by Dinis Cruz and ChatGPT Deep Research and Claude 3.7, 2025/02/10

Second Stories: From Three Mile Island to Cybersecurity

by Dinis Cruz and ChatGPT Pro Deep Research, 10 Feb 2025

Introduction

The concept of "second stories" originates from safety science and the analysis of accidents such as the Three Mile Island (TMI) nuclear incident. In the aftermath of TMI (1979), investigators realized that focusing on who to blame (a **first story**) was less useful than understanding **what** went wrong in the system (the **second story**) (Who Destroyed Three Mile Island? - Ruby Video).

A first story might pin the accident on an operator's mistake, but a second story reveals the broader context – e.g. how confusing instrument design, inadequate training, and other systemic factors set the stage for the error.

In fact, the official TMI review found **"a combination of equipment malfunctions, design-related problems and worker errors"** led to the partial meltdown (Backgrounder On The Three Mile Island Accident | NRC.gov).

Instead of just blaming individuals, the second-story perspective highlights how the system's constraints and flaws influenced human decisions.

As Nickolas Means explains (drawing on Sidney Dekker's work), uncovering these systemic, contextual reasons behind failures leads to a richer understanding of an incident.

In short, *second stories* shift the narrative from individual blame to the systemic conditions that make failures more likely.

Application to Cybersecurity

Just as in engineering accidents, cybersecurity failures often stem from underlying constraints and system flaws rather than one person's mistake. Modern IT environments are **complex, tightly-coupled systems**, so a single misstep (like a misconfiguration or a missed patch) can trigger a breach only when multiple defensive layers and processes have weaknesses. Blaming an individual user or developer for a security incident ("someone clicked a bad link" or "the admin misconfigured a server") is a first-story view.

The second story asks: *What allowed that action to result in a breach?* Often, we find constraints such as insufficient tools, process gaps, or organizational issues at play. For example, a developer might introduce a vulnerability due to deadline pressure and lack of security training – a reflection of management priorities and siloed team structure, not personal negligence. Similarly, an analyst might miss an alert because the **security monitoring systems overwhelm staff with noise**, or because teams are understaffed due to budget limits. These are systemic problems, not individual failings.

Common systemic constraints in cybersecurity include outdated or complex tooling, lack of cross-team communication, unclear accountability, and resource limitations.

All of these can create latent vulnerabilities. A striking real-world parallel comes from a recent NSA/CISA analysis: many organizations share the **same misconfigurations and weaknesses**, suggesting these issues are *"systemic weaknesses across many networks"* rather than one-off human errors (NSA and CISA Red and Blue Teams Share Top Ten Cybersecurity ...).

In other words, the *conditions* in many IT environments (legacy software, poor default settings, etc.) consistently set people up to make similar mistakes.

Much like the engineering decisions and design flaws that predated the TMI accident, factors such as technical debt, rushed deployments, or siloed decision-making can predispose an organization to cybersecurity failure.

By examining these contextual factors, security teams can better understand how an incident happened despite everyone's best intentions, mirroring the second-story approach used in safety engineering.

Case Studies: "Second Stories" in Cybersecurity Incidents

Real cybersecurity incidents illustrate how focusing on systemic causes yields deeper lessons than a blame game. Below are a few notable breaches where the **"second story"** reveals underlying failures, and how a blame-oriented view initially obscured those issues:

- **Equifax Data Breach (2017):** In one of the largest breaches on record, attackers exploited an unpatched web server vulnerability. The *first story* told by Equifax's leadership blamed the incident on a single technician who failed to apply a software patch. Equifax's ex-CEO even testified that **"a single employee's failure to heed a security alert"** led to the missed patch (Equifax Ex-CEO Blames One Employee For Patch Failures). This blame narrative cast the breach as one person's mistake. However, the *second story* (uncovered by investigators and a Congressional report) showed a pattern of systemic failings. Equifax lacked an effective inventory of applications, had an expired SSL certificate that left its intrusion detection blind, and had no clear accountability for patch management (House committee says Equifax data breach was 'entirely preventable'). In fact, lawmakers concluded the breach was **"entirely preventable"** if not for **"widespread, systemic flaws"** in Equifax's security program. The second story here shifts focus from one employee to the company's inadequate processes and culture – for example, why a critical patch alert could be missed without any fail-safe. It became clear that if Equifax had stronger controls (like automated asset scanning, better oversight, and a culture of proactive security), one forgotten patch would not have led to a disaster.
- **SolarWinds Supply-Chain Attack (2020):** The SolarWinds incident was a sophisticated supply-chain compromise, where attackers injected malware into the company's software updates, affecting thousands of organizations. Nevertheless, early public discussion latched onto a simple culprit: a weak password. SolarWinds executives revealed that an intern had set the password "solarwinds123" on an internal system and accidentally exposed it on GitHub – an error they claimed went undetected for years (SolarWinds blames intern for weak 'solarwinds123' password - ITPro). This disclosure led to headlines essentially blaming an intern for the hack. That is the first story. The second story digs deeper: **why** could attackers escalate from a leaked password to inserting code into the build pipeline? The true causes include broader systemic issues like insufficient code review and build security, lack of egress restrictions, and poor credential management practices. In other words, *organizational and process failures* made it possible for a single password mistake to persist and be so damaging. By focusing only on the intern's error, SolarWinds' leadership risked ignoring more fundamental lapses (such as not enforcing password policies or monitoring critical systems). The second story of SolarWinds is a cautionary tale that robust security can't hinge on never making a mistake – mistakes *will* happen, so systems must be designed to be resilient when they do.

- **Target Retail Breach (2013):** Attackers infiltrated Target's network via credentials stolen from a third-party HVAC contractor, then installed malware on point-of-sale systems, stealing 40 million credit card numbers. The simplistic first story blames the vendor's employee for falling for phishing, or Target's security team for *"ignoring alerts."* Indeed, Target had deployed advanced malware detection that did trigger warnings – which operators initially **reviewed and then dismissed** (Target Ignored Data Breach Alarms - Dark Reading). It's easy to point a finger at the analysts who "ignored" those alerts. The second story, however, asks why the alerts were missed. In Target's case, the alert fatigue and unclear escalation procedures played a role – the security team was receiving a high volume of alerts and the true positive got lost among false positives. Additionally, network segmentation was insufficient; once the contractor's access was compromised, attackers could traverse to sensitive systems, which reflects an architectural weakness. The **systemic issues** included an overtaxed security operations center, possible communication gaps between the security tool and incident response processes, and design flaws in network access controls. These factors collectively allowed a phish and a piece of malware to cause massive damage. The lesson is that improving those underlying systems (tuning alerting systems, training staff, segregating networks) is far more effective than blaming an analyst or an unlucky contractor.

In each of these cases, a blame-focused view targeted an individual or single point of failure, while the second story exposed a chain of organizational, technical, and process issues. When companies fixate on **"who clicked the link"** or **"who misconfigured that server,"** they often miss the opportunity to fix the conditions that permitted a single slip to escalate into a breach. By contrast, investigating the broader context – how policies, tools, and cultural factors contributed – leads to more meaningful security improvements.

Frameworks and Strategies for Second-Story Analysis

Adopting a second-stories approach in cybersecurity requires intentional changes in how we investigate incidents and manage security culture. Below are frameworks and strategies to help security teams move from a blame-oriented mindset to systemic problem-solving:

- **Blameless Post-Mortems:** After an incident, conduct a *blameless postmortem* to document what happened and why, without assigning personal blame. The goal is to create a safe space for truth-telling about all contributors to the incident. Teams like Google and Atlassian emphasize identifying **all contributing root causes** and learning from them (How to set up and run an incident postmortem meeting - Atlassian). Concretely, this means an incident report should ask *"How did our defenses and processes fail?"* rather than *"Who screwed up?"* For example, if a critical server was misconfigured, the blameless analysis might reveal that documentation was unclear or the change management process was rushed. A

second story investigation explicitly looks for *"what allowed that human error to have a negative effect"* (Complete Blameless Post-mortem Guide | Smartsheet). By focusing on how the system can be improved (e.g. better automation to catch misconfigurations), the post-mortem turns an incident into actionable lessons.

- **Ask "What" Instead of "Who":** A practical technique during incident analysis is to phrase questions that uncover conditions. Rather than asking *"Who was responsible for X?"*, ask *"What factors led to X happening?"* This is aligned with Sidney Dekker's guidance to find out **"why it made sense"** for people to behave as they did given the information and tools they had. For instance, if a developer introduced a security bug, investigate what constraints (time pressure, lack of checks, insufficient training) existed. One method is to use the **"Five Whys"** (repeatedly asking why an issue occurred), but importantly, **don't stop at a human error** answer. If the first "why" yields "because an admin missed an alert," follow up with "why was it possible to miss it?" – perhaps the alert was buried in a flood of logs or the admin was handling too many duties. By the fifth why, you should be uncovering deeper issues (like inadequate log filtering or understaffing). Some teams prefer **"The Infinite Hows"** – continuously asking *"How did this failure become possible?"* – to drill down into systemic causes (The Infinite Hows (or, the Dangers Of The Five Whys) - Kitchen Soap). These inquiry techniques force a broader view that can identify root causes like process flaws or technology limitations.
- **Just Culture and Psychological Safety:** Shifting to a second-story mindset requires a cultural change. Implement a **"just culture"** in security, where employees are encouraged to report mistakes or near-misses without fear of punishment. Experts note that in a just culture, *"a person making an error is simply the symptom of the underlying problem"* (How Instituting a 'Just Culture' Improves Security - Dark Reading). This perspective is crucial in cybersecurity, because people *will* click phishing emails or configure things wrong occasionally; the organization's response should be to fix the system, not shame the individual. Leadership must set the tone by reacting to incidents with curiosity ("How did this happen?") instead of anger. Over time, this encourages honesty and learning. Teams might openly discuss smaller security slips (lost laptops, minor outages) to practice extracting second stories. When the big incident happens, this habit ensures the response is focused on improvement, not finger-pointing. A practical step is updating incident response policies to prohibit blaming language – for example, replacing phrases like "operator error" with descriptions of the factual circumstances ("the runbook step was missed due to unclear wording"). The end result is a safer environment where issues surface early and employees work together to strengthen defenses.
- **Systemic Fixes for Vulnerabilities:** In vulnerability management, apply second-story thinking by treating each vulnerability as a symptom of a deeper issue. Instead of berating a developer for a SQL injection bug, ask what in the development process allowed that flaw: Was secure coding training provided? Did code review or testing miss it? Perhaps the team lacks an automatic scanner for such patterns. The **fix** then might be introducing better security unit tests or tools, not just telling the developer "be more careful." Similarly, if a misconfiguration exposed a cloud storage bucket, the second story might reveal that the cloud governance tooling is insufficient or the configuration interface is overly complex and

error-prone. Addressing that might involve adopting misconfiguration scanning tools or improving templates and guidelines for cloud resources. By solving the *systemic* issues, you reduce the chances of repeat mistakes. Essentially, **treat vulnerabilities and incidents as feedback** on where your security program or processes need improvement. This approach turns incident response into a cycle of continuous improvement: find the root contributing factors, implement changes (technical controls, training, process tweaks), and measure results.

- **Cross-Team Collaboration:** Many security failures happen at the seams between teams – for example, Ops didn't communicate a change to Security, or developers weren't aware of a new policy. To find second stories, involve all relevant parties in the investigation. Create an incident review committee that includes not just the security analysts, but also developers, IT ops, product managers, etc. This helps surface different perspectives on what happened. One team might highlight that an alert was too technical to understand, while another notes that a critical patch was delayed due to a business decision. Bringing these together constructs a full narrative of the incident. Techniques like **"chronologies"** or detailed timelines can be useful: reconstruct the sequence of events and decisions, then analyze points where a different system condition could have broken the kill-chain. By collaboratively mapping out the incident, the team can identify systemic constraints (perhaps an approval process was too slow, or documentation was missing at a key moment). The outcome should be concrete action items that cut across silos – e.g. updating a process, adding a step to QA, or reallocating budget for better monitoring. This multidisciplinary approach mirrors how the TMI investigators included human factors experts, engineers, and managers to get the full picture. In cybersecurity, breaking down silos ensures the solutions address root causes, not just symptoms in one department.

By adopting these strategies, security teams can institutionalize the practice of finding the second story in every incident. Over time, this leads to more resilient systems. Instead of repeatedly reacting to "human error," organizations fix the underlying design and operational issues that allow minor lapses to become major incidents. The result is a cycle of *learning* – each breach or near-miss teaches the team how to harden the system further, much like each aerospace incident has driven improvements in flight safety. This proactive, blameless approach is now considered a best practice in forward-looking DevOps and SRE organizations, and it's increasingly crucial in cybersecurity as well.

Conclusion

The key takeaway of the *"second stories"* concept is that behind every apparent human error is a set of systemic factors that influenced it. The Three Mile Island accident taught industry observers that pointing fingers at operators was too simplistic; real improvement came from addressing design flaws, training gaps, and organizational issues that constrained operator behavior. Cybersecurity can greatly benefit from the same perspective. By looking beyond the immediate cause of a breach and exploring the environment in which it occurred, security teams can uncover latent vulnerabilities and process weaknesses that would otherwise remain hidden. This report illustrated how many high-profile security incidents were not simply the result of an individual's failure, but of multiple breakdowns in technology, process, and culture.

Embracing second-story thinking in application security and incident response leads to **more effective security practices**. It shifts the focus from punishment to prevention: instead of asking *"Who caused this breach?"*, teams ask *"How can we change our system so this mistake doesn't happen again?"*. This approach fosters a culture of continuous improvement and learning, where people are not afraid to report problems and dissect failures honestly. In practical terms, applying this perspective means conducting blameless post-incident reviews, investing in system-level fixes, and encouraging collaboration across departments to tackle root causes.

Ultimately, **security is a socio-technical challenge** – just as in nuclear safety, the interaction between humans, tools, and processes determines the outcomes. By applying the second stories mindset, organizations move closer to a just, resilient security culture. Rather than fighting the last war or blaming the last person who erred, they build stronger systems for the future. This systemic, blameless approach will not only reduce the likelihood of incidents but also minimize their impact when they do occur, as the organization continually learns and adapts. In a threat landscape where new breaches are inevitable, those who learn the right lessons (and not just the easy, surface-level ones) will be far better positioned to protect their assets and users. Adopting a second-story approach to cybersecurity is thus not just about fairness – it's about effectiveness and resilience in the face of complex challenges. The sooner security teams internalize this lesson from Three Mile Island, the sooner they can drive meaningful, lasting improvements in their security posture.