Do We Agree on What an "Audit" Is? Toward Standardized Smart Contract Audit Reporting

Anonymous Author(s)

Abstract

Smart contract security audits are essential for trust in in decentralized finance (DeFi), yet audit reports from different firms vary widely in scope definition, severity labels, fix verification, and report structure. These differences make it hard for developers, users, and other stakeholders to assess risk. In this paper, we address these issues by empirically analyzing over 130 audit reports from 26 leading auditing firms to uncover patterns and gaps in current practices. Using qualitative content analysis, we extract a taxonomy of 19 common properties that audit reports include (or omit). We then apply Formal Concept Analysis (FCA) to identify five distinct "report styles" used by auditors, and perform a temporal trend analysis to see if the industry is converging on certain best practices. Finally, we synthesize a feature model that specifies a minimal defensible baseline for audit reports, distinguishing mandatory sections from optional extensions to support traceability and consistent interpretation across reports. This model enables reproducible comparisons across auditors, strengthens accountability for scope definition and fix verification, and provides an evidence base to improve the quality and uniformity of smart contract audit reporting.

CCS Concepts

• **Software and its engineering** → **Empirical software validation**; Process validation; *Documentation*; • **General and reference** → *Empirical studies*; • **Security and privacy** → *Software and application security*.

Keywords

smart contracts, security audits, audit reports, decentralized finance, severity labeling, empirical study, formal concept analysis, feature model, reporting standardization, documentation

ACM Reference Format:

Anonymous Author(s). 2025. Do We Agree on What an "Audit" Is? Toward Standardized Smart Contract Audit Reporting. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym MSR).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Decentralised finance (DeFi) projects have suffered numerous smart contract vulnerabilities and high-profile exploits, resulting in major financial losses [2, 4, 15]. In the first quarter of 2025 alone, public incident trackers reported losses exceeding $1.63 billion across hacks and scams, most of which involved smart contract exploits[1]. In response, the community has developed many automated analysis tools for smart contract security, ranging from static analyzers to symbolic execution and formal verification platforms [6, 8, 18]. However, these tools have significant limitations. Their outputs are inconsistent and often include many false positives, and their coverage is incomplete [6, 15, 18]. Because of these gaps, manual code audits by specialised security firms have become widely expected before deploying high-value smart contracts [2, 4]. Today, major DeFi projects often commission one or more independent audits as part of their launch and listing processes[2].

An audit process involves expert auditors manually inspecting the contract code, often with tool support, to identify bugs or security flaws. The outcome is an audit report that describes the code's security posture, discovered issues, and recommended fixes. In principle, a thorough audit and clear report should increase transparency and trust for developers, users, investors, exchanges, and insurers. In practice, however, the quality and clarity of audit reporting vary widely across the industry [2, 4].

Unlike traditional fields such as financial accounting and software engineering, where audits and reviews follow defined standards [12–14], there is no official standard for a smart contract security audit and no shared baseline for audit reports. The term "audit" itself has different meanings to different stakeholders [16]. Each auditing firm tends to produce reports in its own style and terminology, which means that two audits of the same code can yield divergent findings and presentations. Important elements such as the exact code version and components reviewed, the severity rubric used, the methodology and tools employed, and whether fixes were verified are communicated inconsistently from one report to another [2, 4]. This lack of uniformity leaves stakeholders unable to reliably compare results or understand true risk, and it can give a false sense of security when a project is labelled "audited." Several high-profile exploits have followed earlier audits. For example, Euler Finance and KyberSwap were both exploited for tens of millions of dollars after prior reviews[3].

The absence of standards has led to calls for a "Minimum Viable Audit" or similar baselines that define the minimum expected content of a smart contract audit. Emerging efforts have offered a

Conference acronym MSR, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

[1] Immunefi, "Crypto Losses in Q1 2025," Mar. 2025: https://downloads.ctfassets.net/t3wqy70tc3bv/kicQd50NcNkEQyA1QmpOB/b365451e130c1b1c0cbe55c191a22d41/Immunefi_Crypto_Losses_In_Q1_2025.pdf.

[2] See, for example, OpenZeppelin's readiness guidance: https://learn.openzeppelin.com/security-audits/readiness-guide. Exchange due-diligence articles also emphasise independent audits as standard practice.

[3] Euler Finance: Chainalysis, "Euler Finance Flash Loan Attack Explained," Mar. 15, 2023: https://www.chainalysis.com/blog/euler-finance-flash-loan-attack/. Euler later coordinated recovery efforts [7]. KyberSwap Elastic: official post-mortem, Feb. 7, 2024, reporting ∼ $56M affected: https://blog.kyberswap.com/post-mortem-kyberswap-elastic-exploit/. See also [25].

checklist of security requirements[4] and community guides on audit preparation and scope[5]. Although these efforts help, they do not provide a standardized report template.

To make progress in this direction, we first need a data-driven understanding of how audits are currently performed and documented. In this paper, we take a foundational step by performing the first systematic analysis of smart contract audit reports as a body of knowledge. We curate a dataset of approximately 130 publicly available audit reports from 26 major auditing firms (spanning 2018–2025) and examine their contents in depth. Our goal is to empirically characterise the state of practice and identify opportunities for improvement and standardisation. We pose the following four Research Questions (RQs) to guide our study, each addressing a specific aspect of audit reporting:

- **RQ1: What *properties* do smart contract audit reports present, and what knowledge do they communicate?** We aim to gain insights into what auditors convey to stakeholders and identify the essential properties that appear consistently across different firms. To achieve this, we analyse a stratified sample of reports using qualitative content analysis, standardising synonyms and section aliases, and compiling straightforward prevalence and co-occurrence statistics for each attribute. The outcome is a comprehensive taxonomy and codebook of standard properties, complete with an alias map and frequency data that facilitate easier comparison of reports.
- **RQ2: Do audit reports form recognisable *families* of structure?** We seek to determine whether reports group into family styles that represent various reporting philosophies. To achieve this, we construct a presence-absence matrix based on the properties identified in RQ1 and apply Formal Concept Analysis to generate concept lattices and implication rules, supplemented by lightweight clustering when beneficial. The anticipated outcome is a concise collection of report families characterised by defining features and notable omissions, accompanied by straightforward rules that facilitate the identification of each family in practice.
- **RQ3: Are reporting practices converging on key best practices over time?** We aim to assess whether the industry is trending towards standardised practices and to identify which elements are still lacking. Our longitudinal analysis, spanning from 2018 to 2025, monitors adoption rates of essential practices. We measure variations across firms and report on trends, effect sizes, and whether each practice shows evidence of convergence. This will enable stakeholders to easily determine areas of progress and those where gaps remain.
- **RQ4: How can a minimal baseline be formalised as a *feature model*?** We aim to turn the evidence from RQ1–RQ3 into an actionable reporting baseline. We synthesise the findings into a structured feature model that separates mandatory baseline elements from optional enhancements and captures conditional features. We validate the model by mapping held-out reports to its features and by checking internal consistency. The outcome is a report-centric feature model—essentially a concise checklist and template—that improves clarity, comparability, and traceability across firms.

Our analysis shows that audit reports share a predictable core, yet key properties remain inconsistent. Scope, severity definitions, methodology, and verification of fixes are frequently incomplete, and we observe only limited convergence over time with a few clear exceptions. The main contributions of this paper are:

- We present a curated multi-firm dataset of smart contract audit reports and a taxonomy of their sections and properties, shedding light on current industry practice (RQ1).
- We empirically identify common structures and styles in these reports, as well as significant inconsistencies across auditing firms (RQ2).
- We evaluate how certain reporting practices have evolved over the past several years (RQ3).
- We propose a data-informed baseline for audit report content and format, expressed as a report-centric feature model, with the goal of improving the consistency, traceability, and comparability of future audit reports (RQ4).

2 Observed Gaps in Current Reporting Practices

This section summarizes four recurring problems in current smart contract audit reports. We cite representative cases to illustrate their impact, focusing on systemic issues rather than individual blame. These observations, echoed by industry debates about the effectiveness of audits [26, 28], motivate our subsequent analysis and support the need for reporting standards.

Scope definition and coverage. Many audit reports do not clearly define their scope. Critical details like exact code versions, repository commits, and in-scope modules are often missing. As a result, stakeholders cannot be sure what was actually reviewed, which can create a false sense of completeness. Unexamined components may contain vulnerabilities that compromise the entire project. Industry guidance stresses the need to bind scope to specific files and commit hashes [5]. For example, Team Finance lost about $14.5 M in 2022 because the vulnerable migration function was introduced after the initial audit and was therefore never in scope for the auditor [9, 32]. Likewise, the $12 M Cork Protocol hack in 2025 led to public disputes over what code was "out of scope," with leaked materials showing missing commit hashes and boundaries in at least one firm's documentation [29, 30]. Such ambiguity allows auditors to deflect responsibility and leaves protocol teams and users unsure which parts of the system were actually vetted. On the other hand, a clear scope helps auditors highlight what was in scope and protects their reputation. When scope boundaries and commit identifiers are not explicit, the audit's value is undermined.

Severity definition and labeling. Severity labels for findings vary widely between firms, and the criteria for each level are not aligned [28]. The same flaw might be labeled "Low" by one team and "High" by another, which complicates prioritization. Leading guidance warns about "severity inflation" and emphasizes careful definitions [5]. Real incidents show the danger. In the BetterBank case in 2025, a serious vulnerability was misclassified as low severity and was

[4] OWASP SCSVS project: https://owasp.org/www-project-smart-contract-security-verification-standard/.
[5] OpenZeppelin Audit Readiness Guide: https://learn.openzeppelin.com/security-audits/readiness-guide.

later exploited to drain roughly $5 M [21]. In another case, the 2021 Uranium Finance exploit stemmed from an arithmetic bug that an earlier audit initially labeled low severity. The team later recognized it as critical, yet the live code remained vulnerable and an attacker stole about $50 M [10]. Without common severity definitions and a brief rationale for each finding, teams and users cannot reliably estimate which issues are truly urgent.

Fix verification. Audit reports often capture a single point in time and do not track whether identified issues were later fixed and verified. This creates an accountability gap between the report and the deployed code. In Uranium's case, developers recognized the bug and coded a fix, but the unfixed version went live and was exploited shortly before deployment of the patched release [10]. Similarly, BetterBank's auditors provided a patch, but it was not fully implemented before launch [21]. Best practice is to document the project team's response to each finding and whether a fix was validated by the assessor [5]. Some firms conduct explicit "fix reviews" and publish follow-up artifacts, for example Meson Protocol's fix review by Trail of Bits [19, 33]. Without such verification, stakeholders lack evidence about which vulnerabilities remain live, and audits become snapshots rather than part of a secure development lifecycle.

Methodology and report structure. The depth and structure of audit reports vary widely. Some provide an architecture overview, threat assumptions, manual analysis steps, and tool outputs, while others provide only a list of issues with little context. This inconsistency makes it hard to judge quality or compare reports. A clear illustration is the Beanstalk governance attack in 2022. Although the protocol had been audited, the auditor later stated that the specific governance code exploited "was not audited before release" [11]. When reports are not tied to specific commits and do not clearly state their limitations, even careful readers can misinterpret the level of assurance. Inconsistent structures also make it easy to omit important details, for example whether the code was unit tested or formally verified.

Need for reporting standards. These gaps in scope clarity, severity consistency, fix follow-up, and methodology reduce transparency and trust in the audit process. Multiple audits are no guarantee of safety [26]. In this paper we address these issues by proposing a feature model for smart contract audit reports. The model captures essential elements and common variations observed in current practice and provides a foundation for more consistent and complete reporting. By formalizing the common elements of audit reports, the feature model aims to guide auditors in producing outputs that are easier to navigate and compare.

3 Methodology

In this section, we outline our research design and methods for addressing the four research questions. We combined qualitative content analysis with quantitative pattern mining and trend analysis. First, we built a dataset of audit reports via stratified sampling and coded their contents (RQ1). We then identified structural patterns in report composition using Formal Concept Analysis (RQ2). Then, we analyzed how key reporting practices have evolved over time (RQ3). The following subsections detail our data collection and the specific approach for each RQ. Figure 1 presents an overview of our study methodology.

3.1 Data Collection and Sampling

Identifying audit providers. We first compiled an initial pool of audit service providers from public directories. At the time of our study, Etherscan's *Smart Contracts Audit & Security* directory listed 92 firms[6] offering smart contract auditing services. Many of these firms either publish no audit reports or only a few. To focus on mature and transparent practices, we targeted the top quartile of providers by report volume. In particular, 26 firms had published at least 200 audit reports, putting them in roughly the 75th percentile of output. We treated these 26 firms as our target population, as they maintain substantial public archives and represent the most established auditors.

Stratified sampling design. From each of the 26 firms, we sampled approximately 5–6 reports, yielding a final dataset of 160 reports. By allocating an equal number of reports per provider, we avoid over-representing any single prolific firm. This stratified design balances depth and breadth: with multiple reports from each auditor, we capture within-firm patterns, and by sampling across 26 auditors, we capture cross-firm diversity. Within each firm's set, we further diversified the selections by year (spanning 2018–2025), platform (e.g., Ethereum, Solana), and project domain (e.g., DeFi, NFTs, infrastructure, etc.) to maximize coverage of different contexts, if available. Rather than strictly weighting by year or other factors, we ensured that the sample includes a broad mix of report characteristics. *Sampling was not stratified by year; we therefore report per-year sample sizes (reports and unique firms) in the Results overview and include* `year` *as a factor in RQ3 analyses to mitigate imbalance.*

3.2 Identifying Common Report Properties (RQ1)

Qualitative Coding. To identify the major information properties presented in audit reports, we conducted a qualitative content analysis of all 160 reports in our dataset. Two researchers first jointly coded a pilot set of 10 reports to develop a shared codebook and initial taxonomy of report content. Once they reached consistent interpretations, the remaining reports were divided for independent coding. For each audit firm, the reports were divided between two coders to ensure that no single coder analyzed all reports from any one provider, minimizing potential biases or blind spots related to a specific firm's reporting style. Each coder extracted all report section headings and used them (along with surrounding text) as evidence to assign labels representing the content conveyed. In essence, we treated each distinct major section or content element as a candidate property of the report. The goal was to label semantically distinct components of the reports based on their function rather than their literal title. For example, sections titled "Audit Summary" or "Executive Overview" would both be labeled as an *Executive Summary* property if they served the same purpose. Throughout this open-coding process, the coders met regularly to

[6] https://etherscan.io/directory/Smart_Contracts/Smart_Contracts_Audit_And_Security

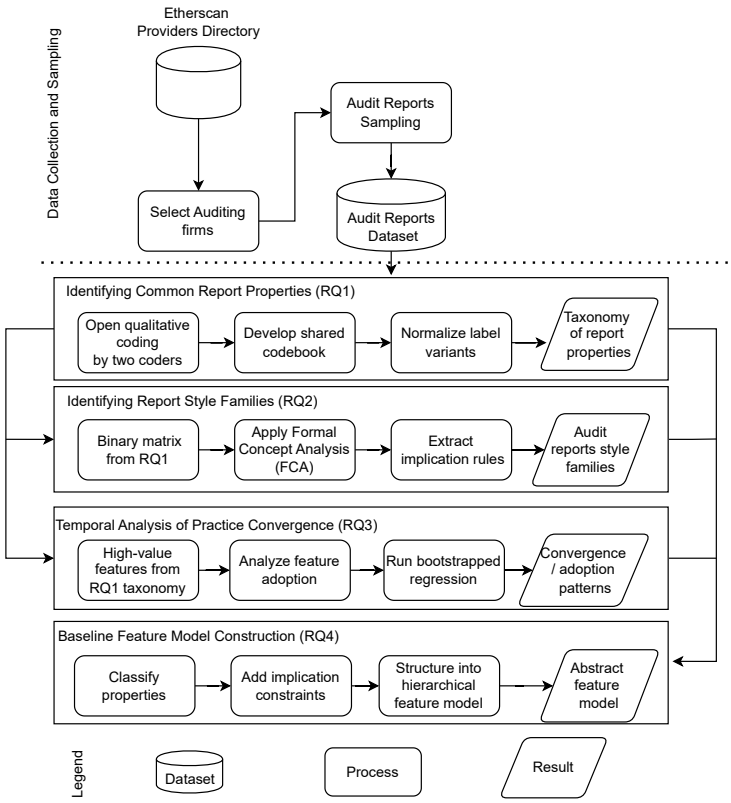

Figure 1: Study methodology

compare findings, reconcile any discrepancies in labeling, refine the definitions of each category, and update the codebook accordingly. We measured intercoder consistency on a subset of overlapping reports. Treating the presence or absence of each property as a binary variable, we computed agreement using Jaccard similarity. The mean Jaccard index was 0.86, indicating a high level of agreement on which properties were present in a given report. All coding disagreements were discussed and resolved through consensus, and any ambiguous cases led to further clarification in the codebook. *In addition to Jaccard, we computed Cohen's κ and Krippendorff's α (binary) on the overlap; coefficients and overlap details are provided in the artifact.*

Taxonomy Construction. Using the coded data, we iteratively constructed a unified taxonomy of report properties. We grouped synonymous or closely related section labels based on their communicative purpose. For instance, labels like "Audit Summary" and "Executive Overview" were merged into a single *Executive Summary* category. We maintained an alias map to normalize variant terminology to a set of property names.

3.3 Identifying Report Style Families (RQ2)

To determine whether audit reports group into recognizable structural families, we analyzed the presence/absence patterns of the extracted properties from RQ1. We constructed a binary matrix with one row per report and one column per canonical property. An entry is 1 if the report contains that property, or 0 if it does not. We then applied Formal Concept Analysis (FCA) to this matrix to uncover natural groupings of reports and properties. FCA computes a lattice of formal concepts, where each concept consists of an extent (a set of reports) and an intent (the set of properties common to those reports). Intuitively, each concept represents a cluster of reports that share a particular combination of properties. By examining concepts with multiple reports, we can identify recurring combinations of report properties that may correspond to meaningful report styles. In our analysis, we filtered out trivial patterns to focus on salient ones. We ignored concepts that only contained a single report (since those represent idiosyncratic cases). We then extracted implication rules from the concept lattice using the Duquenne–Guigues (DG) basis. These rules have the form "if property X is present, then property Y is also present," capturing strong associations between report elements. We selected a minimal

set of high-confidence rules that summarize the most important co-occurrence relationships among properties. Each rule was checked for plausibility against domain knowledge. To avoid any single firm dominating a pattern, we ensured that the key concepts and rules we reported included reports from multiple auditors. *We used an open-source FCA implementation (version pinned in the artifact), retained concepts with extent* ≥ 3 *and intent* ≥ 3*, and reported DG rules with support* ≥ 0.15 *and confidence* ≥ 0.90*; highlighted patterns include reports from multiple auditors.* From the concept lattice and rules, we identified a small number of candidate "report style families." We focused on the concepts with high support (appearing in many reports) and interpretable intents. For each main combination of properties, we examined which known firms' reports fell into that group and how the combination reflected a particular reporting philosophy (e.g., a comprehensive style vs. a minimalist style). The outcome of RQ2 is a characterization of a few distinct report style families, each defined by a signature set of properties. These families illustrate the major documentation philosophies employed by auditors in the industry.

3.4 Temporal Analysis of Practice Convergence (RQ3)

We next examined how key reporting practices have changed over time to see if the industry is converging on certain best practices. We selected a subset of high-value report properties that are most directly tied to clarity, transparency, and assurance. These particular elements are frequently emphasized in practitioner guidelines and community discussions about audits. For each chosen property, we tracked its adoption in our sample year by year from 2018 through 2025. Concretely, we calculated two metrics for each property in each year: (1) the overall adoption rate, defined as the percentage of all audit reports in that year that include the property; and (2) the per-firm adoption rate, defined as the percentage of auditing firms (each firm contributes at most one "yes" per year, to avoid over-counting firms with many reports) that produced at least one report containing the property in that year. To assess convergence, we looked for trends in both metrics over the 2018–2025 period. We interpret a feature as "converging" if its overall adoption is increasing while the variation across firms is decreasing. In practice, if more and more reports include a given element and most firms are adopting it, that suggests movement toward industry-wide standardization. We quantified cross-firm variation (dispersion) in two ways: by computing the standard deviation of the per-firm adoption rates each year, and by measuring the gap between the overall adoption and the fraction of firms adopting. Finally, we tested whether observed trends were statistically significant. We performed a bootstrapped linear regression for each property's adoption trajectory across the 8-year span. Using 10,000 bootstrap resamples, we estimated confidence intervals for the slope of the adoption rate over time. We considered a trend significant if the 95% confidence interval of the slope did not include zero. For a property to be deemed converging, we required that its overall adoption slope is significantly positive ($p < 0.05$) and simultaneously its dispersion (variation across firms) shows a significant downward trend. *To account for repeated measures within firms, we additionally used a firm-level block bootstrap, and we control for multiple testing across properties via Benjamini–Hochberg FDR (we report raw and adjusted p-values).*

3.5 Baseline Feature Model Construction (RQ4)

In RQ4, we translate our empirical insights into a prescriptive feature model that defines a minimal baseline for smart contract audit reports. We treated each property identified in RQ1 as a candidate feature in the model. We then refined the feature set by merging or splitting certain items based on their frequency and distinctiveness. For example, if two properties always appeared together and served a single purpose, we considered combining them; conversely, if a property was very broad, we split it into sub-features for finer granularity. We only retained optional sub-features that appeared in multiple reports. Next, we classified features as mandatory vs. optional based on the empirical prevalence and convergence results. Roughly, properties observed in >90% of reports were designated as mandatory baseline features, whereas those present in less than 50% of reports (or clearly not yet standard) were labeled optional extensions. Additionally, we incorporated conditional relationships using the implication rules derived in RQ2. We organized the feature model hierarchically: an *Audit Report* is the root element, which is composed of major feature groups. Each group contains specific features or sub-features. We specified the model in a machine-readable form and performed sanity checks: a satisfiability check to confirm that there is at least one valid report configuration, and a soundness check to verify that every actual report in our study can be mapped to a valid combination of features in the model.

4 Results

We analyze audit reports from 26 providers spanning **2018–2025**. Solidity and EVM platforms dominate where metadata is disclosed. All CSVs, scripts, and figures are in the project repository.

4.1 RQ1: Common Audit Report Properties and Prevalence

Through qualitative coding of 160 audit reports, we identified **19 canonical report properties**. These represent the major section types that commonly appear in audit reports. Table 1 shows a sample of these properties and what they include. The full list is in the project repository. Figure 2 shows their prevalence.

Many fundamental sections appear in the great majority of reports. As Figure 2 shows, nearly all audits have **Findings** (by definition), and most provide an **Executive Summary**, define **Scope**, describe the **Methodology**, and offer **Recommendations**. A majority also track **Status of Findings** and mention **Tools** used. For example, about 94% of reports include an executive summary, 90% define scope, and 84% describe the audit methodology. These figures reflect a shared consensus that audit reports should at least explain what was audited, what was found, and how to fix it.

By contrast, certain sections are much rarer. Only 26% of reports define a **Severity Legend** explaining what their severity labels mean. This gap reduces the comparability of ratings across providers. Even more rarely, just 0.6% include **Versioning/References** to the exact code version audited (e.g., commit hash), which undermines traceability. These two are among the most important omissions.

Table 1: Sample canonical properties used in our taxonomy.

Property	Definition / What It Covers
Scope	Lists in-scope contracts or components; optionally mentions exclusions.
Methodology	Describes techniques used (manual review, fuzzing, static analysis, etc.).
Findings	Issue descriptions with severity labels and contextual info.
Recommendations	Remediation advice for each issue or class of issues.
Status of Findings	Indicates whether each finding was resolved, acknowledged, or pending.
Tools & Automation	Lists tools used (e.g., Slither, Echidna, MythX) with configurations.
Severity Legend	Defines the severity scale used (e.g., what qualifies as High vs. Medium).
System Overview	Describes architecture or code layout to orient the reader.
Threat/Trust Model	States key assumptions or adversary capabilities.
Versioning/References	Links to repo and gives commit hash or tag for reproducibility.

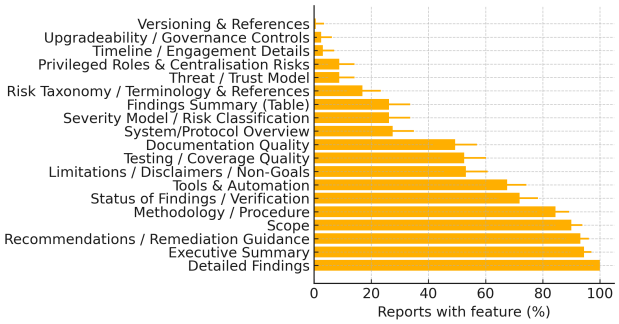

Figure 2: Prevalence of the 19 properties. Bars show percent of reports containing each property.

Takeaway (RQ1): Smart contract audit reports follow a consistent template: they nearly all describe the findings, what was in scope, the audit method, and how to fix issues. However, many fail to define severity criteria or specify the audited code version. These omissions limit comparability and reproducibility.

4.2 RQ2: Report Style Families and Co-Occurrence Patterns

Beyond individual properties, our analysis found that audit reports cluster into a few **style families** based on which combinations of properties they include. Using Formal Concept Analysis on the presence/absence of the 19 properties, we identified overlapping groupings of report styles. Before describing the families, Table 2 shows a few representative co-occurrence rules (implications) among sections.

These patterns suggest that documentation practices tend to appear in bundles. For example, when assumptions are specified, privileged roles are usually enumerated as well. Similarly, a report that defines severity levels is likely to classify its findings under a formal taxonomy.

Table 2: Representative co-occurrence implications.

If a report includes...	It almost always includes... (confidence)
System Overview	Methodology section (96%)
Severity Legend (Criteria)	Risk Taxonomy (93%)
Executive Summary	Status of Findings (91%)
Threat/Trust Model	Privileged Roles (90%)

We identified five style families satisfying three criteria: *high support*, *strong implications*, and *domain plausibility*. Table 3 summarizes each family and its rationale.

Table 3: RQ2 style families: core sections and rationale.

Family	Core sections	Rationale
Engineering Methods (~30%)	Methodology, Tools, Severity Legend	Process transparency, aligns with engineering best practice
Action and Remediation (~95%)	Findings, Recommendations	Clear focus on actionable remediation
Executive Level (~43%)	Executive Summary, Findings Table, Scope	Summary for stakeholders, management readability
Governance Focused (~18%)	Privileged Roles, Upgradeability notes	Emphasis on administrative or upgrade risk
Compliance (~22%)	Risk Taxonomy, Disclaimers	Formalism and disclaimers, compliance style

Summary (RQ2): Most audit reports share a remediation-first structure. Some also prioritize executive readability, technical methodology, compliance formalism, or governance assumptions. These families are not exclusive, but reflect distinct emphases. The stable co-occurrence patterns help explain how sections group together and informed the design of our baseline model in RQ4.

4.3 RQ3: Temporal Trends in Reporting Practices (2018–2025)

Over the 2018–2025 period, we observed a clear trend toward more complete and standardized reports. Our analysis tracked eight features: Executive Summaries, Methodology, Tools, Status tracking, Disclaimers, Severity Legends, Threat Models, and System Overviews. (We excluded Findings and Recommendations since they appear in nearly all reports.)

Several practices grew significantly. For example, Methodology sections appeared in just 40% of reports in 2018 but rose to 93.8% by 2025. Tools usage rose from 40% to 75%. Status tracking reached 81%

in 2025. Executive Summaries became nearly universal. Disclaimers improved but remained under 50% of reports.

Some features improved only slightly: threat models rose from near 0 to 15–20%, and severity legends hovered around 25–30%. This suggests ongoing gaps in risk transparency.

Adoption became more consistent across firms. In early years, some firms included certain sections consistently while others did not. By 2025, nearly all firms in our dataset had adopted the major features in at least one report. Figure 3 shows report- and firm-level adoption.

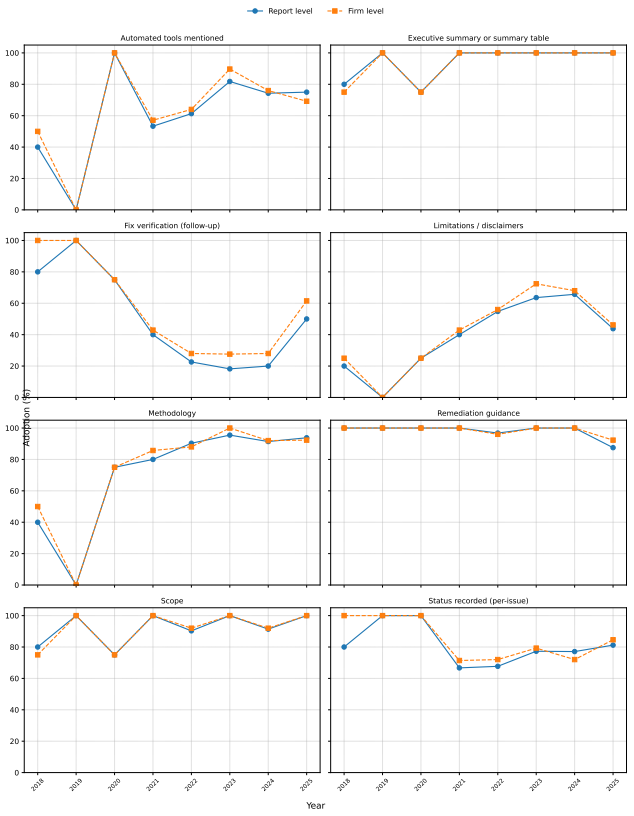

Figure 3: Adoption trends for key report features (2018–2025).

Summary (RQ3): Smart contract audit reports have become more thorough and uniform. Most now include methodology, tools, issue status, and executive summaries. Although risk transparency (e.g., threat models, rating definitions) remains inconsistent, the field is converging toward a stable reporting baseline.

4.4 RQ4: Baseline Feature Model for Audit Reports

Based on our findings, we propose a risk-based baseline model for audit reports. Elements are marked Mandatory [M], Optional [O], Strongly Recommended [O+], or Conditionally Mandatory [M*]. A section is **[M]** if its absence harms clarity, reproducibility, or stakeholder trust. Optional elements are helpful but not critical.

I. Scope & Context

- **I.1.** Scope definition [M] (R1): What was audited and what was excluded.
- **I.2.** System overview [O]: Optional description of architecture or design.
- **I.3.** Threat/trust model [M] (R8): Key assumptions, adversary capabilities.
- **I.4.** Privileged roles [O]: Admins, upgraders, multisigs, if applicable.

II. Methodology & Tooling

- **II.1.** Methodology [M] (R4): How the audit was conducted.
- **II.2.** Tooling details [M] (R4): Tools and configs used.
- **II.3.** Formal methods [O]: Optional details on model/specs used.

III. Findings & Risk Communication

- **III.1.** Detailed findings [M] (R6): Narrative description of each issue.
- **III.2.** Severity ratings & criteria [M] (R5): What qualifies as Critical, High, etc.
- **III.3.** Remediation recommendations [M] (R6): Suggested fix per issue.
- **III.4.** Status of findings [M] (R6): Fixed / acknowledged / open.
- **III.5.** Risk taxonomy [O]: Classification by issue type.
- **III.6.** Summary of findings [O]: High-level overview table.

IV. Traceability & Reproducibility

- **IV.1.** Code reference [M] (R2): Repo + commit or tag.
- **IV.2.** Deployment details [M*] (R3): On-chain addresses, compiler.
- **IV.3.** Reproduction guidance [M*] (R7): PoC or steps for High/Critical issues.
- **IV.4.** Code locations [O]: File/line references for findings.
- **IV.5.** Audit timeline and dates [M] (R9): Audit window + report version.
- **IV.6.** Machine-readable annex [O+] (R10): JSON or similar schema.

Summary (RQ4): The model codifies a practical reporting baseline. It covers scope, audit method, findings, severity criteria, and traceability. Some currently rare but important features (e.g., version hashes, rating definitions) are promoted to mandatory status to improve reproducibility and comparability.

5 Discussion

This section evaluates how current smart contract audit reports align with our baseline model, compares that model with industry best practices, and examines variation in severity schemes. We then propose a short rubric to improve consistency and discuss how report style influences the clarity of severity communication.

5.1 Conformance to the Baseline Feature Model (RQ4)

To assess consistency in audit reporting, we compared 160 audit reports from 19 firms against the baseline feature model proposed in RQ4. The baseline defines mandatory and optional components intended to improve transparency, traceability, and actionable value. We recorded the presence or absence of each of the model's 15 core features for every report, using the manually tagged dataset from RQ2 aligned to the RQ4 taxonomy.

Overall, conformance was low. Only one report (0.6%) included all seven *mandatory* features. Major technical sections were common: *Detailed Findings* appeared in 100% of reports, *Scope* in 90%, and *Methodology* in 84%. Important process elements were often missing. Only 26% of reports defined *Severity*, and just 0.6% provided *Versioning or Traceability*. *Limitations/Disclaimers* appeared in 53% of reports and *Verification Status* per issue in 72%.

Some optional features were widely adopted. An *Executive Summary* appeared in 94% of reports, which suggests de facto standardization. Domain context features were rare. Only about 9% of reports included a *Threat Model* or listed *Privileged Roles*. On average, a report contained four of the seven mandatory features. There is significant room to align practice with the proposed baseline.

5.2 Comparison with Industry Best Practices

The baseline aligns with widely accepted guidance for security assessment reporting. Elements such as *Scope*, *Methodology*, and *Remediation Guidance* are consistent with OWASP [23], NIST [20], and published report templates [31]. The model adds value by making risk communication and traceability explicit. Clear *Severity Legends* and explicit *Versioning* are recommended in professional guides, yet adoption is uneven. Including these features improves stakeholder understanding and supports reproducibility.

The baseline also captures domain-specific expectations for blockchain audits, for example *Trust Assumptions* and *Upgradeability Risks*. These sections are not always necessary but they reflect emerging best practices for system-level context. In summary, the baseline consolidates essential and advanced features from general cybersecurity guidance and domain-specific auditing experience. Broader adoption would improve consistency and interpretability across the industry.

5.3 Variations in Severity Level Definitions Across Firms

Firms do not use a uniform severity scale or terminology. Most use tiers such as *Low*, *Medium*, and *High*, often extended with *Critical* and *Informational*. Some add labels for non-security items like *Gas Optimization* or *Best Practice*. As a result, a finding labeled "Low" in one report can match "Minor" in another. Comparisons across firms are difficult.

A few firms use atypical schemes. Trail of Bits[7] uses eight custom categories that emphasize design strength rather than only exploit risk. Hacken[8] employs a quantitative formula that mixes factors such as likelihood, impact, complexity, and exploitability. Many other firms keep qualitative labels without publishing definitions. In those cases, terms like "High" or "Medium" lack context for the reader.

Some firms anchor their ratings in established models. ChainSecurity[9] and SolidProof[10] use a CVSS v3 style scoring approach that maps numeric scores to Low through Critical. PeckShield[11] follows an OWASP-like idea where risk combines impact and likelihood. These standards make the meaning of each level clearer. Many other firms do not cite a model and rely on expert judgment without a stated rubric, which complicates cross-report comparisons.

We propose a simple rubric that any audit can include. Each severity level should be explained in terms of three factors:

- **Impact**: expected consequence if exploited, for example loss of funds or control.
- **Likelihood**: probability of occurrence or exploit success, considering ease of exploit and preconditions.
- **Urgency**: how quickly the issue should be addressed, given deployment plans and user risk.

A concise mapping then follows:

- **Critical**: very high impact and very high likelihood. Fix immediately or pause deployment.
- **High**: high impact or high likelihood. Fix as a top priority before release when possible.
- **Medium**: moderate impact or lower likelihood. Fix in the next planned update.
- **Low**: small impact or very low likelihood. Fix when convenient or accept with justification.
- **Informational**: no direct security impact. Improves clarity or efficiency.

Each finding should include a one-line rationale that ties facts to the assigned level. This approach is compatible with CVSS and OWASP thinking because it separates impact and likelihood and adds urgency to guide action.

We recommend two simple practices for all reports. First, include a short legend that defines each severity level using impact, likelihood, and urgency. Second, add a one-line rationale under each finding that explains the level. If non-security tags such as *Gas Optimization* or *Best Practice* are used, keep them separate from risk levels. These steps fit both executive and technical styles. Making definitions and reasoning explicit improves consistency and usability across audits.

6 Evaluation

This section evaluates our taxonomy (RQ1) and baseline model (RQ4). We first evaluate completeness and soundness on 160 reports using manual coding with high agreement and constraint checks. We then test generalizability on 14 additional auditing firms and measure baseline compliance.

6.1 Taxonomy Soundness and Completeness

Purpose and approach. We evaluated whether our 19-category taxonomy is complete and sound for classifying smart contract audit reports. Completeness asks if all substantive report sections

[7] https://www.trailofbits.com
[8] https://hacken.io
[9] https://chainsecurity.com
[10] https://solidproof.io
[11] https://peckshield.com

can be mapped to the taxonomy. Soundness asks if the taxonomy admits valid configurations only and contains no contradictory or dead features. Two researchers independently coded a subset of reports and achieved high agreement (Jaccard similarity ≈ 0.85). The agreed rules were then applied to all 160 reports, and constraint checks were run on the resulting feature sets.

Completeness results. Across roughly 1,400 identified sections, fewer than 15 sections (under 2%) could not be mapped. Unmatched items were cosmetic or auxiliary, such as brief *Conclusion* notes, appendices, or auditor company blurbs. No new substantive category was required. This indicates that the 19 categories cover essentially all observed report content.

Soundness results. All encoded structural constraints held across the corpus. For example, whenever a *Findings Summary* table appeared, a *Detailed Findings* section was also present (42 of 42 cases). No report violated dependency rules and no feature was dead. Rare features were present but meaningful, which supports keeping them in the taxonomy.

6.2 Out-of-Sample Generalizability and Baseline Compliance

Sampling and coding. We evaluated 14 additional reports from firms outside the initial 26 to test generalizability and to check baseline compliance. All selected providers had fewer than 200 public audit reports at the time of sampling. We chose one recent report per provider that met our inclusion criteria. Examples include Cyfrin[12], Veridise[13], ImmuneBytes[14], and Pessimistic[15]. We mapped each report to the 19 categories and recorded the presence of baseline features from RQ4.

Generalizability. All sections in the 14 reports mapped to the 19 categories. No new category emerged. On average, a report covered 9.4 of 19 categories, which is consistent with the heterogeneity observed in-sample.

Baseline compliance. All reports contained *Detailed findings* (14 of 14) and a defined *Severity model* (14 of 14). Most included *Scope* and *Methodology* (12 of 14 each) and *Disclaimers* (10 of 14). An *Executive or findings summary* appeared in 11 of 14. The most frequent gaps were *Version reference* such as a commit hash (6 of 14) and *Issue status tracking* for fix verification (5 of 14). Using the strict baseline definition for mandatory features, 2 of 14 reports were fully compliant.

Takeaways. The taxonomy generalizes beyond the original sample and remains complete and sound. Baseline compliance is improving in core narrative elements but remains weak on traceability and verification. The most impactful fixes are to include a clear version reference and per-issue status updates, which directly improve reproducibility and accountability.

7 Related Work

In traditional software engineering, an *audit* means an independent, unbiased review by third-party experts to verify that products or processes meet plans, standards, or contractual requirements [12].

[12]https://www.cyfrin.io
[13]https://veridise.com
[14]https://immunebytes.com
[15]https://pessimistic.io

Table 4: Baseline feature adoption in 14 out-of-sample reports.

Feature	Adoption
Detailed findings	14/14 (100%)
Severity model	14/14 (100%)
Scope	12/14 (85.7%)
Methodology	12/14 (85.7%)
Disclaimers	10/14 (71.4%)
Findings summary (optional)	11/14 (78.6%)
Version reference	6/14 (42.9%)
Issue status tracking	5/14 (35.7%)

Formal standards such as IEEE 1028 define roles, objectives, and checklists for such audits [12]. In contrast, a "smart contract audit" in blockchain is usually an informal security review, not a formal compliance assessment. There is *no official standard* definition or certification for smart contract audits [27], and calling these reviews "audits" is mostly industry shorthand for a paid security review [27].

Guidelines and standards. Community efforts are starting to shape best practices. OpenZeppelin's *Audit Readiness Guide* defines a smart contract audit as a structured review by experts that produces a report listing vulnerabilities and recommended fixes [22]. The draft *OWASP Smart Contract Security Verification Standard (SCSVS)* similarly aims to standardize secure development and auditing practices, including known issues like reentrancy and economic attacks, and helps align expectations across teams [24]. Regulators have also begun to explore certification. A 2024 French ACPR-AMF working group proposed rules for certifying smart contracts, such as requiring public code, reproducible builds, least privilege, and version-specific certification [1].

Empirical studies. Bhambhwani and Huang [3] found that DeFi protocols audited by more or higher-quality firms attract more value and experience price gains after their first audit. Landsman *et al.* [17] analyzed thousands of audit reports and found better outcomes when top-tier firms audited a protocol, though they saw little evidence that audits prevent future hacks. These studies treat audits as events and use broad outcome metrics. In contrast, our study examines *what* information is communicated in audit reports and how reporting practices vary across firms.

Platform-specific work and datasets. Yanovich *et al.* [35] studied audits on the TON blockchain and built a security checklist from 34 reports. The DAppSCAN dataset [34] offers a large collection of Ethereum audits to evaluate bug-finding tools. These works show the value of collecting audit data but focus on bugs rather than how the reports are structured. Our study collects audit reports across many firms and blockchains, then analyzes their format and organization.

Comparison with our work. Prior studies treated audits as binary events or summarized their findings. No existing work has systematically studied the *content* of audit reports across firms. We fill this gap by building a taxonomy of common report sections and identifying reporting styles across the industry. We also study

whether practices are becoming more consistent. Finally, we propose a baseline model to guide the content of future audit reports. Our approach works across multiple firms and chains, offering broader insights than ecosystem-specific efforts.

8 Threats to Validity

Our study aims to provide a data-driven foundation for improving smart contract audit reporting; nonetheless, several threats to validity must be acknowledged. We structure our discussion around construct, internal, external and reliability considerations.

Construct validity. We operationalise reporting practice by coding the*presence* or*absence* of 19 canonical sections in audit documents. This coarse-grained representation may overlook nuances such as the depth or quality of a section and might conflate similarly named but conceptually distinct content. We mitigated this threat through iterative open coding and the use of an alias map to unify synonyms, but some misclassification risk remains—especially for reports with unconventional structures or labels. Furthermore, our definition of "practice convergence" is based on adoption rates of selected features over time; alternative operationalisations (e.g., weighting by issue severity or by audit size) might yield different trends. Future work could incorporate richer content analysis or natural language processing to capture qualitative differences between reports.

Internal validity. Our sampling method stratifies across 26 leading auditing firms, selecting approximately five reports per provider. This design balances coverage and feasibility but may introduce bias if some firms' public portfolios are curated or if our sample coincides with particular client types. The causal interpretation of temporal trends is also limited: year-to-year changes in reporting practice could reflect evolving client demands or shifts in the auditor population rather than true convergence. We do not claim that the identified report families cause better security outcomes; our focus is descriptive. In our Formal Concept Analysis, we enriched the presence–absence matrix using textual signals to reduce false negatives, but this heuristic may introduce false positives.

External validity. We restrict our data to publicly available audit reports written in English. Many audits are private, under non-disclosure agreements. Practices in those contexts may differ. Consequently, our findings may not generalise to all smart contract audits or to other ecosystems. Similarly, we exclude audits performed solely by automated tools as these represent alternative assurance mechanisms whose reporting conventions may vary. While our sample spans 2018–2025, the DeFi landscape evolves rapidly, and future audits may adopt new conventions in response to regulation or novel attacks.

Reliability. Coding was performed manually by two authors using a common codebook. Inter-rater reliability was high, and disagreements were resolved through discussion, but subjective judgement is inherent in qualitative content analysis. All data, code and intermediate artefacts are made available in our replication package to support independent verification. Nonetheless, researchers replicating our study might make different coding choices or interpret section boundaries differently. Our feature model encodes mandatory and optional elements based on observed frequencies, and alternative thresholds or rule sets could produce different feature hierarchies. We encourage future work to validate our model with practitioners and to extend it with additional features or logic constraints.

9 Conclusion

Smart contract audits are now a de facto requirement for high–value deployments, yet our study shows that what gets called an "audit report" varies widely in both content and clarity. By qualitatively coding *160* reports from *26* firms, we distilled a taxonomy of *19* recurring report properties (RQ1), uncovered stable co-occurrence patterns and five report–style families via Formal Concept Analysis (RQ2), measured uneven but tangible convergence in several practices from 2018–2025 (RQ3), and translated the evidence into a report-centric feature model that specifies a defensible baseline for audit reporting (RQ4).

Three empirical takeaways stand out. First, there is a shared core: nearly every report lists findings, defines scope, describes methods at a high level, and gives remediation advice. Second, persistent gaps undermine comparability and traceability: severity criteria are often undefined, audited code versions are frequently not bound to commits or tags, threat/trust assumptions are rarely stated, and per-issue fix verification is inconsistent. Third, while executive summaries, methodology descriptions, tool disclosure, and issue-status tracking have become more common over time, risk communication (e.g., severity legends, assumptions) lags behind. Our out-of-sample check confirms the taxonomy's generality but also shows that full conformance to the baseline remains rare.

Our taxonomy and feature model open several avenues. Automating property extraction can enable large-scale monitoring of reporting quality and conformance. Linking report features to downstream outcomes (exploits, bug-bounty findings, or insured losses) would test whether particular report elements correlate with real-world risk reduction. Finally, a community-maintained, machine-readable schema for audit reports would standardize data interchange and support meta-analysis across firms and chains.

References

[1] ACPR and AMF (France). 2024. Smart Contract Certification Working Group Report. https://acpr.banque-france.fr/en. Accessed: 2025-10-23.

[2] Sagar Bhambhwani et al. 2024. Auditing Decentralized Finance. *Journal of International Money and Finance* (2024). https://www.sciencedirect.com/science/article/pii/S0890838923001270

[3] S. Bhambhwani and J. Huang. 2023. Auditing and Value in Decentralized Finance. *Working Paper* (2023). SSRN: https://ssrn.com/abstract=4564782.

[4] Thomas Bourveau et al. 2024. Decentralized Finance (DeFi) Assurance: Early Evidence. *Review of Accounting Studies* (2024). doi:10.1007/s11142-024-09834-8

[5] ChainSecurity. 2023. How To Read Smart Contract Audit Reports. https://www.chainsecurity.com/blog/how-to-read-smart-contract-audit-reports

[6] Thomas Durieux, João F. Ferreira, Rui Abreu, and Pedro Cruz. 2020. Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts. In *ICSE*. doi:10.1145/3377811.3380364

[7] Euler Labs. 2024. War & Peace: Behind the Scenes of Euler's $240M Exploit Recovery. https://www.euler.finance/blog/war-peace-behind-the-scenes-of-eulers-240m-exploit-recovery. Accessed 2025-10-22.

[8] João F. Ferreira, Pedro Cruz, Thomas Durieux, and Rui Abreu. 2020. SmartBugs: A Framework to Analyze Solidity Smart Contracts. In *ASE*. https://pedrocrvz.me/assets/ase20-smartbugs.pdf

[9] Hacken. 2025. Why Team Finance was exploited for $14.5 million, despite its audit. https://hacken.io/insights/why-team-finance-was-exploited-for-14-5-million-despite-its-audit/

[10] Halborn. 2021. Explained: The Uranium Finance Hack (April 2021). https://www.halborn.com/blog/post/explained-the-uranium-finance-hack-april-2021
[11] Halborn. 2022. Explained: The Beanstalk Hack (April 2022). https://www.halborn.com/blog/post/explained-the-beanstalk-hack-april-2022
[12] IEEE Computer Society. 2008. IEEE Standard for Software Reviews and Audits. Standard. https://ieeexplore.ieee.org/document/4601587 IEEE Std 1028-2008 specifies five types of reviews and audits and defines their procedures.
[13] International Organization for Standardization. 2013. ISO/IEC/IEEE 29119-1:2013 – Software and systems engineering – Software testing – Part 1: Concepts and definitions. Standard. https://www.iso.org/standard/45188.html Defines terminology and general concepts for software testing and describes documentation artefacts such as test plans and reports.
[14] International Organization for Standardization. 2022. ISO/IEC 15408-1:2022 – Information security, cybersecurity and privacy protection – Evaluation criteria for IT security – Part 1: Introduction and general model. Standard. https://www.iso.org/standard/77464.html Part of the Common Criteria, it specifies how to structure security evaluation and reporting so that results from different evaluators are comparable.
[15] Giuseppe Iuliano et al. 2024. Smart Contract Vulnerabilities, Tools, and Benchmarks: An Updated Systematic Literature Review. https://arxiv.org/pdf/2412.01719. Accessed 2025-10-22.
[16] Jason Kowalski. 2023. Getting a Smart Contract Audit: What You Need to Know. Blog post. https://rareskills.io/post/smart-contract-audit Explains that there is no formal definition for a smart contract audit and highlights the prevalence of superficial "rubber stamp" audits.
[17] R. Landsman, A. Kumar, and Z. Ren. 2025. Do Smart Contract Audits Matter? Evidence from 8195 Reports. *Proceedings of the IEEE Symposium on Security and Privacy* (2025).
[18] Bahador Lashkari et al. 2023. Evaluation of Smart Contract Vulnerability Analysis Tools. *Information* 14, 10 (2023), 533. https://www.mdpi.com/2078-2489/14/10/533
[19] Meson Protocol. 2022. Audits | Meson Fi. https://docs.meson.fi/references/audit-reports
[20] National Institute of Standards and Technology. 2008. NIST SP 800-115 Technical Guide to Information Security Testing and Assessment. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.
[21] Olympix. 2025. Post-Mortem: BetterBank $5M Exploit (August 2025). https://olympixai.medium.com/post-mortem-betterbank-5m-exploit-august-2025-1e8657209842
[22] OpenZeppelin. 2023. Audit Readiness Guide. https://docs.openzeppelin.com/learn/audit-readiness. Accessed: 2025-10-23.
[23] OWASP Foundation. 2021. OWASP Testing Guide v4. https://owasp.org/www-project-web-security-testing-guide/.
[24] OWASP Foundation. 2024. Smart Contract Security Verification Standard (SCSVS) - Draft. https://github.com/OWASP/SCVS. Accessed: 2025-10-23.
[25] Michael Peterson. 2023. Latest DeFi exploits show audits are no guarantee. News article. https://blockworks.co/news/audits-cannot-guarantee-defi-exploits Reports that multiple DeFi protocols suffered large hacks despite undergoing multiple audits, underscoring the limits of current audit practices.
[26] Macauley Peterson. 2023. Latest DeFi exploits show audits are no guarantee. *Blockworks* (Nov. 2023). https://blockworks.co/news/audits-cannot-guarantee-defi-exploits
[27] Quantstamp. 2022. What Is a Smart Contract Audit? https://quantstamp.com/blog/what-is-a-smart-contract-audit. Accessed: 2025-10-23.
[28] RareSkills. 2023. Getting a smart contract audit: what you need to know. https://rareskills.io/post/smart-contract-audit
[29] Rekt News. 2025. After the Post-Mortem. *Rekt News* (July 2025). https://rekt.news/after-post-mortem
[30] Rekt News. 2025. Cork Protocol – Rekt. *Rekt News* (May 2025). https://rekt.news/cork-protocol-rekt
[31] TCM Security. 2023. Penetration Test Report Template. https://tcm-sec.com/report-template.
[32] Sebastian Sinclair. 2022. DeFi Platform Exploited for $14.5M Despite Security Audits. *Blockworks* (Oct. 2022). https://blockworks.co/news/defi-platform-exploited-for-14-5m-despite-security-audits
[33] Trail of Bits. 2022. *Meson Protocol Fix Review.* Technical Report. https://static.meson.fi/MesonFi-Audit-Report-R4-2022Oct.pdf
[34] Trail of Bits. 2023. DAppSCAN Dataset. https://github.com/crytic/dappscan. Accessed: 2025-10-23.
[35] A. Yanovich, M. Efimov, and A. Frolov. 2025. Auditing the TON Blockchain: A Domain-Specific Security Study. *Journal of Blockchain Research* 7, 1 (2025), 1–20.