part of the software development policy to prevent such violations. Open source license compliance is not optional – “open source is free, but that doesn’t mean it doesn’t require compliance… companies must put proper policies in place and be vigilant in adhering to them”. In summary, legally I must either cease the violation or face penalties, and organizational policy must be updated to treat license obligations with the same seriousness as proprietary software contracts.*

### *Decision & Rationale*

***Decision:*** *I will* ***remediate the GPL compliance issue by removing or replacing the GPL-licensed code in the proprietary tool, and by providing proper notice to stakeholders.*** *Concretely, the plan is to immediately stop shipping the non-compliant product and engineer a solution that purges the GPL component. I will either rewrite that functionality in-house or substitute an alternative library under a permissive license (MIT, Apache 2.0, etc.) that does not impose copyleft. Once the replacement is ready, I will issue an update to all customers. This approach allows us to honor the GPL (by no longer violating its terms) without having to open source the entire proprietary codebase. I chose this path after considering the alternatives (detailed below) and weighing their impacts.*

***Rationale:*** *Removing the GPL code and using a permissively licensed (or proprietary) alternative lets us become compliant quickly while preserving the business value of the software. Relicensing the entire product under GPL, though legally viable to comply, would have meant forfeiting proprietary control – a step the company isn’t willing to take given the business model. By contrast, replacing the component confines the open source obligation to that component alone (or eliminates it entirely), restoring the freedom to keep the rest of the source closed. This decision also aligns with the ethical obligation to respect open source licenses: I are correcting the mistake without further harm. In tandem, I will craft an open source usage policy (and employee training) to ensure* ***due diligence in license checking*** *going forward. In doing so, I acknowledge the importance of complying with open source licenses to avoid legal risks and to uphold the reputation; indeed, cases have shown that ignoring open source license rules can become extremely costly. The chosen remediation not only fixes the current violation but also helps restore trust with the open source community and the customers by demonstrating that I take license obligations seriously.*

### *Alternatives Considered*

* ***Open Sourcing the Entire Tool under GPL:*** *I considered fulfilling the GPL terms by releasing the whole proprietary codebase as open source (under GPL). This would immediately satisfy the license requirements and avoid violation. However, this alternative was rejected because it would irreversibly expose the proprietary business logic and potentially erode the competitive advantage. The company determined this impact was too detrimental to pursue except as a last resort.*
* ***Negotiating a License Exception or Dual License:*** *Another idea was to contact the copyright owner of the GPL component to seek a special permission (e.g. a commercial license or exception) allowing use in the closed-source product. In some cases, GPL projects offer dual licensing or can grant exceptions for a fee. This was deemed impractical, as it’s uncertain such permission would be granted (and could be costly and time-consuming to arrange). There is no guarantee the authors would relicense their code for me.*
* ***Maintain Status Quo (Do Nothing):*** *I briefly discussed taking no action and continuing to ship the product as-is. This would avoid immediate engineering cost. This option was quickly discarded on ethical and legal grounds. Intentionally flouting the GPL would violate the values and almost certainly lead to legal action or community backlash. The potential consequences (lawsuits, injunctions, damages, and loss of goodwill) far outweigh the short-term convenience.*
* ***Remove Feature Entirely:*** *As a variant of replacing the GPL component, I considered dropping the feature/functionality provided by that component from the tool. This would eliminate the need for that code. This approach was feasible technically, but it would reduce product functionality for users. I decided that if a suitable replacement could be found or built quickly, that would be better than depriving users of a feature. I kept this as a fallback in case a replacement library was not available in time.*
* ***Delay and Monitor:*** *Another alternative was to temporarily halt new distributions and quietly monitor if any complaint or enforcement action arises, essentially buying time. This passive approach was not acceptable, as it still leaves current customers with an undisclosed GPL violation and does not actually resolve the issue. It would also be unethical to hide the violation; proactive disclosure and resolution is the more responsible path.*

*(After evaluating all options, the replacement strategy was chosen as the most balanced solution to achieve compliance with minimal negative impact.)*

## *AI Use Note*

*(Assistance from AI was used in preparing this report.) In the course of researching license specifics and best practices, an AI language model (ChatGPT) was utilized to gather information and suggest phrasing. All AI-provided content was carefully reviewed, fact-checked against official sources, and edited to ensure accuracy, relevance, and compliance with academic standards. The final content, structure, and analysis presented here are my own, with the AI serving as a research aid.*