The Ethics of Cybersecurity Vulnerability Disclosure

Daniel C. DiPietro Jr.

University of Rhode Island

CSC120G: The Impacts of Tech on Society

Professor Armenti

9 December 2025

Introduction

A cybersecurity analyst discovers a critical vulnerability in a widely used hospital software system and fears that attackers may eventually find it too. Should the analyst disclose the vulnerability quickly, wait for the system's engineers to patch it, or just keep it a secret? This report covers the ethical dilemma of vulnerability disclosure, analyzing its background, methods, and ethics. A **software vulnerability** is defined in ISO/IEC 29147 as functional behavior in a product or service that violates an implicit or explicit security policy (ISO/IEC 29147, 2018, sec. 3.1). **Vulnerability disclosure** refers to the act of providing previously unknown vulnerability information to a party not believed to be aware of it (ISO/IEC 29147, 2018, sec. 3.2).

Before approaching disclosure of the vulnerability, there is tension between protecting the software's users quickly, giving vendors time to patch the flaw, and governments wanting offensive cyber capabilities. This tension is the essence of the ethical dilemma, and formal frameworks now exist to manage it. The purpose of this research paper is to propose the most ethical way to handle cybersecurity vulnerabilities to best protect users while treating researchers and vendors fairly.

Background

Software vulnerabilities are unintentional flaws in code that can result in unauthorized access, data exposure, or system corruption, and vulnerability disclosure is the process of letting that vulnerability be known to any party currently unaware of it (ISO/IEC 29147, 2018, sec. 3.1-3.2). It is important to have standardized definitions of these terms to avoid any confusion during the disclosure process, especially when researchers, vendors, and other stakeholders must coordinate their responses.

Vulnerability disclosure exists to protect users, reduce security risk, and give vendors the information needed to fix flaws before they are exploited. A **vendor** is the company or organization that created, maintains, or is responsible for the software containing the vulnerability. A structured framework is necessary for any organization with cyber infrastructure because it provides a consistent process for receiving, analyzing, and resolving vulnerability information. Rather than reports being handled informally or inconsistently, disclosure frameworks route the information through a structured process, ensuring it reaches the appropriate parties first and that remediation begins as quickly as possible.

There are three main approaches to handling vulnerability information. **Full disclosure** means that the researcher publicly releases the vulnerability details immediately or very quickly, usually before a patch exists. This approach is often used when vendors ignore the researcher or take too long to fix the issue, there is public pressure to respond quickly to the vulnerability, or for defenders to begin monitoring their systems instantly. However, this also means that attackers

learn of the vulnerability at the same time as everyone else, and if no fix exists yet, then attackers may exploit it first (OWASP Foundation, n.d.).

Coordinated disclosure is the most commonly recommended approach and involves the researcher privately reporting the vulnerability to the vendor, allowing them time to create and test a patch before anything is released publicly. This model is designed to manage risk while disclosing enough information for users to protect themselves. Organizations like NIST describe coordinated disclosure as the best option that best balances user safety, vendor responsibility, and long-term transparency (Schaffer et al., 2023).

A third approach to handling vulnerabilities is **zero-day retention**, where a vulnerability is intentionally kept undisclosed so it can be used for intelligence or offensive cyber operations. Instead of alerting the vendor, the information is withheld, allowing the discoverers to exploit the flaw before anyone else knows it exists. In the United States, this process is managed under the **Vulnerabilities Equities Process** (VEP), a federal framework that evaluates whether retaining or disclosing vulnerability better serves national interests (White House, 2017).

Case Study

Log4Shell (CVE-2021-44228) is a **remote code execution** (RCE) vulnerability, meaning attackers can use it to run their own code on a victim's machine without authentication. It was discovered in **Apache Log4j 2**, a Java logging library used in millions of applications. Log4j appears in many common systems, including Minecraft servers, cloud services, enterprise software, and government systems. The exploit is triggered when an attacker causes the system to log a specially crafted string using a **Java Naming and Directory Interface** (JNDI) lookup

(Cloudflare, 2021). Log4j allowed user-controlled text to trigger a JNDI lookup, and when Log4j logged a piece of malicious text, it would contact the attacker's server, the attacker's server would respond with malicious Java code, and Log4j automatically loaded and executed the attacker's code without authorization or verification. This process is called **JNDI Injection**. Log4Shell was rated CVSS 10.0, meaning maximum severity and ease of exploitation, placing it in the top tier of dangerous vulnerabilities.

On November 24, 2021, the vulnerability was privately reported to Apache by Alibaba Cloud Security Team. Apache developers began working on a fix privately (coordinated disclosure phase). Then, on December 9, 2021, the public exploit PoC appeared online, and Log4Shell became widely known. On December 10, 2021, Apache released Log4j 2.15.0, the first patch of the exploit. The vulnerability became publicly disclosed at this time due to the exploit circulating. Subsequent issues were discovered, which led to later versions of the patch (i.e., 2.16.0, 2.17.0, and so on). Multiple follow-up vulnerabilities appeared because the initial patches didn't fully address all attack vectors. The disclosure-to-exploitation window was extremely short, as scanning began within hours (Cloudflare, 2021).

The impact of Log4Shell was widespread and severe (CISA, 2022). Log4j was used in millions of products, many of which vendors were unaware of its presence. Cloud providers like AWS, Google Cloud, Steam, and Apple iCloud were all impacted. Attackers immediately began scanning the entire internet, and companies like Recorded Future and Cloudflare saw massive exploit attempts within twenty-four hours (CISA, 2022). The Cybersecurity and Infrastructure Security Agency (CISA) issued an emergency directive that urged all federal agencies to patch

the vulnerability immediately (CISA, 2022). The vulnerability facilitated the creation of cryptomining malware, botnet recruitment, and ransomware. Log4j was hard to patch because it was included deep inside other software, and many systems remained vulnerable for months after its discovery.

This case shows how coordinated disclosure was attempted, as the initial report was kept private. However, since the exploit was leaked early, public disclosure happened before full patches were ready. Log4Shell also demonstrates the challenge of huge dependency chains and slow vendor response. It also highlights how disclosure timing affects defenders who need time, attackers who exploit early, and vendors who must apply patches fast. Overall, it serves as an example of how even "ideal" coordinated disclosure can break down in a real-world context.

Ethical Analysis

Utilitarian and Kantian perspectives will be applied to ethically analyze the vulnerability disclosure process and the Log4Shell case study. To begin, the disclosure process will be examined through a utilitarian lens, measuring total benefits versus total harm (Deaton, 2013). Full disclosure gives defenders instant information, but also gives attackers immediate access. This can result in significant short-term harm, as users may be exposed to the vulnerability before a patch is available. Log4Shell showed this when exploit code was leaked early, as it led to global exploitation within hours.

Coordinated disclosure gives vendors time to prepare secure patches and reduces total harm. Public disclosure still happens, but the users are eventually protected. It is also backed by NIST and ISO for reducing population-level risk. Zero-day retention can benefit intelligence

operations under certain circumstances, but leaves the public at risk for long periods, as no patch is immediately introduced. The vulnerability can lead to high potential harm if independently discovered by attackers. The utilitarian conclusion is that coordinated disclosure produces the lowest overall harm, while full disclosure and zero-day retention both create unnecessary risks. Log4Shell demonstrated what happens when coordinated disclosure breaks down.

On to the Kantian perspective, which holds that people must act from duty and not convenience (Deaton, 2013). Vendors have a duty to protect users and fix vulnerabilities quickly, and failing to patch or communicate violates the duty to treat users as ends in themselves. Researchers have a duty not to cause preventable harm, and reckless full disclosure treats users as a means, risking collateral damage. Kantian ethics requires responsible reporting that could be universalized as a rule. Users have a right to security and to know when their systems or products are at risk. Retaining vulnerability information violates users' autonomy and right to informed decision-making. Zero-day retention knowingly leaves users exposed, which violates a government's duty to protect its citizens. Government benefits do not justify using people as a means to its goals, and Kantianism rejects withholding patches from the public, as it cannot be universalized. Thus, a Kantian conclusion is that coordinated disclosure best aligns with duties and user rights, and zero-day retention is least compatible with this perspective.

Applying ethics to Log4Shell shows that it initially abided by the earlier utilitarian and Kantian conclusions, as it was privately reported to Apache on November 24, 2021, following coordinated disclosure principles. However, the breakdown occurred when the exploit was leaked early on December 9th, leading to full disclosure before widespread patching. The

utilitarian impact of Log4Shell is shown through the resulting massive exploitation, while millions of systems were left vulnerable because patches couldn't be applied instantly, increasing total harm beyond measure (Deaton, 2013). The early leak of information violated Kantian duty not to expose people to avoidable harm, as users became collateral damage and were treated as a means. The ethical takeaway from this section is that Log4Shell shows how disclosure timing can create real ethical consequences, and coordinated disclosure should be done correctly and consistently.

Proposed Solution

The default approach should be coordinated disclosure with a clear deadline (e.g., 60-90 days). Vendors should receive private notice first, but if they are too slow to respond, then disclosure must still occur. Short timelines prevent suppression of the vulnerability while giving enough time for vendors to prepare. Vendors should also maintain a public and easy-to-use vulnerability reporting system, and must respond promptly to researchers and begin patch development directly (Schaffer et al., 2023). They should also publish **security advisories** (i.e., a communication that provides information about potential security issues, vulnerabilities, or changes that may affect an organization's security) once a patch is ready or the deadline expires (Schaffer et al., 2023). These advisories should include clear severity, risk, patch instructions, and current known exploitation status (Schaffer et al., 2023). These advisories reduce confusion for both users of the vulnerable software and its relevant organizations.

Researchers play a significant role in the disclosure process and should be legally protected from retaliation if they handle the vulnerability information in good faith. They should be protected from lawsuits when reporting responsibly, as this encourages responsible disclosure

instead of underground or anonymous releases. These protections would also reduce the risk of accidental full disclosure caused by fear of vendor backlash.

For governments interested in retaining zero-day exploits, there should be heavy regulation and restriction due to the risks involved. Vulnerabilities should only be retained when the national security value is extremely high. The retention should also be reviewed regularly (for example, on a monthly basis) and time-limited, to ensure that vendors will be able to patch the exploit as soon as possible. Agencies should justify retention using a formal process like the VEP (The White House, 2017). These suggestions acknowledge government benefits of zero-day retention while reducing public risk. Overall, coordinated disclosure with clear deadlines, researcher safety, vendor duties, and restricted retention creates the most ethical and practical disclosure process. The proposed solution aligns with utilitarianism and Kantianism as it reduces total harm while respecting rights and duties (Deaton, 2013).

Conclusion

Responsible vulnerability disclosure strikes a balance between user safety, vendor requirements, and government interests as the timing and scale of disclosure ultimately determine whether users are protected or exposed. Full disclosure creates high short-term risk because attackers gain information before patches exist. Zero-day retention benefits niche government objectives but leaves the public vulnerable to attacks, and coordinated disclosure minimizes total harm and respects stakeholder duties. The case study Log4Shell showed how disclosure breakdowns can cause real-world harm, and the early exploit leakage illustrated why structured disclosure processes matter. Both utilitarianism and Kantian ethics support coordinated disclosure because it reduces overall risk while respecting users' rights and avoiding

treating them as a means. Overall, a structured, coordinated disclosure model with clear deadlines is the most ethical and practical method for handling vulnerability information. Effective frameworks for disclosure strengthen cybersecurity for both individuals and society as technology continues to evolve.

References

CISA. (2022). *2021 top routinely exploited vulnerabilities*. Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-117a

Cloudflare. (2021). *Inside the LOG4J2 vulnerability (CVE-2021-44228)*. The Cloudflare Blog. https://blog.cloudflare.com/inside-the-log4j2-vulnerability-cve-2021-44228

Deaton, M. (2013). *Ethics in a nutshell: An intro for ethics bowlers* (3rd ed.). EthicsBowl.org. https://www.ethicsbowl.org/wp-content/uploads/2019/02/Ethics-in-a-Nutshell-for-Ethics-Bowlers-3rd-Edition.pdf

ISO. (2018). *ISO/IEC 29147:2018 Information technology — Security techniques — Vulnerability disclosure*. International Organization for Standardization. https://www.iso.org/standard/72311.html

OWASP. (n.d.). *Vulnerability disclosure cheat sheet*. OWASP Foundation. https://cheatsheetseries.owasp.org/cheatsheets/Vulnerability_Disclosure_Cheat_Sheet.html

Schaffer, K., Mell, P., Trinh, H., & Van Wyk, I. (2023, May 24). *Recommendations for federal vulnerability disclosure guidelines.* CSRC. https://csrc.nist.gov/pubs/sp/800/216/final

The White House. (2017). *Vulnerabilities Equities Policy and Process for the United States government.* National Security Archive. https://nsarchive.gwu.edu/document/17688-white-house-vulnerabilities-equities-policy