Eduvest – Journal of Universal Studies
Volume 5 Number 4, April, 2025
p- ISSN 2775-3735- e-ISSN 2775-3727

ANALYZING SYSTEMIC FAILURES IN IT INCIDENT MANAGEMENT: INSIGHTS FROM POST-MORTEM ANALYSIS

Faris Arifiansyah*, Yuanita Handayati
Institut Teknologi Bandung, Indonesia
Email : faris_arifiansyah@sbm-itb.ac.id, yuanita@sbm-itb.ac.*id*

ABSTRACT

The reliability of IT systems is critical for fintech companies, where service disruptions can lead to significant financial losses and reputational damage. Despite established incident management frameworks, recurring IT incidents persist, indicating systemic weaknesses in prevention, detection, and response. This study aims to identify the root causes of significant IT incidents, assess detection and resolution challenges, and provide actionable recommendations to enhance incident management. Using a qualitative approach, the research analyzed 26 post-mortem reports from an Indonesian fintech company (August 2023–2024), employing thematic analysis to categorize systemic failures. Findings revealed that 80% of incidents stemmed from internal changes, primarily due to inadequate testing, weak deployment controls, and misconfigured production settings, while 69% lacked proactive alerts, delaying detection. Incident response inefficiencies, such as slow escalations and insufficient post-fix monitoring, further prolonged resolution times. The study highlights the need for stricter change validation, standardized alerting mechanisms, and automated deployment checks to mitigate disruptions. These insights offer practical guidance for fintech and technology companies to reduce incident frequency, improve detection capabilities, and optimize response efficiency. The research contributes to the broader IT incident management field by empirically validating failure patterns in fintech environments and proposing data-driven solutions. Future research could explore AI-driven automation and organizational factors influencing incident handling.

KEYWORDS *IT Incident Management, Incident Prevention, Incident Detection, Fintech, Root Cause Analysis, System Reliability*

Article Info:
Submitted: 04-04-2025 Final Revised: 19-04-2025 Accepted: 26-05-2025 Published: 30-05-2025

How to cite: Arifiansyah, F., & Handayati, Y. (2025). Analyzing Systemic Failures in IT Incident Management: Insights from Post-Mortem Analysis. Journal Eduvest. *5*(4): 4260-4273.
E-ISSN: 2775-3727

INTRODUCTION

Mobile apps have become essential to modern life, streamlining everything from communication and task management to payments and entertainment. Technology companies are leading this digital revolution. These innovative organizations develop and deliver the software, hardware, and services that power our digital lives. However, despite the convenience and benefits they offer, technology companies face the challenge of achieving zero incidents. An Information Technology (IT) incident refers to an unexpected event or unplanned interruption that disrupts business operational processes or reduces the quality of an IT service (Standardization, 2018). The complex nature of modern technology systems and the constant introduction of new products and features make it more difficult to eliminate disruptions entirely (Z. C. et al., 2020).

In recent years, numerous IT incidents have highlighted the importance of service reliability. For example, the CrowdStrike incident in July 2024 caused widespread disruptions as thousands of commercial flights globally were canceled due to a faulty update released by CrowdStrike that triggered the Blue Screen of Death (BSOD) in Windows OS computers. This incident resulted in estimated losses between US$300 million and US$1 billion (Carpenter, 2024), and the company's stock price fell more than 15% in the following days (Saul, 2023). Another notable incident occurred in October 2021 when Facebook and its affiliated services, such as WhatsApp and Instagram, experienced a global outage that lasted nearly 6 hours. The outage was caused by a misconfiguration, leading to a significant drop in Facebook's share price by almost 5% (Brown, 2018). It is estimated that the company lost approximately US$79 million in ad revenue during the outage (Lee, 2020). These incidents have demonstrated the significant consequences of disruptions, including financial losses, reputational damage, and customer dissatisfaction.

The annual outages analysis report from Uptime Institute shows that the overall frequency of publicly reported outages from 2019 to 2022 remains high despite technological advancements. There is no sign of decreasing, even though there is an improvement in handling the impact of the outages (Simon, 2023). This trend indicates that the fundamental risk of disruptions still persists. From an industry perspective, the financial impact of server downtime varies across different sectors. The banking and finance industry experiences the biggest impact from a server downtime. It incurs an estimated loss of $9.3 million per hour of downtime (ITIC, 2017) underscoring the importance of incident prevention and response efficiency.

The fintech industry has experienced rapid growth in recent years (S. K. et al., 2018), fueled by increasing digital financial transactions. However, fintech firms also face increasing risks of IT disruptions due to their reliance on high-frequency transaction processing, microservices architectures, and third-party integrations (Efunniyi et al., 2022). While research on IT incident management has explored outage trends, failure patterns, and strategies for improving deployment reliability, there is still a lack of in-depth studies focusing on incident root causes, detection gaps, and response behaviors in fintech environments.

Chen et al. (2020) examined incident management challenges in a large-scale cloud system, identifying key weaknesses in impact estimation, service dependency mapping, and the triaging process. Their study found that incorrect incident classification and frequent reassignment significantly prolonged resolution times, with some incidents being transferred multiple times before reaching the appropriate team. While this research provides valuable insights into post-incident handling, it primarily focuses on large-scale cloud service providers rather than organizations operating in fintech environments. Additionally, its emphasis remains on optimizing incident response rather than addressing systemic root causes that contribute to recurring failures.

Aceto et al. (Aceto et, 2018) conducted a comprehensive survey of internet outages, analyzing causes related to network failures, external attacks, and cloud infrastructure issues. However, their study focused on industry-wide failures rather than internal enterprise IT incidents. Similarly, (2016) and (Kapel, 2023) investigated incident prevention strategies through structured change management, proposing predictive models to identify high-risk changes before deployment. Another study analyzed challenges in identifying incident-inducing changes, highlighting the need for improved traceability, data quality, and postmortem practices to enhance change failure analysis (E. Kapel D. Spinellis, and A. Van Deursen, 2024)While these studies emphasize prevention, they focus less on post-incident detection and response mechanisms, which this study aims to address. Gunawi et al. (2016) examined cloud service outages, highlighting failure patterns in large-scale distributed systems. However, research on cloud-focused failures often lacks insights into application-layer failures and the organizational processes governing incident management.

This research aims to bridge these gaps by conducting an in-depth post-mortem analysis of significant incidents in a fintech company. Unlike previous studies, this study investigates the full incident lifecycle, from root cause identification to detection gaps and resolution behaviors. The objective is to uncover recurring failure patterns and provide practical recommendations that technology-driven organizations can adopt to enhance incident prevention, detection, and response. By analyzing real-world post-mortem reports, this study provides data-driven insights that can help organizations improve their IT incident management strategies, particularly in high-transaction environments like fintech.

METHOD

This study analyzed 26 major IT incidents at a fintech company in Indonesia that were captured in post-mortem documents from August 2023 to August 2024. These documents were retrieved from the company's internal knowledge management system, where incident reports are archived. Post-mortems were selected as the data source because they contain factual and structured information compiled by engineering teams immediately after an incident occurs. Unlike interviews or surveys, which personal biases or memory recall limitations may influence, post-mortem reports provide a historical record of incidents, including detection timelines, root cause analysis, and corrective actions. This structured documentation makes them suitable for identifying systemic patterns across

multiple incidents. Each post-mortem follows a standardized format, capturing key attributes of the incident. Table 1 outlines the key details available in the reports, which form the basis of the study's analysis.

Table 1. Details Captured in Post-Mortem Reports

Item Name	Description
Incident Date	Date when the incident started
Title	Incident title summarizing the issue
Squad Owner	The team that owns the incident and post-mortem document decided based on the cause that triggers the incident
GMV Impact	Estimated financial loss in gross merchandise value (GMV)
Revenue Impact	Estimated direct revenue loss due to the incident
Time Incident Started	Exact timestamp of when the disruption began
Time Incident Detected	When the issue was first identified, whether via automated alerts, manual monitoring, or customer complaints
Time Incident Resolved	When the incident was fully mitigated
Time Post-Mortem Closed	When all corrective actions were completed and documented
Root Cause Category	Classification of the primary failure category, agreed upon by stakeholders
Incident Summary	Brief explanation of what occurred and how the issue was mitigated
Impact	Description of the functional disruption and affected users
Trigger	The initiating event or action that caused the incident
Detection	Explanation of how the issue was identified
Root Cause Analysis	Detailed analysis using the 5 Whys method to trace the underlying failure
Timeline	Chronological sequence of key events and actions from detection to resolution
Resolution & Recovery	Actions taken to mitigate and resolve the issue.
Corrective & Preventive Measures	List of actions aimed at preventing recurrence, categorized into corrections, preventions, and improvements.
Lessons Learned	Reflection on what went well, what failed, and unexpected factors that influenced the outcome
Related Squads	List of teams affected by the incident

To systematically extract meaningful insights from these post-mortems, this study applies thematic analysis, a widely used qualitative research method for identifying patterns in textual data. It involves a process where researchers engage with the data to identify and develop themes that emerge from the qualitative data set (Varpio, 2020). Following the methodology outlined in (Clarke, 2006), this study follows a structured process with six key phases.

The first phase is familiarization with the data, where researchers review all post-mortem documents to understand incident contexts. Next, initial codes (tags) are generated, focusing on failure patterns, detection gaps, and response inefficiencies. To avoid confusion between "code" in qualitative research and "code" in software engineering, this study uses the term "tagging" instead of

"coding." Tagging was performed using Taguette, an open-source qualitative data analysis tool, to streamline the annotation process.

After tagging was completed, the data was exported and structured for further analysis. Although this step is not explicitly listed in the original framework outlined in (Clarke, 2006)This study introduced it to ensure a consistent representation of incident factors across all cases. Each tag was limited to a single occurrence per incident to prevent the overrepresentation of frequently discussed issues in individual reports.

The next phase involved searching for themes and grouping similar issues to identify broader systemic patterns in IT failures. These themes were then reviewed and refined to ensure they accurately represented the underlying data. Finally, themes were defined and named, classifying root causes and response inefficiencies. This structured process ensures that the findings are based on empirical evidence rather than anecdotal observations.

RESULT AND DISCUSSION

Root Causes of Major IT Incidents

The analysis of 26 post-mortem documents revealed that internal changes, such as new feature deployments, system migrations, and configuration modifications, triggered 80% of major incidents. This indicates that failures mainly originate from software development, deployment, and operational processes rather than external factors like infrastructure failures or unexpected traffic surges. Several recurring systemic issues contributed to incidents across multiple root cause categories. Fig. 1 summarizes the most common failure patterns identified in the analysis.

The most frequently observed issue was inadequate testing, contributing to 10 out of 26 incidents. Many deployments lacked regression testing, leading to failures in existing critical functionality. Additionally, incomplete test coverage resulted in undetected failures that only surfaced in production environments. Deployment deficiencies were the second most frequent cause, accounting for 6 incidents. In some cases, manual deployment change logs contained errors or outdated information, leading to unreviewed changes being applied. Additionally, a lack of automated deployment validation resulted in production environments missing essential configurations.

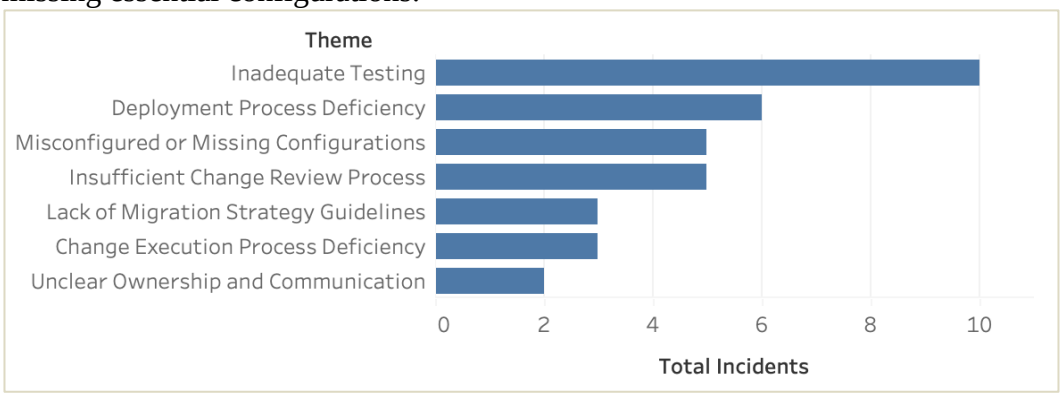

Figure. 1 Systemic issues contributing to IT incidents

Insufficient change review processes led to multiple incidents where modifications were merged or applied without proper peer or product owner approvals. Similarly, misconfigured or missing configurations caused system failures, as engineers often configured settings in staging but forgot to use them in production. Change execution deficiencies and a lack of standardized migration strategies also contributed to incidents. Teams frequently failed to actively monitor key metrics after changes, leading to delays in identifying failures. Furthermore, due to a large amount of work during infrastructure migration initiatives, each team executed their migrations independently with no guidance from the taskforce team that drives the project, leading to knowledge gaps and unexpected failures. Lastly, ownership ambiguities and communication failures also contributed to several disruptions where third-party IP changes were not communicated effectively between internal teams, leading to integration failures.

In addition to these systemic issues, several code and configuration-related incidents stemmed from unique causes. The incomplete validation logic caused two incidents, leading to duplicate database entries that triggered unintended feature behaviors. Another incident resulted from misunderstood function logic, where incorrect assumptions about how a function interacted with other system components led to unintended consequences. A performance-related incident was caused by a database indexing issue, where queries lacked proper indexing, causing significant slowdowns under high traffic conditions. Cache-related failure was also identified, where there was a bug in the cache implementation that excessively increased traffic to the authentication server by continuously requesting new tokens, ultimately overwhelming the system. In another case, an incident was caused by a manual certificate renewal process where the team was unaware that Kubernetes SSL certificates required manual updates, leading to a system integration failure with a third-party provider.

Incident Detection and Response Gaps

The post-mortem documents provide detailed insights into incident detection and resolution times. Each post-mortem contains Time to Detect (TTD) and Time to Resolution (TTR) data, allowing for a quantitative analysis of how long incidents remained undetected and how quickly they were mitigated once identified. Table 2 summarizes these times across all major incidents.

Table 2. Time to Detect (TTD) and Time to Resolution (TTR) for Major Incidents

Incident Number	TTD (In Minutes)	TTR (In Minutes)	Total Incident Duration (In Minutes)
2024080704	0	14	14
2024080501	5583	5110	10693
2024071101	24529	1371	25900
2024070301	870	370	1240
2024070202	11	78	89
2024070102	148	311	459
2024052901	6288	30519	36807

2024042201	11429	10085	21514
2024042101	254	6	260
2024031903	7401	49	7450
2024022901	0	175	175
2024021901	5464	306	5770
2024020601	7260	300	7560
2024013001	815	30	845
2024011801	40839	8664	49503
2023112401	0	96	96
2023110301	348	60	408
2023102601	1090	14	1104
2023101901	31	30	61
2023101702	8562	1939	10501
2023101701	790	170	960
2023092202	17	173	190
2023092102	11	28	39
2023091301	16	12	28
2023081501	6859	184	7043
2023081001	75	60	135
Average	4950	2314	7263

The results show that the Mean Time to Detect (MTTD) was 82.5 hours, while its Mean Time to Resolution (MTTR) was 38.5 hours. These values are substantially higher than industry benchmarks. A 2023 survey on business outages found that 44% of companies reported an MTTD of 30 minutes or less, while only 21% exceeded 60 minutes. For resolution times, 60% of organizations resolved incidents within 30 minutes, while only 34% took longer than one hour [18]. Compared to these figures, the studied fintech company's MTTD is significantly longer, extending incident durations and business disruptions.

A key contributing factor to prolonged detection times is the lack of automated alerts for business-critical metrics and system health indicators. The analysis revealed that 18 out of 26 incidents (≈69%) were detected through manual means, such as customer complaints, manual employee checks, or third-party partners, rather than through proactive alerting systems. Further investigation into post-mortem documents, particularly the Detection section, showed that many incidents remained undetected because of lacked automated alerts on critical indicators such as: (1) Transaction success rate drops (for specific products, payment methods, or partners). (2) CPU utilization spikes. (3) Application crashes. (4) User registration or visit declines. To illustrate this, Table 3 presents a sample of incidents that lacked automated alerts, showing how they were eventually detected.

Table 3. Sample Incidents with Missing Alerts

Incident Number	Post-Mortem Excerpt Indicating Missing Alert	Alert Category
2024070202	The Auth server CPU was impacted due to the increase in load, but had no CPU alerting, making it harder to investigate	High CPU Utilization
2024042201	Detection: Complaint from users … Some users are getting stuck on a page in the app when trying to open the QRIS feature	App Crash / Stuck
2024013001	Detection: User complains through our CS, and CSM reports to us through Slack … The user cannot create a transaction for Product A in the platform	Transaction Drop for Specific Product
2023101702	Detection: The Product team first detected the issue when getting complaints from users … Impact: Opportunity lost to use Payment Method A as the payment method	Transaction Drop for Specific Payment Method
2023101701	Detection: A support channel report said several Partner A products were closed.	Transaction Drop for Specific Partner
2023081501	Detection: Raised by the payment team … Impact: Product B transaction with Payment Method B dropped to zero for 5 days.	Transaction Drop for Specific Payment Method
2024070102	Detection: Complaint from users … Impact: Customers can't transact Electricity Prepaid, Property tax, and Vehicle Tax products that are supplied by Partner X	Transaction Drop for Specific Partner
2024071101	Detection: Got a report from the product team and business team … Impact: The User is unable to open the app from push notification. Tapping it does nothing and does not redirect to the respective screen in the app.	Visit Drop from Specific Channel
2024052901	Detection: We got some reports from our users that they can't do QRIS registration … App version V will crash when the user does QRIS registration.	App Crash / Stuck, User Registration Drop

The most common problem was missing alerts for drops in transaction success rates. Eleven of the 18 missing alerts (61%) were related to transaction drops for specific product types (like phone credit or e-wallets), partners or suppliers, payment methods (like bank virtual accounts or paylater), or a

combination of these factors. A deeper analysis revealed that these alerts were never configured but misconfigured or inconsistently applied. The primary challenge stemmed from the microservices architecture, where each product type operates on its own microservice with custom metrics, requiring separate alert configurations for each. The absence of a standardized alerting policy resulted in many critical alerts being overlooked. As a result, detection often relied on manual monitoring or user complaints, contributing to longer detection times and prolonged incident durations.

While detection delays contributed to prolonged incidents, the analysis also revealed inefficiencies in the response process. Even when incidents were identified, resolution times remained high. The time to resolve the incident (TTR) varied significantly across cases. While some incidents were mitigated within minutes, others remained unresolved for multiple days due to response inefficiencies. Table 4 highlights incidents where response inefficiencies led to extended service disruptions.

Table 4. Incident Response Issues

Incident Number	Tags	Explanation
2024052901	Lack of Post-Fix Monitoring	After deploying the initial fix in version V (on the same day the issue was detected), the team did not actively monitor crash metrics to verify its effectiveness. Another engineer eventually flagged the recurring crash 11 days after the 50% rollout began.
2024042201	Delayed Incident Reporting	The incident was detected on April 18, 2024, but the incident report was only created on April 22, 2024, leading to a four-day delay.
2024011801	Slow Incident Resolution	The anomaly was reported on January 12, 2024, but the bug was only identified on January 16, 2024, after a four-day delay. It highlights a significant delay in identifying the root cause after the detected issue. This suggests inefficiencies in the debugging or escalation process that could have prolonged the incident resolution
2024080501	Delayed Resolution Due to Dependency on External Partner	The resolution process was delayed because it relied on adjustments from the external partner. Although the team acted promptly by regrouping and escalating the issue, the dependency on the partner's system changes prolonged the resolution process
2023101702	Delayed Incident Reporting, Slow Incident Resolution	There was a two-hour delay in escalating the issue from the customer support to the incident support channel. After the war room was created, there was a delay in deploying the fix, which could have been expedited. The time gap between identifying the root cause and deploying the fix suggests room for improvement.
2024071101	Delayed Incident Reporting,	The issue was detected at 5:17 PM on July 11, 2024, but it was only reported to the incident channel at

	Prolonged Testing and Deployment Process	8:53 PM. The fix was also merged at 10:32 AM on July 12, 2024, but the app hotfix was only released on the Google Play Store at 4:08 PM, almost six hours later.
2024070301	Slow Incident Resolution	The finance team reported the issue on July 3, 2024, at 10:55 AM, but the fix was only deployed and tested by 5:05 PM on the same day, over six hours after detection.
2024021901	Delayed Incident Reporting	The issue was reported in the customer support channel at 9:14 AM, but it was only reported as an incident at 1:18 PM, nearly four hours later
2024020601	Slow Incident Resolution	The anomaly affecting Product A transactions was found at 09:00 AM on February 5, 2024, but the deployment to resolve the issue did not occur until 3:00 PM on the same day

The analysis of incident response behaviors reveals several systemic gaps that contributed to prolonged incident resolution times. The most frequent issue was delayed incident reporting, which appeared in at least four separate incidents. In multiple cases, incidents were already detected but not immediately escalated to the incident support channel. Another recurring issue was slow incident resolution, where teams took excessive time between identifying the root cause and deploying a fix. This often stemmed from delays in debugging, fixing deployment, and testing before release. Additionally, a notable issue was a lack of post-fix monitoring, where teams deployed fixes but did not actively monitor the impact, leading to recurring failures. One incident remained unresolved for 11 days after an initial fix was rolled out. These findings indicate several process inefficiencies in the incident response workflow, including timeliness of reporting, debugging speed, and post-resolution validation.

The findings of this study highlight systemic weaknesses in the studied fintech company's incident and change control processes, particularly in change validation, detection mechanisms, and response efficiency. The high frequency of incidents triggered by internal changes underscores the lack of sufficient preventive controls in testing and deployment. Additionally, gaps in automated monitoring led to a heavy reliance on manual detection. This significantly prolongs the Mean Time to Detect (MTTD). Once detected, inefficient escalation and debugging processes further delayed Mean Time to Resolution (MTTR), increasing operational and financial risks. These patterns indicate that while the organization has an established incident management framework, the lack of standardized enforcement mechanisms, automation, and proactive strategies leaves the system vulnerable to recurring failures.

These observations align with existing research on software reliability and IT incident management. Studies have shown that inadequate testing and misconfigurations lead to system failures. For instance (Yuan et, 2014), analyzed large-scale distributed system failures and found that misconfigurations caused 23% of outages, while 58% of catastrophic failures could have been prevented through simple pre-release testing. This underscores the need for stronger validation

mechanisms before deployment. One potential improvement is to mandate a quarterly review of automated test coverage to identify critical business flows missing from test scenarios. Additionally, integrating these automated tests into the deployment pipeline can help ensure that production releases only proceed if all tests pass successfully. This aligns with the Service Validation and Testing framework of the Information Technology Infrastructure Library (ITIL) (Axelo, 2019), emphasizing pre-deployment verification to minimize failure risks.

Beyond testing deficiencies, deployment and change control gaps were another major source of incidents. Multiple failures were caused by insufficient change approvals, where modifications were merged without code owner or product owner validation. A widely adopted solution is the Code Owners feature, available in version control platforms including GitHub, GitLab, and Bitbucket (Neville-O'Neill, 2023). It helps to prevent merging changes unless explicitly reviewed by designated owners. Studies have shown that clear code ownership improves software quality and reduces defects, and integrating code review activities further enhances these benefits (Bird et, 2011; Greiler et, 2015; Thongtanunam et, 2016) .

Another deployment-related issue was human error in change tracking. Manual deployment logs resulted in unreviewed modifications being released unintentionally. Organizations should consider automating deployment change log generation within the CI/CD pipeline to mitigate this. Logs can be auto-generated by comparing release tags with the live production state, ensuring that all changes are explicitly reviewed before rollout. Furthermore, adopting progressive deployment strategies, such as canary releases, can help detect anomalies before full-scale rollout. Companies like Google and Facebook have successfully leveraged these techniques to maintain high availability and stability during deployments (M. D. P. et al., 2023; T. S. et al., 2016).

Detection gaps were another key factor contributing to prolonged incident durations. 69% of major incidents were detected manually, through customer complaints, partner notifications, or employee observations, rather than automated alerts. Studies have emphasized that effective incident detection relies on real-time monitoring of key business and system health indicators (H. W. et al., 2024)Based on this study's findings, standardized alerts can help reduce MTTD. Table 5 shows the baseline alert that each service should have, particularly in a fintech company.

Table 5. Baseline Alert Standards for Incident Detection

Category	Required Alert Type	Justification
Business Impact Metrics	Drop in Successful Transactions per Product Type, Payment Method, and Partner	Many incidents were detected late because no alert was in place to track sudden drops in success rates. This should be standardized across all financial transactions.
System Performance	High CPU Utilization Alert, High Latency, High Error Rate	A previous incident revealed that CPU usage spiked but was not detected due to missing alerts, delaying response times.

External Dependencies	Partner API Response Time & Error Rate	Third-party partner failures are a recurring issue. All partner integrations must implement automatic monitoring for slow response times or increased failure rates.
Infrastructure Health	Database Query Execution Time Spikes, Database High CPU Utilization, Message Queue High Consumer Lag	Multiple incidents were caused by slow queries, DB locks, or replication lag, which should trigger real-time alerts.

Incident response inefficiencies further exacerbated resolution times. Delayed incident reporting and slow debugging were observed in multiple cases. Research has demonstrated that structured on-call escalation procedures help accelerate response times (P. C. et al., 2012). Organizations like Netflix conduct failure injection drills, simulating outages to train engineers in real-time troubleshooting, reducing resolution delays during actual incidents (Alvaro, 2016). Fintech companies could adopt similar approaches to enhance incident response readiness.

While automation offers promising solutions, full-scale adoption of AI-driven solutions such as AIOps requires substantial investment in infrastructure and expertise. Studies have found that AIOps implementations reduce MTTD and MTTR by enabling predictive analytics and automated incident resolution (Z. C. et al., 2020; Tian, 2025). However, challenges such as false positives, algorithmic bias, and the need for human intervention remain (Tian, 2025)A more pragmatic approach for fintech companies would be to improve their structured incident escalation processes before considering AI-driven automation.

CONCLUSION

This study analyzed 26 post-mortem reports from a fintech organization (August 2023–2024) to identify root causes of significant IT incidents, revealing that 80% stemmed from internal changes due to inadequate testing, weak deployment controls, and misconfigured production settings, while 69% lacked proactive alerting, delaying detection. The research highlights systemic gaps in incident management, including slow escalations and insufficient post-fix monitoring, and proposes solutions such as stricter change validation, progressive deployment strategies, automated checks, and standardized alerting baselines to reduce disruptions. Although limited by reliance on documented post-mortems, which may omit informal coordination challenges, the findings offer actionable insights for fintechs to strengthen incident prevention and response, with future research directions including AI-driven resolution and organizational behavior analysis.

REFERENCES

Aceto et, al. G. (2018). A comprehensive survey on internet outages. *Journal of Network and Computer Applications*, *113*, 36–63. https://doi.org/10.1016/j.jnca.2018.03.026

al., H. W. et. (2024). *Anomaly detection for incident response at scale*. https://doi.org/https://arxiv.org/abs/2404.16887

al., M. D. P. et. (2023). Refining a software system deployment process model through empirical studies. *J Comput Sci Technol*, *23*(1), e06. https://doi.org/10.24215/16666038.23.e06

al., P. C. et. (2012). *Computer security incident handling guide: Recommendations of the National Institute of Standards and Technology*. Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-61r2

al., S. K. et. (2018). Indonesia's fintech industry is ready to rise. *BCG*. https://doi.org/https://www.bcg.com/publications/2023/fintech-industry-indonesia-growth

al., T. S. et. (2016). Continuous deployment at Facebook and OANDA. *38th International Conference on Software Engineering Companion*, 21–30. https://doi.org/10.1145/2889160.2889223

al., Z. C. et. (2020). Towards intelligent incident management: why we need it and how we make it. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, *1487–1497*. https://doi.org/10.1145/3368089.3417055

Alvaro, al. P. et. (2016). Automating Failure Testing Research at Internet Scale. *Seventh ACM Symposium on Cloud Computing*, 17–28. https://doi.org/10.1145/2987550.2987555

Axelo. (2019). *ITIL 4: Foundation* (4th ed. (ed.)). AXELOS.

Bird et, al. C. (2011). Don't touch my code! examining the effects of ownership on software quality. *19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 4–14. https://doi.org/10.1145/2025113.2025119

Brown, A. (2018). Facebook lost about $65 million during hours-long outage. *Forbes*. https://doi.org/https://www.forbes.com/sites/abrambrown/2021/10/05/facebook-outage-lost-revenue/

Carpenter, G. (2024). *A closer look: unveiling the global impact of crowdstrike event*. https://doi.org/https://www.guycarp.com/content/dam/guycarp-rebrand/insights-images/2024/07/2024-8-1-Unveiling-the-Global-Impact-of-CrowdStrike-Event.pdf

Clarke, V. B. and V. (2006). Using thematic analysis in psychology. *Qual Res Psychol*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

E. Kapel D. Spinellis, and A. Van Deursen, L. C. (2024). On the difficulty of identifying incident-inducing changes. *46th International Conference on Software Engineering: Software Engineering in Practice*, 36–46. https://doi.org/10.1145/3639477.3639755

Efunniyi, A. G. A. C. P., Akwawa, L. A., Azubuko, C. F., & Sanyaolu, T. O. (2022). Optimizing systems integration for enhanced transaction volumes in Fintech. *Finance & Accounting Research Journal*, *4*(5), 345–363. https://doi.org/10.51594/farj.v4i5.1511

Greiler et, al. M. (2015). Code ownership and software quality: a replication study. *12th Working Conference on Mining Software Repositories*, 2–12.

Gunawi et, al. H. S. (2016). Why does the cloud stop computing? *Proceedings of the Seventh ACM Symposium on Cloud Computing*, 1–16. https://doi.org/10.1145/2987550.2987583

ITIC. (2017). *Average cost per hour of server downtime worldwide in 2017, by vertical industry (in million U.S. dollars)*. https://doi.org/https://www.statista.com/statistics/780699/worldwide-server-hourly-downtime-cost-vertical-industry/

Kapel, E. (2023). Incident prevention through reliable changes deployment. *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 200–202. https://doi.org/10.1109/ICSE-Companion58688.2023.00055

Lee, J. (2020). *Here's how many millions in ad revenue facebook could have lost during outage*. https://doi.org/https://www.snopes.com/news/2021/10/04/facebook-ad-revenue/

Murthy, S. G. and K. (2016). Understanding the role of change in incident prevention. *2016 12th International Conference on Network and Service Management (CNSM)*, 268–271. https://doi.org/10.1109/CNSM.2016.7818430

Neville-O'Neill, B. (2023). A modern guide to CODEOWNERS. *Aviator*. https://doi.org/https://www.aviator.co/blog/a-modern-guide-to-codeowners

Saul, D. (2023). Crowdstrike stock tanks 15%—set for worst day since 2022. *Forbes*. https://doi.org/https://www.forbes.com/sites/dereksaul/2024/07/19/crowdstrike-stock-tanks-15-set-for-worst-day-since-2022/

Simon, A. L. and L. (2023). *Annual outages analysis 2023*.

Standardization, I. O. for. (2018). *Information technology — service management — part 1: service management system requirements (ISO/IEC Standard No. 20000-1:2018)*. Geneva.

Thongtanunam et, al. P. (2016). Revisiting code ownership and its relationship with software quality in the scope of modern code review. *38th International Conference on Software Engineering*, 1039–1050. https://doi.org/10.1145/2884781.2884852

Tian, Y. Y. (2025). Analyzing the effectiveness of automated incident response mechanisms in reducing downtime and improving service reliability in large-scale distributed systems. *International Journal of Site Reliability Engineering (IJOSRE)*, *6*(1), 1–10. https://doi.org/10.5281/zenodo.14799653

Varpio, M. E. K. and L. (2020). Thematic analysis of qualitative data: AMEE Guide No. 131. *Med Teach*, *42*(8), 846–854. https://doi.org/10.1080/0142159X.2020.1755030

Yuan et, al. D. (2014). Simple testing can prevent most critical failures: an analysis of production failures in distributed data-intensive systems. *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, 249–265. https://doi.org/10.5555/2685048.2685068