

A Multi-Criteria Framework for Performance, Security, And Cost Optimization in Cloud Computing Applications

Dr. Praveen Kumar Shrivastava

Department of I.T. KIT, Raigarh, Chhattisgarh, India

Abstract

Cloud computing has become a major computing paradigm for delivering scalable infrastructure, application platforms, and software services. However, selecting and managing cloud resources remains challenging because application performance, security requirements, and operational expenditure frequently compete with one another. Configurations that provide low response times may increase resource costs, while aggressive cost reduction may introduce performance degradation or unacceptable security risks. This paper proposes a Multi-Criteria Cloud Application Optimization Framework (MCCAF) for evaluating cloud application configurations through integrated performance, security, and cost assessment. The framework combines workload characterization, security-policy enforcement, resource configuration, multi-objective optimization, and sensitivity analysis. Its mathematical formulation minimizes application response time and total operating cost while maximizing verified security-control coverage, subject to application-specific service-level requirements. A Pareto-based decision process is proposed to identify non-dominated configurations before selecting an appropriate solution according to stakeholder priorities. The framework also specifies an experimental methodology using CloudSim or a comparable test environment, benchmark scheduling methods, repeated workload trials, and statistical evaluation. Unlike a single-score optimization approach, the framework treats mandatory security requirements as constraints and evaluates the trade-offs between competing objectives. The proposed methodology establishes a foundation for reproducible cloud application optimization, although its effectiveness and superiority require empirical validation. The study contributes an application-oriented decision framework for researchers and cloud administrators seeking balanced performance, security, and economic efficiency.

Keywords: Cloud computing; multi-criteria optimization; application performance; cloud security; cost optimization; resource allocation; Pareto optimization; service-level agreement; cloud resource management.

1. Introduction

Cloud computing enables organizations to access computing resources, storage, networks, databases, and software services through configurable, on-demand infrastructure. Its service models include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), while deployment models include public, private, community, and hybrid clouds [1].

Cloud adoption supports applications in healthcare, education, finance, e-commerce, manufacturing, scientific computing, and government services. These applications differ in their computational demands, data sensitivity, response-time requirements, and financial constraints. Consequently, cloud resource management requires more than simply selecting the least expensive

virtual machine or the configuration with the highest processing capacity.

Three objectives are particularly important.

1. **Performance optimization:** Applications must respond within acceptable time limits, process workloads efficiently, and maintain throughput under changing demand.
2. **Security optimization:** Cloud configurations must enforce appropriate identity management, access control, encryption, monitoring, and data-protection policies.
3. **Cost optimization:** Resource allocation should avoid unnecessary expenditure while maintaining the required service quality and security posture.

These objectives are interdependent. Increasing compute capacity may reduce execution time but increase expenditure. Stronger encryption or additional security inspection may introduce processing overhead, although the impact depends on the implementation. Similarly, reducing the number of active instances can save money but may increase latency or reduce resilience.

The problem is therefore a constrained multi-objective optimization problem rather than a simple resource-selection task.

1.2 Problem statement

Existing cloud optimization approaches frequently emphasize selected objectives such as task completion time, resource utilization, energy consumption, or infrastructure cost. Research has also investigated multi-criteria cloud infrastructure selection using dependability and cost models [2]. Other studies have formulated hybrid-cloud scheduling as a multi-objective problem involving security risk, resource utilization, cost, and completion time [3].

Recent research continues to explore multi-objective resource scheduling and evolutionary optimization in IaaS environments [4]. This demonstrates the relevance of the problem but also means that a new framework cannot claim novelty merely by combining performance, security, and cost.

The research problem addressed here is how to develop an application-oriented decision framework that jointly evaluates these objectives, preserves mandatory security requirements, and produces transparent trade-offs among feasible cloud configurations.

1.3 Research objectives

The objectives are:

- To formulate a unified model for evaluating cloud application performance, security, and cost.
- To identify measurable indicators for each optimization dimension.
- To design a constrained multi-objective optimization process for selecting cloud configurations.
- To develop an evaluation methodology that supports reproducible comparison with baseline methods.

- To examine the sensitivity of configuration rankings to changing workloads, security requirements, and stakeholder preferences.

1.4 Research questions

RQ1: How can application performance, security, and cost be represented in a common cloud optimization model?

RQ2: How can mandatory security requirements be maintained while optimizing performance and expenditure?

RQ3: How can Pareto-based optimization support cloud configuration selection when objectives conflict?

RQ4: How sensitive are configuration rankings to workload intensity, security requirements, and objective weights?

RQ5: Under what conditions does the proposed framework improve the balance between performance, security, and cost compared with conventional selection strategies?

1.5 Research contributions

The intended contributions of this study are:

1. A constrained mathematical formulation for cloud application optimization.
2. A modular framework connecting workload characterization, security validation, configuration evaluation, and multi-objective selection.
3. A Pareto-based decision procedure that separates mandatory security constraints from negotiable performance and economic objectives.
4. A reproducible experimental protocol for testing the framework under different workload conditions.
5. A sensitivity-analysis procedure for examining whether the selected configuration remains suitable when requirements change.

The framework's originality must be confirmed through a systematic comparison with closely related studies before submission.

2. Related Work

2.1 Cloud computing architecture and service measurement

The NIST definition establishes the principal characteristics and service models of cloud computing [1]. These characteristics provide the architectural foundation for evaluating resource provisioning, elasticity, and measured service consumption.

Cloud service selection also requires consistent metric definitions. NIST Special Publication 500-307 discusses the description of measurable cloud service properties, including quality of service, availability, and reliability [5]. Such metrics are important because comparisons are meaningful only when measurement units, conditions, and service boundaries are consistent.

2.2 Cloud infrastructure selection using multiple criteria

Araujo et al. investigated decision-making in cloud environments through multiple-criteria analysis and stochastic models, with attention to dependability and cost [2]. Their work demonstrates the value of combining service-reliability considerations with economic assessment rather than treating infrastructure selection as a single-objective problem.

The proposed MCCAFA framework builds on this general principle but places explicit emphasis on application performance, security-policy compliance, and operating expenditure. Its intended distinction is the separation of mandatory security constraints from objectives that can be optimized according to stakeholder preferences.

2.3 Multi-objective scheduling in hybrid clouds

Research on many-objective hybrid-cloud scheduling has considered factors including task completion time, cost, resource utilization, and risk [3]. These approaches are relevant because hybrid environments introduce additional decisions concerning workload placement, privacy, and resource selection.

However, the proposed study focuses on application configuration assessment rather than only task scheduling. Its evaluation should therefore consider both application-level service outcomes and the security requirements imposed on the selected deployment.

2.4 Recent optimization approaches

A 2025 study in *Swarm and Evolutionary Computation* proposed a hybrid multi-objective resource-scheduling approach for IaaS and evaluated it using CloudSim [4]. Such work illustrates the continuing development of optimization algorithms for cloud scheduling.

NIST has also described automated cloud-security configuration and continuous monitoring as mechanisms for reducing manual configuration errors and checking the presence of required controls [6].

Together, these research directions motivate a framework that connects resource optimization with explicit security validation. The proposed work should be benchmarked against these and other closely related approaches rather than claiming that earlier research has ignored multi-objective optimization.

2.5 Research gap and novelty assessment

The specific research gap proposed for investigation is the need for a transparent decision procedure that simultaneously:

- enforces non-negotiable security requirements;
- optimizes measurable application-level performance and cost;
- exposes non-dominated configuration alternatives;
- supports stakeholder-specific selection among those alternatives; and
- evaluates ranking stability under uncertainty.

These features form a testable research proposition. They are not, by themselves, proof that the framework is the first of its kind. A completed Scopus and Web of Science review is necessary to determine whether comparable frameworks already provide the same combination of capabilities.

3. Proposed Methodology

3.1 Research design

The study follows a design-and-evaluation methodology. It first specifies the mathematical model, then implements the decision procedure, and finally evaluates its behavior against baseline configuration-selection strategies.

The research comprises six phases:

1. Workload and application characterization.
2. Definition of performance, security, and cost metrics.
3. Generation of feasible cloud configurations.
4. Evaluation and security validation.
5. Multi-objective optimization and decision selection.
6. Experimental comparison and sensitivity analysis.

The initial study may use a simulated IaaS environment. Subsequent validation should use real cloud measurements to examine whether simulation assumptions adequately represent production behavior.

3.2 System model

Let a cloud application have a workload (W) and a set of candidate configurations:

$$X = \{x_1, x_2, \dots, x_n\}$$

Each configuration (x_i) defines the relevant resource and policy settings, including:

- virtual machine or container specifications;
- number of application instances;
- resource-allocation policy;
- storage and networking configuration;
- scaling parameters;
- security controls; and
- pricing assumptions.

The system observes a workload characterized by request volume, concurrency, request complexity, data sensitivity, and latency requirements.

For each candidate configuration, the evaluation module estimates or measures response time, throughput, operating cost, and security compliance. The resulting measurements are passed to the optimization module.

3.3 Performance model

Let $(T(x, W))$ denote the response time associated with configuration (x) under workload (W).

Response time should be measured using both average and percentile statistics. The mean indicates overall behavior, while the 95th and 99th percentiles help identify delays affecting a smaller proportion of requests.

Let $(Q(x, W))$ denote throughput, expressed as completed requests or jobs per unit time.

A performance objective can be formulated as:

$$\begin{aligned} & [\\ & \min_{x \in X} f_P(x) = \\ & \alpha T_{\{95\}}(x, W) + \\ & \beta T_{\{99\}}(x, W) \\ &] \end{aligned}$$

where:

- $(T_{\{95\}})$ is the 95th-percentile response time;
- $(T_{\{99\}})$ is the 99th-percentile response time; and
- (α) and (β) are non-negative weights.

The weights must be selected according to the application's service objectives. Alternatively, response time and throughput may be maintained as separate objectives to avoid concealing trade-offs through a weighted sum.

3.4 Security model

Security assessment must distinguish between the existence of a control and evidence that the control operates effectively.

Let $(c_j(x))$ represent whether mandatory security control (j) has been verified for configuration (x). Let (r_j) represent its requirement level.

The configuration is security-feasible only when:

$$\begin{aligned} & [\\ & c_j(x) \geq r_j, \quad j=1, \dots, k \\ &] \end{aligned}$$

For binary controls, $(c_j(x))$ may take values of zero or one. More detailed controls may require graded assessments supported by measurable evidence.

Examples of security requirements include:

- identity and access management;
- encryption in transit and at rest;
- least-privilege permissions;

- secure configuration and vulnerability management;
- audit logging and monitoring;
- backup and recovery;
- data residency and retention requirements, where applicable.

A security score can summarize additional desirable controls, but it must not compensate for failure to satisfy a mandatory requirement. A configuration that fails a mandatory security constraint is excluded from the feasible solution set.

The security assessment should use an explicitly documented control catalogue, threat model, verification procedure, and evidence source. A high control-coverage score must not be interpreted as proof that a system is immune to attack.

3.5 Cost model

Let $(C(x, W))$ denote the total operating cost of executing workload (W) under configuration (x) .

A practical cost model may include:

$$[\begin{aligned} C = & C_{\text{compute}} + \\ & C_{\text{storage}} + \\ & C_{\text{network}} + \\ & C_{\text{services}} + \\ & C_{\text{security}} + \\ & C_{\text{operations}} \end{aligned}]$$

The terms represent compute consumption, storage, data transfer, managed services, security-related services, and operational expenses, respectively.

For a simulation, the model may use the simulated resource prices. For a real cloud deployment, prices should be taken from the relevant provider's pricing information and actual usage records.

Where configurations complete different amounts of useful work, cost per successfully completed request or job should also be reported:

$$[\begin{aligned} C_{\text{unit}} = & \frac{C(x, W)}{N_{\text{successful}}} \end{aligned}]$$

where $(N_{\text{successful}})$ is the number of successfully completed requests or jobs. This measure is undefined when no work completes successfully, in which case the configuration must be treated as infeasible or failed.

3.6 Multi-objective optimization model

The framework minimizes performance delay and operating cost while enforcing security and service requirements:

$$[\begin{aligned} \min_{x \in X} & \\ F(x) = & \left[\begin{aligned} & f_P(x), \\ & C(x, W) \end{aligned} \right] \end{aligned}]$$

subject to:

$$[\begin{aligned} \begin{aligned} & c_j(x) \geq r_j \quad \forall j \\ & T_{95}(x, W) \leq L_{95} \\ & Q(x, W) \geq Q_{\min} \\ & C(x, W) \leq B \end{aligned} \end{aligned}]$$

Here, (L_{95}) represents the maximum acceptable 95th-percentile response time, $(Q_{\min})$ is the minimum throughput requirement, and (B) is the available budget when a budget limit applies.

This formulation makes security a feasibility requirement rather than a freely exchangeable objective. Where security risk can be quantified reliably, a separate risk objective may be added, provided that the risk model and evidence are explicitly defined.

3.7 Pareto-based selection

A configuration is Pareto-dominated if another feasible configuration is at least as good across every objective and strictly better on at least one.

The Pareto-optimal set contains feasible configurations for which no such dominating alternative exists. It exposes the trade-offs between faster response times and lower costs without assuming that one stakeholder preference is universally correct.

After generating the Pareto set, a decision-maker can select a configuration using an explicitly documented policy, such as:

- minimum cost subject to response-time and security constraints;
- minimum response time subject to a budget limit; or
- a normalized weighted score for performance and cost among security-feasible configurations.

The selected configuration and its alternatives should be reported together so that decision-makers can inspect the consequences of their preferences.

4. Proposed MCCAFA Architecture

The framework contains five logical modules.

Module 1: Workload monitor. Collects workload characteristics, including request rate, concurrency, application demand, and changes in load over time.

Module 2: Configuration generator. Generates candidate configurations from permitted instance types, scaling settings, network options, storage configurations, and security policies.

Module 3: Performance and cost evaluator. Measures or estimates response time, throughput, resource consumption, and expenditure under a consistent workload.

Module 4: Security validation engine. Verifies mandatory security controls and excludes configurations that fail the required policy.

Module 5: Multi-objective decision engine. Identifies Pareto-optimal candidates and selects a configuration according to explicit service and budget priorities.

The architecture should retain an audit record of input workloads, configuration parameters, metric definitions, policy checks, and the final decision. This enables the same experiment to be repeated and the reasons for a configuration choice to be reviewed.

4.1 Proposed algorithm

Algorithm 1: Multi-Criteria Cloud Application Optimization

Input: Workload (W), candidate configurations (X), security policy (S), latency limit (L), throughput requirement ($Q_{\min}$), and budget (B).

Output: A selected feasible configuration and the Pareto-optimal alternatives.

1. Initialize the candidate configuration set (X).
2. Characterize the application workload and service requirements.
3. For each configuration ($x \in X$), configure the required resources and policies.
4. Validate all mandatory security controls.
5. If any mandatory control fails, mark (x) as infeasible and continue to the next configuration.
6. Execute or simulate the workload under the specified conditions.
7. Measure response-time percentiles, throughput, resource consumption, and operating cost.
8. Check latency, throughput, and budget constraints.
9. Mark configurations violating mandatory constraints as infeasible.
10. Store the objective values and experimental metadata for each feasible configuration.
11. Identify the Pareto-optimal set from the feasible candidates.
12. Apply the documented stakeholder decision policy to select one configuration.
13. Record the selected configuration, trade-offs, and alternative Pareto-optimal solutions.
14. Return the decision and evaluation record.

The procedure assumes that candidate configurations can be evaluated using either simulation or controlled experiments. In an operational deployment, configuration changes must also account for migration overhead, warm-up periods, and the possibility of service interruption.

4.2 Computational complexity

If (n) candidate configurations are evaluated and each experiment requires time (E), the evaluation stage requires approximately ($O(nE)$) total experimental work, excluding parallel execution.

A straightforward pairwise Pareto-dominance comparison over (n) candidates and (m) objectives

requires ($O(mn^2)$) time. More efficient non-dominated sorting methods can reduce overhead in appropriate settings, but their actual performance depends on the algorithm and problem size.

The principal practical cost may be configuration evaluation rather than Pareto sorting. Therefore, future implementations should investigate candidate pruning, surrogate models, or adaptive experiment selection, while ensuring that these methods do not discard potentially superior configurations.

5. Experimental Design and Evaluation

5.1 Experimental environment

A reproducible simulation can be developed using CloudSim or a comparable cloud modeling environment. CloudSim is appropriate for studying resource allocation, virtual machine placement, and scheduling under controlled conditions. Security-policy evaluation must be modeled separately unless the chosen simulator explicitly represents the relevant controls and their effects.

The experimental setup should document:

Parameter	Proposed specification
Simulation platform	CloudSim or a documented equivalent
Infrastructure	Heterogeneous virtual machines and hosts
Workloads	Low, medium, and high request loads
Configuration variables	Instance type, instance count, scaling policy, security controls
Performance metrics	Mean, p95, and p99 response time; throughput
Security metrics	Verified mandatory-control compliance and documented residual-risk indicators
Cost metrics	Total operating cost and cost per successful request
Repetitions	Multiple independent runs per experimental condition
Statistical reporting	Means, dispersion, confidence intervals where appropriate

The actual software version, host specifications, workload sizes, pricing assumptions, and number of repetitions must be recorded once the experiment is implemented.

5.2 Baseline approaches

The proposed framework should be compared with at least three decision strategies.

1. **Lowest-cost selection:** Chooses the least expensive candidate that satisfies the explicitly enforced constraints.
2. **Performance-first selection:** Chooses the fastest candidate that satisfies mandatory security and budget requirements.
3. **Weighted-sum selection:** Selects a configuration using a documented normalized combination of performance and cost, subject to the same security requirements.

A further comparison with an established multi-objective optimizer, such as NSGA-II, would help determine whether MCCAF offers an advantage beyond the use of Pareto selection itself. The baseline methods must operate on the same candidate space, workload, constraints, and measurement procedure.

5.3 Workload scenarios

Experiments should examine different request volumes, levels of concurrency, application resource demands, and security-policy configurations.

For each scenario, the framework should record:

- average, p95, and p99 response times;
- throughput and failed-request rate;
- compute, memory, and network utilization;
- total operating cost and cost per successful request;
- mandatory security-control compliance;
- feasibility rate and number of Pareto-optimal configurations; and
- optimization and evaluation overhead.

The experiments should include both stable workloads and changing workloads. A framework that performs well under a constant load may not remain effective during sudden traffic spikes or changes in application demand.

5.4 Performance calculations

The relative change in response time between a baseline and the proposed framework can be calculated as:

$$\left[\begin{array}{l} \Delta T(\%) = \\ \frac{T_{\text{baseline}} - T_{\text{MCCAF}}}{T_{\text{baseline}}} \times 100 \end{array} \right]$$

Similarly, the relative change in operating cost is:

$$\left[\begin{array}{l} \Delta C(\%) = \\ \frac{C_{\text{baseline}} - C_{\text{MCCAF}}}{C_{\text{baseline}}} \times 100 \end{array} \right]$$

Positive values indicate a reduction in the relevant metric. Negative values indicate an increase.

These calculations should be performed using measured results under equivalent conditions. If the baseline value is zero, the percentage formula is undefined and an alternative comparison must be used.

Security performance should be reported independently through control compliance and appropriately validated risk measurements. A reduction in cost or response time must never be presented as an overall improvement if mandatory security requirements are violated.

5.5 Statistical validation

Each experimental condition should be repeated using independent runs and, where applicable, different random seeds. The study should report sample sizes, central tendencies, variability, and confidence intervals.

Suitable statistical tests should be selected according to the distribution, dependence structure, and experimental design. Effect sizes should accompany significance tests where appropriate. Multiple comparisons should be addressed when numerous configurations or workload scenarios are evaluated.

Sensitivity analysis should vary the budget, response-time limits, workload intensity, and decision weights. If small changes in these inputs produce substantial changes in configuration rankings, the paper should report that instability rather than present one configuration as universally optimal.

6. Results and Discussion: Reporting Framework

This section specifies how experimental findings should be presented after execution. No measured performance improvement is claimed in the present manuscript.

6.1 Performance analysis

The performance analysis should compare response-time distributions and throughput across the baseline methods and MCCAF. The analysis should identify whether any reduction in response time is achieved consistently across workload levels or only under selected conditions.

If the framework introduces an optimization delay, that delay should be reported separately from application response time. This distinction is important because an expensive optimization process may reduce steady-state cost but be unsuitable for highly dynamic applications.

6.2 Security analysis

The security analysis should report the number and proportion of configurations that satisfy all mandatory controls. It should identify which controls eliminate candidate configurations and whether the remaining candidates meet the application's security requirements.

The analysis should not equate control coverage with complete protection. Residual risks, limitations in the verification method, and unmodeled threats must be discussed explicitly.

6.3 Cost analysis

Cost analysis should compare total expenditure and cost per successful request or job. It should also examine whether the least-cost configuration remains feasible under peak loads and strict response-time limits.

The results should distinguish infrastructure charges from security-service charges, data-transfer expenses, and optimization overhead wherever those costs are material.

6.4 Trade-off analysis

The Pareto-front analysis should show which configurations offer the most favorable combinations of response time and cost while satisfying security constraints.

The discussion should identify whether the framework provides a meaningful set of alternatives or whether a small number of configurations dominate most of the candidate space. The stability of these results under

changing workloads and decision preferences should also be reported.

6.5 Results table template

The following table should be populated with actual experimental measurements.

Method	Mean response time	p95 response time	Cost per successful request	Security-feasible
Lowest-cost selection	To measure	To measure	To measure	To verify
Performance-first selection	To measure	To measure	To measure	To verify
Weighted-sum selection	To measure	To measure	To measure	To verify
MCCAF	To measure	To measure	To measure	To verify

No numerical values should be inserted until the simulations or experiments have been completed.

7. Threats to Validity and Limitations

Several limitations must be considered.

Construct validity: Response time, cost, and control coverage may not capture every aspect of application quality or security. The study should justify its metrics and measurement boundaries.

Internal validity: Differences in workload, instance configuration, network conditions, or experimental setup may influence results. All compared methods should use consistent conditions.

External validity: Simulation results may not generalize directly to commercial cloud platforms, different regions, or production applications. Real-world validation is required.

Model validity: Security requirements and residual-risk indicators may be difficult to represent numerically. Mandatory controls should be independently verified rather than inferred from an aggregate score.

Decision sensitivity: The selected configuration may depend on stakeholder priorities, pricing assumptions,

or the normalization method. Sensitivity analysis is therefore essential.

Novelty limitation: Existing research already investigates multi-objective scheduling, cloud infrastructure selection, and security-aware optimization. The proposed framework's distinct contribution must be established through a rigorous review and comparative evaluation.

8. Conclusion and Future Work

This paper proposes a Multi-Criteria Cloud Application Optimization Framework for evaluating cloud configurations against application performance, security, and cost requirements. The framework formulates configuration selection as a constrained multi-objective optimization problem, validates mandatory security controls, measures application-level outcomes, and uses Pareto analysis to expose trade-offs among feasible solutions.

The central design principle is that mandatory security requirements must remain constraints rather than variables that can be exchanged for faster response times or lower expenditure. Performance and cost can then be optimized within the feasible configuration space, with final selection based on documented stakeholder priorities.

The proposed methodology establishes a basis for systematic experimentation using cloud simulation or controlled deployment. Future work should implement the framework, compare it with established optimization methods, measure the overhead of configuration evaluation, and investigate its robustness under changing workloads and pricing conditions. A systematic literature review should also determine whether the proposed combination of security constraints, application-level metrics, and Pareto-based selection offers a sufficiently distinct contribution.

The framework should be considered a research proposal until its implementation and empirical evaluation are completed. Claims of improved performance, lower cost, or superior security must be supported by reproducible measurements.

References

- [1] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*. NIST Special Publication 800-145, National Institute of Standards and

Technology, 2011.
<https://doi.org/10.6028/NIST.SP.800-145>

- [2] J. Araujo, P. Maciel, E. Andrade, G. Callou, V. Alves, and P. Cunha, "Decision making in cloud environments: An approach based on multiple-criteria decision analysis and stochastic models," *Journal of Cloud Computing*, vol. 7, article 7, 2018.
<https://doi.org/10.1186/s13677-018-0106-7>
- [3] "Many-Objective Optimization-Based Task Scheduling in Hybrid Cloud Environments," *Computer Modeling in Engineering & Sciences*, vol. 136, no. 3, 2023, pp. 2425–2450.
<https://doi.org/10.32604/cmescs.2023.026671>
- [5] F. J. de Vault, E. D. Simmon, and R. B. Bohn, *Cloud Computing Service Metrics Description*. NIST Special Publication 500-307, National Institute of Standards and Technology, 2018.
<https://doi.org/10.6028/NIST.SP.500-307>
- [6] C. Tunc, S. Hariri, M. Merzouki, C. Mahmoudi, F. J. de Vault, J. Chbili, R. B. Bohn, and A. Battou, "Cloud Security Automation Framework," in *Proceedings of the IEEE Workshop on Automated Cloud Configuration and Operations*, 2017.
<https://doi.org/10.1109/FAS-W.2017.164>

