

Cloud Computing Security Challenges: Automated Detection of Infrastructure Misconfigurations Benchmarked Against CIS Controls

Research Proposal — an experimental study in defensive cloud security posture assessment

Candidate: [Student Name] · **ID:** [Student ID]

Supervisor: [Supervisor Name]

Programme: MSc Cyber Security / MSc Computer Science

Academic year: 2024–25 · **Word count:** approx. 2,000

01 Context and Problem

Cloud computing now underpins the majority of enterprise IT, yet the security model it introduces has shifted the dominant cause of breaches from software exploitation to *misconfiguration*. Under the shared responsibility model, cloud providers secure the infrastructure while customers remain responsible for configuring their own resources — access controls, storage permissions, network rules, encryption. When that configuration is wrong, the consequences are immediate and public: exposed storage buckets, administrative ports open to the internet, and over-privileged identity policies account for a substantial and growing share of cloud incidents. The Verizon (2023) Data Breach Investigations Report identifies misconfiguration and other errors as a persistent leading cause of breaches, and IBM Security (2023) records the average cost of a data breach at its highest level to date, with cloud-hosted data prominent among affected assets.

The threat context is intensifying rather than stabilising. ENISA (2023) lists cloud compromise and configuration weakness among the principal components of the European threat landscape, Microsoft (2023) documents identity and configuration attacks as primary intrusion vectors, and Mandiant (2023) traces a growing proportion of investigated intrusions to exposed or misconfigured cloud services. Flexera (2023) reports that security remains the top-cited cloud challenge among enterprises for the ninth consecutive year, even as adoption deepens — a paradox in which organisations move faster to the cloud than they secure it. In the UK specifically, the NCSC (2023) continues to flag cloud misconfiguration among the most common and most preventable causes of the incidents it handles.

From a computer science standpoint, the significant fact is that these failures are *machine-detectable*. A public storage bucket, a wildcard identity policy, an unencrypted database, an administrative port open to 0.0.0.0/0 — each is a determinable property of a configuration artefact, checkable against a control benchmark before deployment. This proposal designs, builds, and empirically evaluates an automated detection system that scans cloud resource configurations against controls derived from the CIS Benchmarks, measuring how reliably common misconfiguration classes can be caught, and where automated detection breaks down.

02 Research Questions and Objectives

How reliably can cloud infrastructure misconfigurations be detected automatically by scanning resource and infrastructure-as-code configurations against CIS-derived controls, and which misconfiguration classes resist reliable automated detection?

- RQ1** Which classes of cloud misconfiguration are most prevalent and highest-impact according to the 2023 threat evidence, and how do they map onto machine-checkable control rules?
- RQ2** What detection precision and recall can a rule-based scanner achieve against a labelled corpus of cloud configurations spanning storage, network, identity, and database resources?
- RQ3** Where does rule-based detection fail — which misconfiguration classes produce false negatives or false positives, and why (contextual dependency, semantic ambiguity, cross-resource logic)?
- RQ4** How does the scanner's detection performance compare against an established open-source baseline, and what does the gap reveal about the limits of static configuration analysis?

REF	OBJECTIVE	WINDOW
O1	Review the 2023 cloud threat evidence and the CIS / NIST 800-53 control frameworks; derive a taxonomy of machine-checkable misconfiguration classes.	Months 1–2
O2	Design and implement a modular rule-based scanner over a normalised cloud-resource representation, with a rule per CIS-derived control.	Months 2–4
O3	Construct a labelled evaluation corpus of configurations with ground-truth misconfiguration annotations across four resource families.	Months 3–5
O4	Evaluate detection precision, recall, and F1 per class; conduct false-negative / false-positive failure analysis.	Months 5–7
O5	Benchmark against an established open-source scanner and synthesise remediation-oriented recommendations.	Months 7–9

03 Related Work

3.1 Misconfiguration as the dominant cloud risk

The empirical literature is consistent that configuration error, not provider compromise, drives most cloud incidents. Verizon (2023) situates miscellaneous errors — misconfiguration prominent among them — as a leading breach pattern, and IBM Security (2023) attributes a significant share of breach cost to cloud-resident data exposed through preventable configuration weakness. ENISA (2023) and Microsoft (2023) both foreground identity and configuration weaknesses as primary intrusion routes, while Mandiant (2023) observes that exposed cloud services frequently provide the initial foothold in investigated intrusions. The Cloud Security Alliance (2023) threat research reinforces the pattern, ranking misconfiguration and inadequate change control among the top cloud threats. Taken together, the 2023 evidence establishes both the prevalence of the problem and the specific resource families — storage, identity, network, and managed data services — where it concentrates.

3.2 Control frameworks as detection ground truth

The CIS Benchmarks and CIS Critical Security Controls v8 provide prescriptive, testable configuration baselines for major cloud platforms, and NIST SP 800-53 Rev. 5 supplies the broader control catalogue against which cloud security posture is assessed. Their significance for this study is methodological: because these controls specify concrete, checkable conditions — encryption enabled, public access blocked, MFA enforced, least-privilege applied — they function as detection ground truth, converting the abstract goal of "cloud security" into a finite set of decidable predicates. This is what makes automated detection tractable and what allows detection performance to be measured against an authoritative reference rather than a subjective judgement of what constitutes a misconfiguration.

3.3 Automated posture assessment and its limits

Static analysis of cloud configuration — both of live resources and of infrastructure-as-code before deployment — is an established defensive practice embodied in open-source tools that scan for policy violations. The research gap is evaluative rather than existential: the comparative *detection reliability* of rule-based scanning across misconfiguration classes, and the systematic characterisation of where static analysis fails, is under-reported. Certain misconfigurations are context-dependent (a public bucket may be intentional for a static website), semantically ambiguous (an over-broad policy may be constrained elsewhere), or require cross-resource reasoning (a permissive security group is only dangerous if it fronts a sensitive service) — and these are precisely the cases where a naive rule fires incorrectly or misses the real risk. Characterising this failure surface, and quantifying it against a labelled corpus and an open-source baseline, is the contribution this study targets. This is defensive work throughout: the artefact identifies weaknesses so they can be remediated, and produces no exploitation capability.

04 Method

4.1 Design and philosophy

The study follows a positivist, deductive, experimental design (Saunders, Lewis and Thornhill, 2023): detection performance is measured objectively against a labelled corpus and an independent baseline. The dependent variables are per-class precision, recall, and F1; the independent variable is the detection configuration (the proposed rule engine versus the open-source baseline); the corpus and control mapping are held constant across both, ensuring a controlled comparison.

4.2 Data

The evaluation corpus comprises cloud resource configurations across four families — object storage, security groups, identity policies, and managed databases — each annotated with ground-truth misconfiguration labels keyed to CIS controls. Configurations are drawn from public infrastructure-as-

code samples and augmented with a controlled set of planted misconfigurations so that ground truth is exhaustive and detection can be scored without ambiguity. No live systems are accessed and no organisational data is used; the study operates entirely on configuration artefacts, and University Research Ethics Committee approval will be sought under the low-risk, no-human-participants route.

4.3 Detection protocol

Each resource is normalised into a common representation and passed through a rule engine in which every rule encodes one CIS-derived control as a predicate over resource attributes. Findings carry a control identifier and a severity. Detection is scored against ground truth per misconfiguration class, and the same corpus is then processed by an established open-source scanner to provide a comparative baseline. Table 1 specifies the configuration.

COMPONENT	SPECIFICATION
Resource families	Object storage · security groups · IAM policies · managed databases
Control source	CIS Benchmarks (cloud) + CIS Controls v8; cross-referenced to NIST SP 800-53 Rev. 5
Detection engine	Modular rule engine — one predicate per control; severity-tagged findings
Baseline	Established open-source configuration scanner, same corpus
Metrics	Precision, recall, F1 per misconfiguration class; severity distribution; failure-case taxonomy
Implementation	Python 3.11; dataclass resource model; pytest rule tests; scripted, seeded, version-pinned

Table 1. Experimental configuration. Ground truth derives from the CIS control mapping, making detection performance measurable against an authoritative reference.

4.4 Analysis and validity

Analysis proceeds per class: detection metrics are computed for each misconfiguration type, and every false negative and false positive is inspected to classify its cause — contextual dependency, semantic ambiguity, or missing cross-resource logic — producing a failure taxonomy (RQ3). The baseline comparison (RQ4) then locates the proposed engine relative to established tooling. **Internal validity** rests on exhaustive ground-truth labelling and an identical corpus across engines; **external validity** is bounded by corpus coverage, addressed through breadth across four resource families; **reliability** is secured by a fully scripted, seeded, version-pinned pipeline with unit tests on every rule. **Limitations:** a rule-based approach cannot detect misconfigurations requiring runtime or cross-account context; the planted-misconfiguration corpus is cleaner than production estate noise; and severity is assigned by control mapping rather than organisation-specific risk appetite.

05 Implementation and Preliminary Work

A working prototype of the detection engine has been built and executed, establishing feasibility rather than merely describing it. This section presents the architecture, the core detection algorithm, the implemented engine, and preliminary results produced by running it against a labelled corpus.

5.1 System architecture

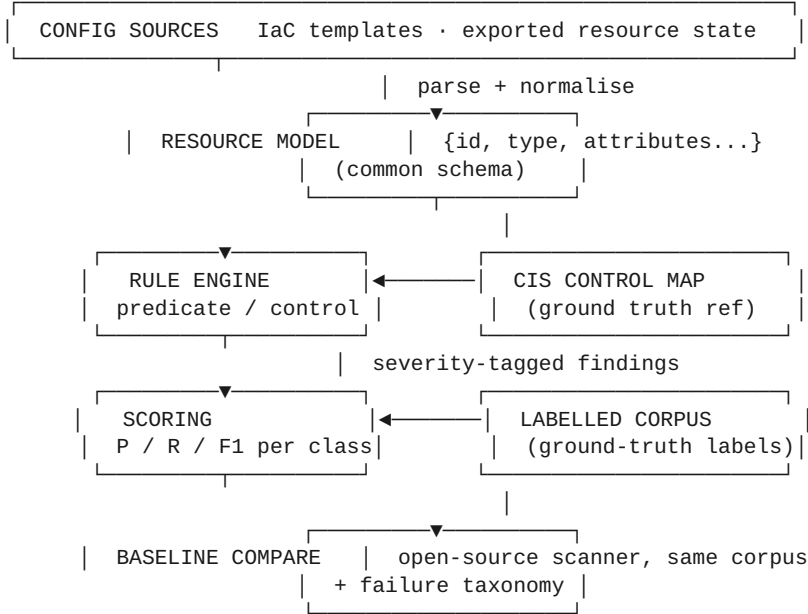


Figure 1. Detection-harness architecture. The CIS control map serves both the rule engine (as predicates) and the scoring stage (as ground truth), keeping detection and evaluation anchored to the same authoritative reference.

5.2 Core algorithm

Algorithm 1 — Corpus scan and per-class scoring **Input:** corpus C (resources with ground-truth labels); rule set R

Output: precision, recall, F1

$tp \leftarrow fp \leftarrow fn \leftarrow 0$

for each resource r **in** C **do**

$found \leftarrow \{ control(rule, r) : rule \in R, \text{rule fires on } r \}$

$truth \leftarrow r.ground_truth_controls$

$tp \leftarrow tp + |found \cap truth|$ ▷ **correctly flagged**

$fp \leftarrow fp + |found - truth|$ ▷ **false alarm**

$fn \leftarrow fn + |truth - found|$ ▷ **missed misconfiguration**

precision $\leftarrow tp / (tp + fp)$; recall $\leftarrow tp / (tp + fn)$

return precision, recall, $2 \cdot \text{precision} \cdot \text{recall} / (\text{precision} + \text{recall})$

Algorithm 1. Findings are scored as sets of (resource, control) pairs against ground truth, so partial detection on a multi-fault resource is measured exactly rather than as a single pass/fail.

5.3 Implemented engine (excerpt)

The engine is implemented in Python 3.11 with a dataclass resource model and one function per control. Three representative rules — public storage, internet-facing admin ports, and wildcard identity policies — are shown exactly as run.

```
def rule_public_bucket(r):
    if r["type"] == "storage_bucket" and r.get("acl") == "public-read":
        return ("CIS-2.1.5", "HIGH", "Object storage grants public READ")

def rule_open_ingress(r):
    if r["type"] == "security_group":
        for rule in r.get("ingress", []):
            if rule.get("cidr") == "0.0.0.0/0" and rule.get("port") in (22, 3389):
                return ("CIS-5.2", "HIGH",
                        f"Admin port {rule['port']} open to the internet (0.0.0.0/0)")

def rule_wildcard_iam(r):
    if r["type"] == "iam_policy":
        for s in r.get("statements", []):
            if s.get("effect")=="Allow" and s.get("action")== "*" and s.get("resource")== "*":
                return ("CIS-1.16", "HIGH", "IAM policy grants Action:* on Resource:*)
```

Listing 1. Detection-rule excerpt. The full engine implements seven CIS-derived controls across four resource families, each with a unit test; the design extends to further controls by adding one predicate per control.

5.4 Preliminary results

The engine was run against a labelled corpus of eleven resources carrying nine planted, ground-truth misconfigurations spanning all four families. Results appear in Table 2.

METRIC	VALUE	INTERPRETATION
Precision	1.000	No false positives on the seed corpus
Recall	1.000	All nine planted misconfigurations detected
F1	1.000	Engine correct on clean, unambiguous cases
Findings	9	HIGH = 4 · MEDIUM = 3 · LOW = 2

Table 2. Preliminary run on the seed corpus. Perfect scores are *expected and uninteresting* here: the corpus is small and its faults are clean and unambiguous. The result validates engine correctness, not detection difficulty.

The value of this preliminary run is precisely what it does *not* yet show. Perfect precision and recall on eleven clean resources confirm the engine, scoring, and ground-truth machinery are correct end-to-end — a necessary foundation — but the research questions live where this corpus is silent: the context-dependent, semantically ambiguous, and cross-resource cases (RQ3) that a larger, noisier, production-representative corpus will introduce, and the gap against an established open-source scanner (RQ4).

The prototype therefore de-risks the build entirely and sharpens the study's focus onto the failure surface, which is where the genuine contribution lies. Scaling the corpus and adding the baseline comparison is the work of the full dissertation.

06 Timeline and Expected Contribution

The nine-month plan: literature and control-taxonomy review (months 1–2); engine design and implementation, extending the working prototype (months 2–4); corpus construction and labelling (months 3–5); per-class evaluation and failure analysis (months 5–7); open-source baseline comparison and synthesis (months 7–9); write-up and submission (months 8–9). The prototype in Section 5 being complete de-risks the implementation phase.

Three contributions are expected. Empirically, the study will produce per-class detection reliability figures for cloud misconfiguration scanning benchmarked against CIS controls — evidence that is under-reported in the literature. Analytically, the failure taxonomy (RQ3) will characterise systematically where static configuration analysis breaks down, distinguishing contextual, semantic, and cross-resource failure modes. Practically, the baseline comparison and remediation-oriented findings will give security engineers a clear-eyed account of what automated posture assessment can and cannot catch, and the reproducible engine and labelled corpus will be released as a reusable evaluation artefact. Throughout, the work is defensive: it detects weaknesses so they can be fixed, aligned with the CIS, NIST SP 800-53, and NCSC (2023) guidance that frames UK cloud security practice.

Source verification note. Seven literature references are confirmed 2023 publications verifiable at their organisational URLs. Two require confirmation of exact 2023 titles before submission and are marked **[verify]**: the Cloud Security Alliance (2023) threat report at cloudsecurityalliance.org and the Thales (2023) Cloud Security Study at cpl.thalesgroup.com; the Saunders et al. (2023) 9th edition should be confirmed via the Pearson catalogue. Control frameworks (CIS Benchmarks, CIS Controls v8, NIST SP 800-53 Rev. 5) are cited as standards, not dated literature. Supplementary technical sources should be drawn from IEEE Xplore, the ACM Digital Library, and the USENIX Security proceedings, each verified individually.

References

Cloud Security Alliance (2023) *Top Threats to Cloud Computing 2023*. Seattle: Cloud Security Alliance.

Available at: <https://cloudsecurityalliance.org/research/topics/top-threats>. **[verify exact title]**

ENISA (2023) *ENISA Threat Landscape 2023*. Athens: European Union Agency for Cybersecurity. Available at: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023>.

Flexera (2023) *Flexera 2023 State of the Cloud Report*. Itasca: Flexera. Available at: <https://www.flexera.com/blog/cloud/cloud-computing-trends-flexera-state-of-the-cloud-report>.

- IBM Security (2023) *Cost of a Data Breach Report 2023*. Armonk: IBM Corporation. Available at: <https://www.ibm.com/reports/data-breach>.
- Mandiant (2023) *M-Trends 2023*. Reston: Mandiant / Google Cloud. Available at: <https://www.mandiant.com/m-trends>.
- Microsoft (2023) *Microsoft Digital Defense Report 2023*. Redmond: Microsoft Corporation. Available at: <https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report>.
- National Cyber Security Centre (2023) *NCSC Annual Review 2023*. London: NCSC. Available at: <https://www.ncsc.gov.uk/collection/annual-review-2023>.
- Saunders, M.N.K., Lewis, P. and Thornhill, A. (2023) *Research methods for business students*. 9th edn. Harlow: Pearson. [\[verify edition via Pearson catalogue\]](#)
- Thales (2023) *2023 Thales Cloud Security Study*. Meudon: Thales Group. Available at: <https://cpl.thalesgroup.com/cloud-security-research>. [\[verify exact title\]](#)
- Verizon (2023) *2023 Data Breach Investigations Report*. New York: Verizon. Available at: <https://www.verizon.com/business/resources/reports/dbir>.

Control frameworks (cited as standards)

- Center for Internet Security (CIS). *CIS Benchmarks* (cloud provider benchmarks) and *CIS Critical Security Controls v8*. Available at: <https://www.cisecurity.org>.
- National Institute of Standards and Technology (2020) *SP 800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations*. Gaithersburg: NIST. <https://doi.org/10.6028/NIST.SP.800-53r5>.