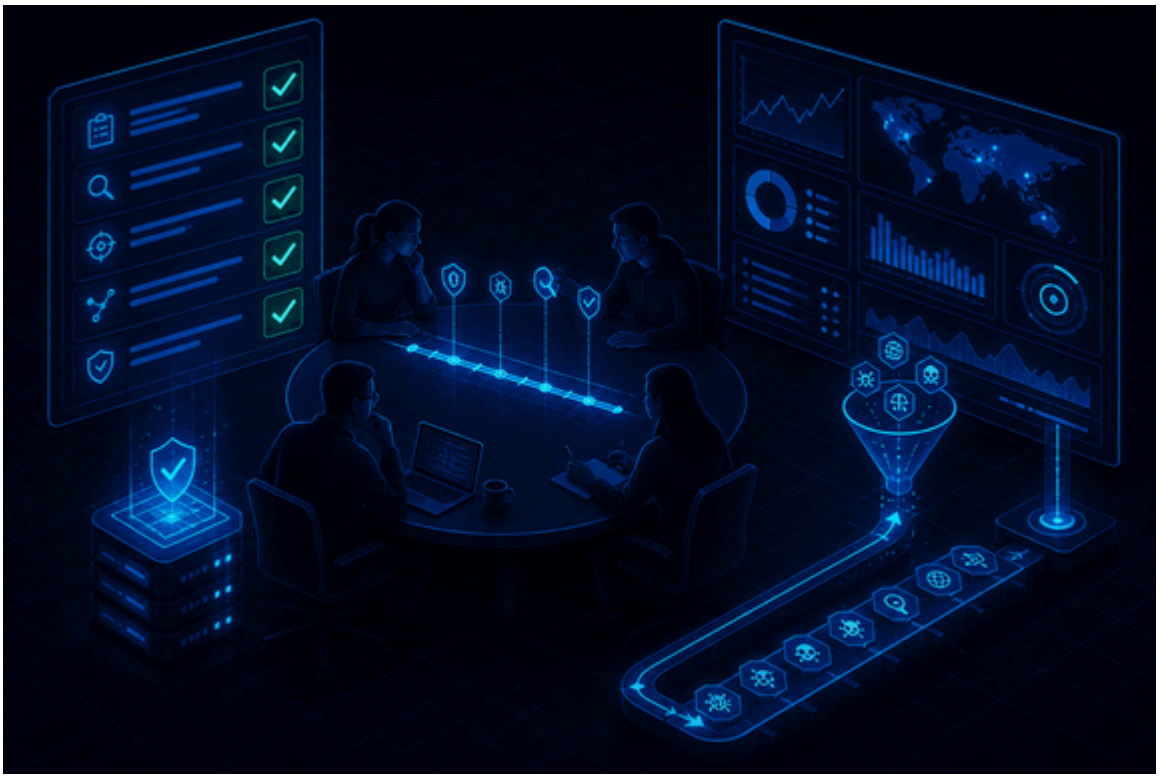
Closing the Incident: Validation, Lessons Learned, and What You Keep

Mon, Sep 14, 6:00 PM EDT · https://scottslab.io/posts/post-incident-validation-and-lessons-learned

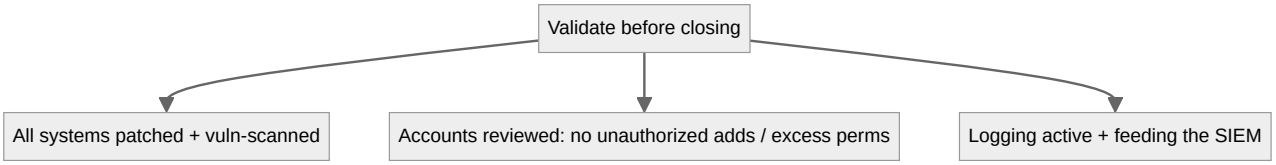


TL;DR

Don't declare victory until you've validated: every system patched and vuln-scanned, accounts reviewed for unauthorized additions or excess permissions, and logging confirmed active and feeding the SIEM. Then run a lessons-learned session. NIST says within a few days, not weeks, while it's still fresh, led by a neutral facilitator so it stays blameless rather than defensive, then write an incident summary report for institutional memory. Retain evidence per your data-retention policy (and confirm no pending legal action before discarding), feed any new indicators of compromise into monitoring, and give every action item an owner and a date, or it never happens.

The most dangerous moment in incident response is the one where everyone's exhausted and wants to call it done. Before you close an incident, validate that it's actually closed. Confirm every affected system is patched and then vulnerability-scanned to prove it. Review the accounts for anything the attacker left behind: unauthorized additions, or existing accounts quietly granted excess permissions, which is a classic persistence move. And verify that logging is active and feeding into the SIEM, because attackers disable logging and a "recovered" system that isn't being

watched is just the next incident waiting quietly. Validation is the difference between "the alerts stopped" and "the attacker is actually gone."



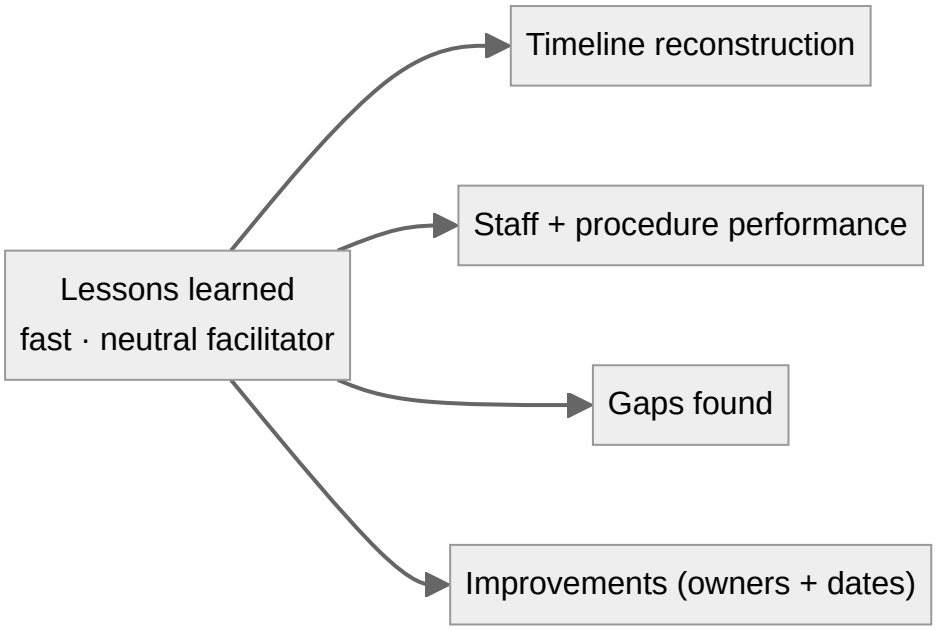
Closing the Incident: Validation, Lessons Learned, and What You Keep

Run it in days, not weeks

Then comes the part that makes the next incident less painful: the lessons-learned session. NIST is specific about timing here: hold it within several days of the incident ending, not whenever the calendar has a gap, because the details degrade fast and the timeline blurs the moment people move on to the next thing. Have a neutral facilitator lead it, not the person whose decisions are under review, so it stays honest and blameless rather than defensive.

"Blameless" isn't a mood, it's a mechanism. The moment a lessons-learned session can end someone's career, people stop volunteering what actually happened, and the record quietly rewrites itself into something defensible instead of something true. A blameless format, pioneered in incident-heavy fields like aviation and adopted widely in tech operations, works because it separates the question "what happened" from the question "whose fault is it," and only the first question actually makes the next incident shorter.

NIST's own list of questions is worth stealing wholesale rather than improvising your own: exactly what happened and at what times; how well staff and management performed and whether documented procedures were followed and were adequate; what information was needed sooner; whether any action taken actually got in the way of recovery; what staff and management would do differently next time; how information sharing with other organizations could improve; and what precursors or indicators should be watched for to catch a repeat earlier. That last one closes the loop back to detection. A lessons-learned session that doesn't produce new things to watch for hasn't actually learned anything.



Run it in days, not weeks

The session should also produce an honest, objective grade on the response itself, not just the incident. That's a different question, and it's the one people skip because it's uncomfortable. Did the incident actually cause damage before anyone detected it, or did the tooling catch it in time? Was the real attack vector and root cause actually identified, or is "we're not entirely sure how they got in" quietly getting reported as resolved? Is this a recurrence of something you've already seen? If it is, that's not a new incident. It's the same finding from last quarter's report showing up again with a different timestamp. Comparing your initial impact assessment against what the damage actually turned out to be is worth doing every time, too; a team that consistently over- or under-calls severity in the first hour has a triage problem worth fixing on its own.

Where lessons go to die

A few artifacts and obligations close it out. Write an incident summary report: the technical record that becomes institutional knowledge and training material, so the next responder inherits what you learned instead of relearning it the hard way. Retain the evidence according to your data-retention policy. Critically, confirm there's no pending or anticipated legal action before you discard anything, because destroying evidence under a live hold is its own serious problem (the eDiscovery preservation trap). And finally, mine the incident for new indicators of compromise: the IPs, hashes, domains, and behaviors the attacker used. Add them to your monitoring program, so the same actor trying the same thing trips a wire next time.

None of that matters if the action items don't survive contact with next week. "We should improve backups" is not an action item. It's a mood. It needs an owner, a date, and a home in the same backlog where real engineering work lives, not a line in a report nobody reopens. That's the actual mechanism behind the most reliable finding in this field: the same incident recurring, because the

last report's lessons were identified correctly and then implemented by no one in particular. Track time-per-incident too: how long from first indicator to detection, to containment, to closure. That trend line, watched across a year of incidents, tells you whether the program is actually getting better or just staying busy.

An incident handled well doesn't just end. It makes the organization measurably harder to hit the same way twice. The only thing standing between "measurably" and "hopefully" is whether anyone was still tracking the action item three months later.

Written by Scott S. — Contact: scott@scottschlangen.com — scottslab.io — https://scottslab.io/posts/post-incident-validation-and-lessons-learned