Threat-Path Analysis and Residual Exposure in Distributed Cloud Environments: Credentials, Privilege, Tenant Isolation and Software Supply Chains

[1*]Raghvendra Rao,[2*]Dr. Yashpal Joshi
[1*] Phd. Research Scholar, Department of Computer Science and Engineering, Saveetha Amarawati University, Vijayawada, Andhra Pradesh, India
[2*] PhD Guide, Department of Computer Science and Engineering, Saveetha Amarawati University, Vijayawada, Andhra Pradesh, India

Abstract
Distributed cloud services depend on identities, policies, shared infrastructure and automated delivery pipelines that span regions and providers. Security assessments often list controls without showing which attack paths they interrupt or what exposure remains afterwards. This paper presents a threat-path approach that follows each threat from its entry point through the authority it obtains, the resources it can reach and the harm it can cause, to containment and a stated residual exposure. Ten threats are analysed—including credential theft, excessive privilege, tenant isolation failure, supply-chain compromise and secret exposure—and mapped to seven control planes. Two analytical models show how residual exposure should be quantified. For a stolen token, the expected misuse window depends on both token lifetime and the time to detect and revoke: with one-hour detection and revocation, a 24-hour token gives an expected window of 59 minutes, compared with 12 hours if it is never revoked. For layered defences, a shared bypass dominates: four independent layers each bypassed with probability 0.1 give a joint failure probability of 10^{-4}, but a 1% common-mode bypass raises this to about 10^{-2}. The analysis supports short-lived scoped credentials, fast revocation, separation of destructive authority and independent verification paths, and it recommends that residual exposure be documented for every threat path. All numerical values are stated assumptions, not measurements.

Index Terms—*Distributed cloud, threat modelling, residual risk, credential theft, least privilege, tenant isolation, software supply chain, zero trust, defence in depth*

1. Introduction

Cloud computing provides on-demand access to pooled and elastic resources [1]. Distributing services across regions and providers improves capacity and fault isolation, but it also multiplies the identities, interfaces, administrative paths and software components that an attacker can target. When an organisation uses public cloud services, security responsibilities are shared between the provider and the customer, and the customer must understand which responsibilities remain its own [2].

Many cloud incidents originate in ordinary operational mechanisms rather than exotic attacks: misconfiguration, faulty updates and hidden dependencies cause a large share of outages [3], and the same mechanisms can be exploited deliberately. Security and reliability are therefore attributes of the same dependable system [4]. A control catalogue alone does not show whether these mechanisms are covered. Assessors need to know which path a control interrupts, which path it leaves open and how much exposure remains when it works as intended.

This paper, derived from a doctoral thesis on security and reliability challenges in distributed cloud environments, contributes:

1) A threat-path model that follows each threat from entry to residual exposure, and a mapping of ten threats to seven control planes.
2) An analytical model of the credential misuse window as a function of token lifetime and detection-and-revocation time.
3) An analytical model of layered containment that shows how a shared bypass limits the benefit of additional layers.
4) Evidence requirements and residual-exposure statements that turn proposed controls into reviewable decisions.

2. Related Work

2.1 Identity-Centred Security

Zero trust architecture removes implicit trust based on network location and evaluates each request against identity, context and policy [5]. For cloud-native applications across multiple clouds, NIST describes identity-tier and network-tier policies enforced at each service [6]. These approaches build on long-standing principles such as least privilege, separation of privilege and complete mediation [7], and on cloud access-control guidance that addresses multi-tenant enforcement [8].

 ISSN : 3139-1141

2.2 Isolation and Supply-Chain Risk

Shared infrastructure creates isolation risks: co-resident virtual machines can be located and used to extract information through shared hardware [9]. Containers add lifecycle risks in images, registries, orchestrators and hosts [10], and microservice architectures add many service-to-service interfaces [11]. Supply-chain attacks target dependencies, build systems and distribution channels before software reaches production [12]. Secure development practices [13] and pipeline integrity controls [14] aim to ensure that released artefacts can be traced to reviewed source.

2.3 Research Gap

Existing guidance describes controls in depth but rarely quantifies what remains after a control operates. In practice a revoked token may stay usable while revocation propagates, a detector may miss low-volume exfiltration, and several layers may share one identity provider. Residual exposure of this kind is seldom stated explicitly, which encourages overconfidence.

3. Threat-Path Model

3.1 Path Structure

Fig. 1 shows the model. Each threat begins at an entry point, such as a stolen token, an exposed key or a compromised build. It obtains some authority, defined by the scope and lifetime of the identity it misuses. That authority determines its reach: the tenants, data and actions it can access. Reach determines the possible harm, such as exfiltration, destruction or a hidden change. Controls act on one or more steps by reducing authority, limiting reach or enabling containment. The final step states the residual exposure that remains after controls work as designed.

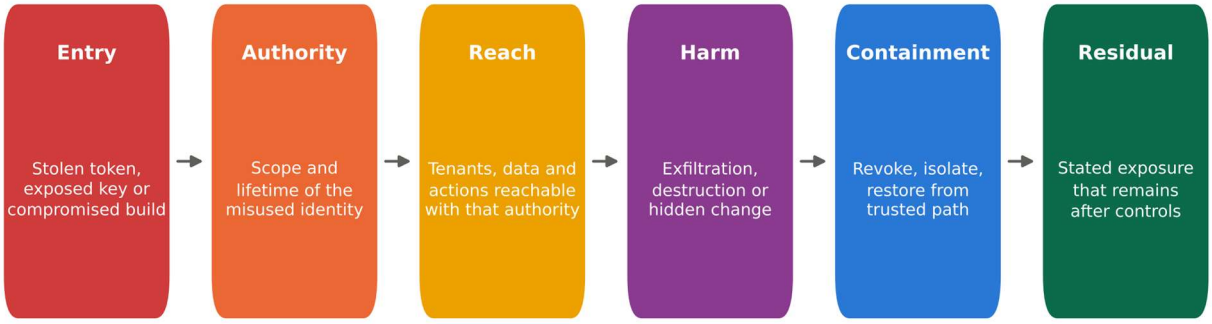

Fig. 1. Threat-path model. Each threat is followed from entry through authority, reach and harm to containment and a stated residual exposure.

3.2 Threat Catalogue

Table I lists the ten threats analysed in this study, the main control direction for each and the evidence that would show the control is effective. The evidence column matters because a configured control is not necessarily an effective one: a clean infrastructure template does not prove that production matches it.

TABLE I THREATS, CONTROL DIRECTIONS AND REQUIRED EVIDENCE

Threat	Control direction	Evidence of effectiveness
Credential theft	Short-lived, scoped credentials; rapid revocation	Issuer logs, access patterns, revocation timing
Excessive privilege	Separate read, change and destructive authority	Effective-permission review; denied-operation tests
Tenant isolation failure	Derive tenant context from trusted identity; enforce at data access	Cross-tenant access tests; ownership checks
Insecure interfaces	Validate request meaning; per-operation resource limits	Negative tests; resource profiles per query
Configuration exposure	Policy as code; drift detection	Deployed-state comparison with approved baseline
Supply-chain compromise	Controlled path from reviewed source to signed artefact	Provenance records; artefact inventory
Secret and key exposure	Managed custody; rotation with propagation checks	Rotation logs; secret scanning results
Data exfiltration	Export approval; scoped datasets; transfer monitoring	Export audit trail; destination ownership

 ISSN : 3139-1141

Threat	Control direction	Evidence of effectiveness
Denial of service and abuse	Budget work by cost and priority, not only rate	Resource use per operation; rejection records
Insider and administrative risk	Explicit targeting; independent review of destructive actions	Elevation records; change confirmation

3.3 Mapping Threats to Control Planes

The integrated framework in the thesis organises controls into planes for identity, policy, network, data, telemetry, delivery and recovery. Fig. 2 maps each threat to the planes that provide its primary and supporting controls. Policy and telemetry give primary coverage for most threats, because authority must be constrained and its use must be observed. No threat is covered by a single plane, and supply-chain compromise and secret exposure depend on the recovery plane: after such an incident, the organisation must be able to rebuild from trusted artefacts and keys rather than from the compromised path.

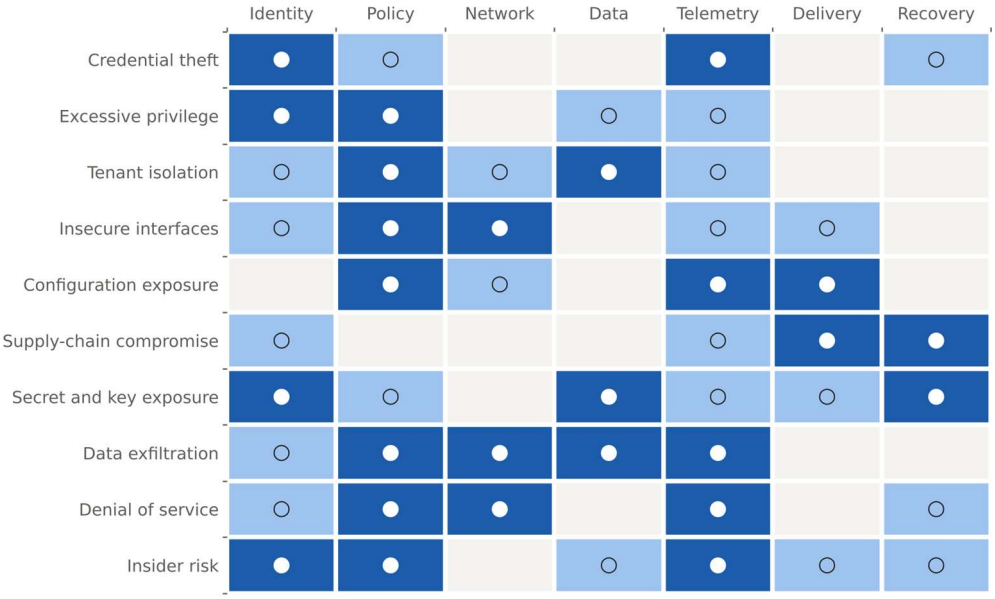

Fig. 2. Mapping of threats to control planes. Filled circles mark primary control planes and open circles supporting planes. The mapping is a design analysis, not an audit result.

4. Quantifying Residual Exposure

4.1 Credential Misuse Window

A stolen credential remains usable until it expires or is revoked. Let L be the token lifetime and R the combined time to detect misuse and complete revocation. If theft occurs at a random point in the token's life, the remaining lifetime U is uniformly distributed on $[0, L]$, and the misuse window is $\min(U, R)$. Its expected value is

$$E[W] = L/2 \text{ if } L \le R; \quad E[W] = R - R^2/(2L) \text{ if } L > R \qquad (1)$$

Fig. 3 evaluates Eq. (1). Without revocation, the expected window is half the token lifetime, so a 24-hour token gives 12 hours. With one hour to detect and revoke, the same token gives 59 minutes, and with 15 minutes it gives 15 minutes. Shortening token lifetime helps most when revocation is slow: with four-hour revocation, cutting lifetime from 24 hours to 4 hours reduces the expected window from 220 to 120 minutes. The model shows that lifetime and revocation speed must be designed together. It assumes that revocation propagates completely; cached authorisations or long sessions extend the real window and must be measured.

 ISSN : 3139-1141

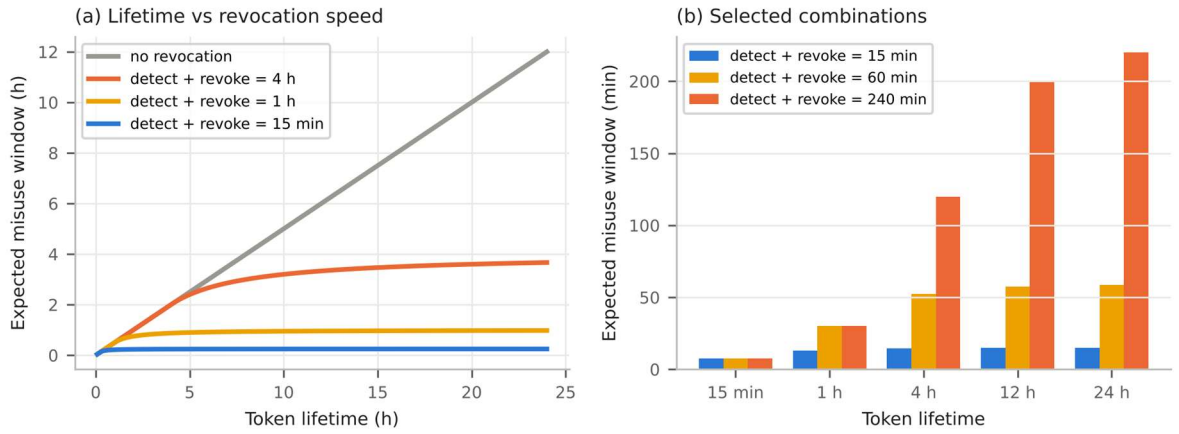

Fig. 3. Expected credential misuse window (Eq. 1). (a) As a function of token lifetime for different detection-and-revocation times. (b) Selected combinations of lifetime and revocation time. Analytical illustration under stated assumptions.

4.2 Layered Containment and Common-Mode Bypass

Defence in depth places several controls on a threat path. If each of k layers is bypassed independently with probability q, the probability that all fail is q^k. Real layers often share a dependency, such as one identity provider, one policy engine or one administrative account. If such a shared condition bypasses every layer with probability c,

$$P_{all} = c + (1 - c)\, q^k \qquad (2)$$

Fig. 4 shows the result with $q = 0.1$. With independent layers, four layers reduce the joint failure probability to 10^{-4}. A shared bypass of only 0.1% raises this to 1.1×10^{-3}, and a 1% shared bypass to about 1.0×10^{-2}; beyond three layers, additional layers add almost nothing. The practical conclusion is that removing shared dependencies between layers—for example, giving the recovery plane its own identity and approval path—can be worth more than adding another layer that shares the same weakness.

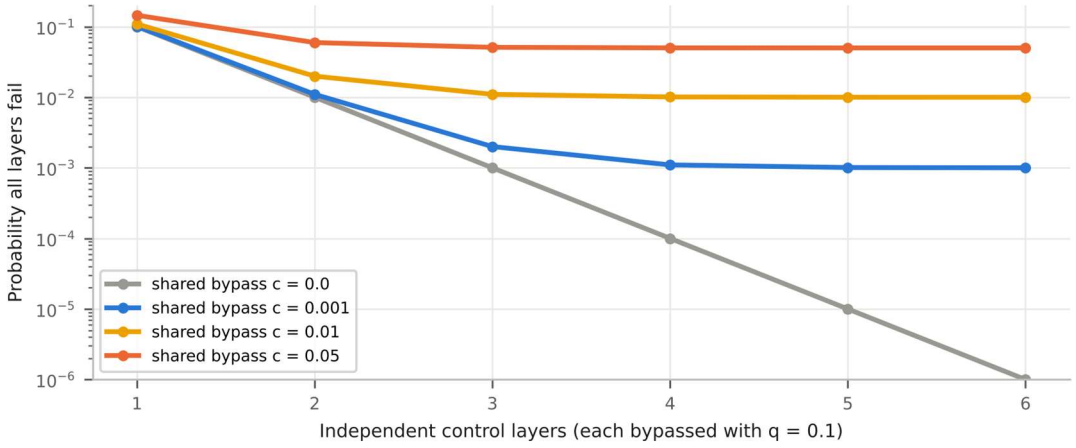

Fig. 4. Probability that all control layers on a threat path fail, as a function of the number of layers, for different common-mode bypass probabilities (Eq. 2). Analytical illustration.

5. Threat Analyses

5.1 Credential Theft and Excessive Privilege

Stolen credentials let an attacker act through apparently normal interfaces, so detection must examine what an identity does and whether that fits its task. Response distinguishes identification, containment and restoration: replacement authority must be issued through a trusted path, and actions taken during the exposure window must be reviewed. The impact of a stolen credential is bounded by its privilege. Separating read, change and destructive authority, and requiring explicit targeting and independent confirmation for destructive actions, limits reach even when a credential is misused [7], [8].

5.2 Tenant Isolation

Tenant boundaries exist in the hypervisor, kernel, network, identity system and data model. Infrastructure isolation does not remove application-level risk: a

tenant identifier supplied by a client must not be trusted merely because authentication succeeded. Isolation must also hold for background jobs, caches, search indexes and analytics exports, which can create secondary paths that bypass the primary database check. Evidence should therefore follow a record through its lifecycle, and side-channel risks from shared hardware should be considered for sensitive workloads [9].

5.3 Supply Chain, Secrets and Keys

A compromised build inherits ordinary deployment trust and can affect many releases, including versions that passed tests [12]. Protection requires a controlled path from reviewed source to signed artefact, and the signing service must not accept arbitrary output from an untrusted job [13], [14]. Response requires an artefact inventory and a way to identify the last trustworthy build. Secrets differ in consequence: a leaked database password exposes one system, but a leaked signing key can create authority across many. Key management is also a continuity dependency, because inaccessible keys can make healthy data unusable, so protection against disclosure and protection against loss need coordinated custody [15].

5.4 Residual Exposure Statements

Table II gives examples of residual-exposure statements. Each states what remains after the control works, so that the owner can decide whether further action, an interim restriction or explicit acceptance is required.

TABLE II EXAMPLES OF RESIDUAL-EXPOSURE STATEMENTS

Threat path	Control in place	Residual exposure to state
Stolen workload token	One-hour tokens; revocation on detection	Token usable until revocation propagates; cached authorisations
Cross-tenant read	Tenant derived from trusted identity at data layer	Secondary paths such as caches and exports not yet tested
Malicious dependency	Signed artefacts from controlled pipeline	Dependencies trusted at build time are not re-verified
Low-volume exfiltration	Export approval and transfer monitoring	Small, slow transfers below detection thresholds
Destructive administrator action	Independent confirmation for deletion	Shared administrative identity across planes

6. Discussion

The threat-path model changes the question asked of a control from “is it present?” to “which path does it interrupt, and what remains?”. Its main practical lessons follow from the two analytical models. First, token lifetime and revocation speed should be engineered together, and revocation should be tested for propagation as well as issuance. Second, layered defences are only as independent as their dependencies; shared identity, policy or administrative paths should be identified and, where consequences are severe, separated. This is consistent with zero-trust guidance [5], [6] and with the cyber-resilience goal of limiting the effects of a successful compromise [15].

The study has limitations. The threat catalogue and plane mapping are design analyses rather than audit results, and the numerical values are stated assumptions chosen to illustrate the models. The uniform-theft assumption and the common-mode model are simplifications. Empirical work should measure revocation propagation, detection delay and shared dependencies in deployed systems, and repeat the analysis when trust relationships change.

7. Conclusion and Future Work

This paper presented a threat-path approach for distributed cloud environments that follows ten threats from entry to residual exposure and maps them to seven control planes. Two models showed how to quantify what remains: the expected credential misuse window falls from 12 hours to 59 minutes for a 24-hour token when misuse is detected and revoked within an hour, and a 1% common-mode bypass limits four independent layers to about a 10^{-2} joint failure probability. Future work will apply the method to deployed services, measure the inputs that the models assume and integrate residual-exposure statements into continuous threat-model review.

References

[1] P. Mell and T. Grance, “The NIST definition of cloud computing,” National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-145, 2011, doi: 10.6028/NIST.SP.800-145.

[2] W. Jansen and T. Grance, “Guidelines on security and privacy in public cloud computing,” National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-144, 2011, doi: 10.6028/NIST.SP.800-144.

[3] H. S. Gunawi *et al.*, “Why does the cloud stop computing? Lessons from hundreds of service outages,” in *Proc. 7th ACM Symp. Cloud Comput. (SoCC)*, 2016, pp. 1–16, doi: 10.1145/2987550.2987583.

 ISSN : 3139-1141

[4] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, 2004, doi: 10.1109/TDSC.2004.2.

[5] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-207, 2020, doi: 10.6028/NIST.SP.800-207.

[6] R. Chandramouli and Z. Butcher, "A zero trust architecture model for access control in cloud-native applications in multi-cloud environments," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-207A, 2023, doi: 10.6028/NIST.SP.800-207A.

[7] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975, doi: 10.1109/PROC.1975.9939.

[8] V. C. Hu, M. Iorga, W. Bao, A. Li, Q. Li, and A. Gouglidis, "General access control guidance for cloud systems," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-210, 2020, doi: 10.6028/NIST.SP.800-210.

[9] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, 2009, pp. 199–212, doi: 10.1145/1653662.1653687.

[10] M. Souppaya, J. Morello, and K. Scarfone, "Application container security guide," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-190, 2017, doi: 10.6028/NIST.SP.800-190.

[11] R. Chandramouli, "Security strategies for microservices-based application systems," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-204, 2019, doi: 10.6028/NIST.SP.800-204.

[12] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "SoK: Taxonomy of attacks on open-source software supply chains," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2023, pp. 1509–1526, doi: 10.1109/SP46215.2023.10179304.

[13] M. Souppaya, K. Scarfone, and D. Dodson, "Secure software development framework (SSDF) version 1.1," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-218, 2022, doi: 10.6028/NIST.SP.800-218.

[14] R. Chandramouli, F. Kautz, and S. Torres-Arias, "Strategies for the integration of software supply chain security in DevSecOps CI/CD pipelines," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-204D, 2024, doi: 10.6028/NIST.SP.800-204D.

[15] R. Ross *et al.*, "Developing cyber-resilient systems: A systems security engineering approach," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-160 Vol. 2 Rev. 1, 2021, doi: 10.6028/NIST.SP.800-160v2r1.