

April, 7, 2026

(Initial notes/input to the tool Loveable of an idea for an application)

Let's build a "white hat" tool that scans sites and applications for SOC 2 vulnerabilities. Here is a list of typical issues reviewed and often found during security audits: Common SOC 2 Compliance Violations

SOC 2 audits evaluate controls across five Trust Services Criteria (TSC): Security, Availability, Processing Integrity, Confidentiality, and Privacy. Most violations cluster around a handful of recurring themes:

- Access Control Weaknesses
- Overly permissive user access (violating least privilege)
- Failure to revoke access promptly when employees are terminated or change roles
- Lack of multi-factor authentication (MFA) on critical systems
- Shared or generic accounts without individual accountability
- Insufficient Change Management
- Deploying code or infrastructure changes without formal approval workflows
- No separation of duties between developers and production environments
- Missing or incomplete change logs
- Inadequate Risk Assessment
- No formal, documented risk assessment process
- Risks identified but not assigned owners or remediation timelines
- Infrequent reassessment (e.g., only annually when the threat landscape shifts faster)
- Vendor and Third-Party Management
- No formal process for evaluating the security posture of vendors
- Missing Business Associate Agreements (BAAs) or Data Processing Agreements (DPAs)
- Not reviewing vendor SOC 2 reports before granting data access
- Monitoring and Logging Gaps
- Insufficient log retention periods
- Logs collected but never actively reviewed or alerted on
- No intrusion detection or anomalous activity alerting

- Vulnerability Management
- Infrequent or inconsistent patch cycles
- No formal penetration testing or only ad hoc scanning
- Known vulnerabilities left unresolved beyond acceptable SLAs
- Incident Response Failures
- No documented, tested incident response plan
- Incidents not formally logged, tracked, or post-mortemed
- Failure to notify affected customers within required timeframes
- Physical and Environmental Controls
- Inadequate controls around who can physically access servers or offices
- No visitor logs or escort policies for data center areas
- Policy and Training Gaps
- Security policies that are outdated, undocumented, or not communicated to staff
- No evidence of employee security awareness training
- Policies exist on paper but are not operationally enforced
- Data Handling and Encryption
- Sensitive data stored or transmitted without encryption
- No clear data classification or retention/disposal policies
- Personal data collected beyond what is necessary (relevant to the Privacy criteria)

The most frequent root cause across all of these is the gap between having a policy written down and actually operating to it consistently. Auditors look for evidence of controls working over time, not just documentation that they exist. Organizations often fail not because they lack policies, but because they can't produce logs, tickets, or records proving the controls were followed.

The application we're going to build will probe sites and applications for vulnerabilities and, when an issue is discovered, it will 1. surface it with appropriate, clear, contextually relevant details for the admin viewing the interface, and 2, with a brief pause for review by the admin, automatically generate a brief, friendly notification email to the offending site or software owner detailing the issue that was detected. The software itself must be security-hardened. The interface will rely on a Material Design-based ui design system and will default to a "dark" mode.

Concerns regarding probing or scanning someone's site or offering: There are legal nuances around unsolicited scanning of third-party systems, even with good intent. Depending on jurisdiction, the CFAA in the US in particular has been interpreted broadly. Worth understanding before you get too far in.

Second pass prompts:

Resetting the interface:

The interface will require a means to "clear" the current scan and results. When this is initiated, the application should "archive" the last scan, creating a directory labeled with a human-readable name and then storing the scan results data in an equally human-readable format, either a mimic of the current readout screen or output as a pdf.

Scan history detail:

The current interface includes a "Scan History" pane. This needs to be enhanced by adding a more complete date stamp, one that shows the date as well as the time.

Notify owner button:

At this point, I'm not certain how the tool would know where to send a notification email, thus the "Notify Owner" button feels ungrounded in reality.

Perhaps we can show what we have found or identified as owner information/details first, in the interface somewhere distinct/discrete, and then, once that information has been vetted/validated by the user, they are presented with the "notify owner" button. My initial thought is to do a quick scan of the site or software to determine if any contact information is readily available and, if an email address is found, use that for the initial contact.

Actual notification:

Are we building a messaging/email platform here, too? Or will we launch the users designated email application? Do we need to ask them to identify what email application they want to use in the pre-flight/set-up stage or process?

Dismiss button:

What happens if the user clicks the "dismiss" button in a finding? Where does that finding go? I believe it should be archived, in the same directory that scan results would go into. So we'll need a clear information hierarchy created/established in relation to the storage of scan results and dismissed alerts.

As a part of archiving a "dismissed" notification, the interface should show to the user that the notification has been archived to such and such location, clearly showing the user what that location is (such as a path) and, by pointing to or highlighting where they can click to access archived project notifications.

NOTE: this indicates a new user interface element, perhaps a simple icon in the topline nav, with an appropriate icon so that our user can navigate directly to the archives from the main interface. I would like to explore utilizing the lower portion of the "Scan History" window as a viewer of the "archived" scans.

Iconography or creating a "key" for issue sorting:

In the interface, we're currently showing both "critical" and "high" as red. There needs to be a clearer distinction made between these two visually. A quick scan of the list of issues would find our user perhaps conflating or not giving enough critical attention to an actual "critical" issue if they inadvertently missed the labeling of the issue as critical.

Incident severity definitions:

It feels to me that it might be useful and perhaps instructive to our user what the various severity levels actually mean, how do they translate or map to real-world implications? Without muddying the interface, perhaps using progressive disclosure or an icon in the topline nav, we could make available, in minimal clicks, access to a glossary or key to understanding the severity levels. And it would be nice not to lose context when accessing this information.

Clear affordances:

While in Nielsen's ux heuristic checklist there is an item specifying the use of a simple, flat interface, I do not want to avoid the use of clear affordances on controls. This must be respected and implemented as a feature is introduced. A user must be able to clearly see, without having to click first, that an interface element offers some sort of action.

The uploaded document describes seven areas of improvement. Here is the plan organized by feature.

1. Clear / Reset Scan with Archiving

****What changes:**** Add a "Clear Scan" button (visible when a scan is completed) in the header or scan form area. Clicking it archives the current scan into an in-memory archive store (with a human-readable name like `example.com — Apr 8 2025 2:34 PM`) and resets the UI to idle.

****Files touched:****

- `ScannerDashboard.tsx` — add archive state (`ArchivedScan[]`), clear handler that pushes current findings + metadata into archive, resets status/findings/progress

- `ScanForm.tsx` — show a "Clear Scan" button next to the scan button when status is `completed`

- `types/scanner.ts` — add `ArchivedScan` interface (id, name, url, date, findings, dismissed findings)

2. Scan History — Full Date+Time Stamp

****What changes:**** Show date and time (e.g., "Apr 8, 2025 · 2:34 PM") in history entries instead of just time.

****Files touched:****

- `ScanHistory.tsx` — change `toLocaleTimeString()` to a formatted date+time string using `toLocaleDateString` + `toLocaleTimeString`

3. Rethink "Notify Owner" — Contact Discovery First

****What changes:**** During the scan, attempt to identify contact information (simulated). Each finding will optionally carry a `detectedContact` field. In the FindingCard, show discovered contact info first; the user validates/edits it, and only then is the "Notify Owner" button enabled.

Rather than building an email platform, we'll compose a `mailto:` link that opens the user's default email client with a pre-filled subject and body summarizing the finding.

****Files touched:****

- `types/scanner.ts` — add optional `detectedContact?: { email?: string; source?: string }` to `Finding`

- `lib/scanner.ts` — add simulated contact info to some finding templates

- `FindingCard.tsx` — show contact info section with an editable email field; "Notify Owner" becomes a `mailto:` link button that only appears after the user confirms the email

4. Dismiss → Archive with Feedback

****What changes:**** When a finding is dismissed, instead of just fading it, move it to an archived/dismissed list. Show a brief toast/inline message: "Archived to [Scan Name] → Dismissed Findings" with a clickable link.

****Files touched:****

- `ScannerDashboard.tsx` — track dismissed findings separately; show toast with archive path
- `FindingCard.tsx` — dismiss animation, then remove from active list
- `ScanHistory.tsx` — add a lower section "Archived Scans" that lists archived scans; clicking one shows its findings (including dismissed ones) in a sheet/dialog

5. Archive Viewer in Scan History Pane

****What changes:**** The lower portion of the Scan History sidebar becomes an "Archived Scans" viewer. Add a nav icon (e.g., `Archive`) in the top header bar for quick access.

****Files touched:****

- `ScanHistory.tsx` — split into two sections: recent scans (top) and archived scans (bottom)
- `ScannerDashboard.tsx` — header gets an Archive icon button that scrolls to / highlights the archive section
- New component: `ArchivedScanViewer.tsx` — a sheet or inline expandable that shows full findings of an archived scan

6. Visual Distinction: Critical vs High Severity

****What changes:**** Critical and High currently both appear red. Differentiate them clearly:

- **Critical:** deep red/magenta background pulse or glow, filled shield icon, bolder card border
- **High:** orange-red, triangle icon (already different), subtler styling

Also add a **Severity Guide** accessible from the header (info icon or '?' button) as a popover/dialog explaining each level's real-world implications — using progressive disclosure so the user doesn't lose context.

Files touched:

- `index.css` — adjust `--critical` to a more distinct hue (e.g., magenta-leaning `340 85% 55%`) vs `--destructive` staying orange-red
- `SeverityBadge.tsx` — add a subtle pulse animation class for critical
- `FindingCard.tsx` — critical cards get a left accent bar or glow
- New component: `SeverityGuide.tsx` — a dialog/popover with definitions for each severity level and real-world mapping
- `ScannerDashboard.tsx` — add severity guide trigger icon in header

7. Clear Affordances on Controls

What changes: Ensure all interactive elements have visible affordance cues without requiring a click to discover them. Buttons get visible borders/shadows, hover states are more pronounced, expandable cards show a visible chevron with "View details" text hint, dismiss/notify buttons have distinct outlined styles.

Files touched:

- `FindingCard.tsx` — add "View details" micro-label next to chevron; ensure buttons have visible borders not just ghost styles

- `ScanForm.tsx` — scan button gets a subtle shadow/elevation
- General: audit all `variant="ghost"` buttons to ensure they have visible borders or underlines

Implementation Order

1. Types & data model updates (scanner.ts types, archive interfaces)
2. Critical vs High visual distinction + Severity Guide
3. Contact discovery + mailto notification flow
4. Dismiss → archive with feedback
5. Clear scan + archive storage
6. Archive viewer in history pane
7. Affordance audit across all controls
8. History date+time (quick fix alongside other changes)

All second-pass enhancements are now live:

Clear/Reset Scan — "Clear Scan" button archives current results and resets UI

Full date+time in scan history entries

Contact discovery + mailto — editable email field with confirm step; "Notify Owner" opens a pre-filled mailto link

Dismiss → Archive with toast feedback

Archive viewer — sidebar shows archived scans; clicking opens a sheet with full findings

Critical vs High distinction — critical is now magenta with pulse glow + left accent bar; high is orange-red

Severity Guide dialog accessible from the header

Affordance improvements — "View details" label on cards, visible borders on buttons, shadow on scan button

My input/guidance after reviewing the second pass.

having just ran a scan on my own personal website, I see that "ops@www.michaelstewart.net is what you're offering as the contact. This is not correct. And the experience we're presenting, a simple button with the term "confirm" on it is weak. What does that mean in this context? Will clicking that button save that info? If it's not correct, what confidence will our user have if they click into the field and edit/update what's there? Can we provide some in-context information about where this data was found? I know for a fact that this is a bogus contact, I have no such contact on my site, so it would appear to be made up. Perhaps we need to provide, in context, a link to the scanned site (? posing as question as I'm not certain of the best course of action here) And, do we want the contact function to be line by line or issue by issue? Perhaps we need to allow our user to select from among the issue found/presented and allow them to check/select from among them so that, when the email is composed, only those issues that they've selected will be included/forwarded as a part of the contact email. In response to this observation, mock-up what our contact email might look like when a user reaches out regarding found issues.

Loveable's response:

Here's what changed:

No more fake contacts — SentinelSOC no longer fabricates email addresses. Instead, it honestly tells the user to find the contact themselves, with a "Visit site" link to the scanned URL for reference.

Batch notification — The per-finding "Notify Owner" flow is replaced by a Compose Notification panel that appears after a scan completes. Users select which findings to include via checkboxes (select all / individual), enter the recipient email once, and compose a single consolidated email.

Email preview — A "Preview Email" button opens a dialog showing exactly what the email will look like: To, Subject, and full body with all selected findings formatted clearly.

FindingCard simplified — Individual cards no longer carry notification logic; they just show details and allow archiving.

Next iteration notes:

The "checkboxes" that you've added to the selection pane are, by appearance, radio buttons. That is the incorrect visual style to present to our user -- it would be confusing, since radio buttons only allow for single-select from a list of options. Also, and make a hard and fast "rule" note of this: no em dashes are to appear in any text in this application. If we need or want to break a line, we'll use a colon. Now, after reviewing what we've got so far, I'm actually thinking that it might be handier to have a selection control within each issue, to add it to the list to send, or something along those lines. Then, we show the build list as issues are selected in the preview pane at the bottom, before the user jumps to the composition of the contact email. Our contact email should have some "boilerplate" inserted that:

1. Introduces who we are and what this email is about
2. Should clearly rank order the issues, just as we do here in the interface, so that the recipient has a clear picture of the severity, as presented by the order.

(Loveable's response)

Here's what changed:

Per-finding selection: Each finding card now has an "Include in report" / "Remove" button so you select issues individually as you review them.

Notification composer shows selected preview: The composer at the bottom displays only selected findings, rank-ordered by severity with numbered positions.

Professional email boilerplate: The email now opens with a proper introduction explaining who SentinelSOC is, what the email is about, and that it's a good-faith notification. Findings are rank-ordered from most critical to least.

No em dashes: All em dashes replaced with colons across the app (saved as a permanent rule).

No more fake radio buttons: The old checkbox-based selector is gone, replaced by clear action buttons on each card.