Secure and Extensible Platform Architectures for Enterprise Network Orchestration

Manevannan Ramasamy

Software Engineering Senior Technical Leader, Milpitas, California, United States of America

Email ID: manevannan3@gmail.com

Abstract

Products related to enterprise network orchestration are expanding to include heterogeneous infrastructure, automation operations, use of third parties and programmable interfaces and have become a core conflict between extensibility, operationsecurity. An example of recommended safe and scalable platform design in this paper is one that imposes definite boundaries of trust between the users, services, devices adapters, automation processes and execution platforms. The architecture separates the central control plane, integration and execution territory and is based on workload identity, policy enforced access, least privilege access, intent checking, managed secret use, auditing (tamper evidenced), and software provenance checking. The versioned APIs are event based interfaces and narrow adapters to ensure extensionality without the need to use unconstrained plug-in and database dependency. Other functions that are incorporated in the proposed architecture are pre-execution policy-enforcement, capability based isolation, managed failure and continuity checking of extension behavior. A framework assessment looks at the effectiveness of authorization, component isolation, supply-chain integrity, audit completeness, policy enforcement and safe failure behavior in representative orchestration scenarios. This discussion shows that extensible does not mean compromising the platform security in the event they are managed as independent managed capabilities whose authority, data access and resource consumption are well delimited. The article lays down a viable constructionist base of architecture.

Keywords Secure Architecture, Network Orchestration, Zero Trust, Workload Identity, API Security, Software Supply Chain

DOI: 10.21590/ijhit.06.01.07

1. Introduction

Network orchestration systems have also become a notable strategic control plane of enterprise infrastructure that they arrange configuration, monitoring, policy implementation, service provision and automated recovery on heterogeneous network

systems. In contrast to traditional information systems, such platforms often contain privileged credentials, topology data, device catalogs, work policies, and have the power to edit configurations of large groups of network devices. An extension or erroneously granted extension may then leak out of a single application boundary, and into multiple domains in the infrastructure. The security design of an orchestration platform should thus be able to cover not just authentication and confidentiality but also the limits of authorization, execution controls, software integrity, auditability, and operating side effect containment.

Another architectural issue is getting extensibility. Enterprise networks are seldom comprised of equipment of one vendor, or technology generation. The orchestration platforms will have to be integrated with routers, switches, firewalls, wireless controls, and cloud networking services, monitoring, ticketing platforms, identity providers and other tools used in the operation. This interoperability is facilitated by extensions like device adapters, policy modules, event processors and reporting services and automation workflows. These constituents however, do not have equal authority levels. An extension with a simple inventory-read-only interface can be linked to a minor operational risk but with an extension that creates configurations, manages access-control rules, or responds to network-events it may impact a larger portion of the infrastructure. The consideration of extensions to be equally reputable, therefore, predisposes exposure to unwarranted exposure.

These do not lend themselves very well to more traditional perimeter-based methods of security as the network location cannot be trusted to exude trust. They can exist within the same infrastructure yet they exhibit a much different set of responsibilities and privileges including internal services, APIs, adapters, workflows, and plug-in. A particularly perilous element associated with one of the vulnerabilities acquired by an internal component compromised can be the large range of credentials it possesses or the multiple access (unlimited access) to orchestration services. Security should, therefore, be determined at the level of individual subjects, workloads, capabilities, resources and requested actions as opposed to being determined only at network boundaries.

The definition for zero trust architecture by the National Institute of Standards and Technology (NIST) offers a valuable basis in terms of helping to overcome this issue. NIST has concentrated on the fact that one does not need to implicitly trust to unleash access based on the network location but must become the master of the access choice on an ongoing basis as per the subjects, assets, resources, and implemented policies [1]. When applied to an orchestration platform, the consequences of this principle are that every user, every service, every device adapter, every workflow, and every automation element must have an explicitly defined identity and can only be allowed to run its explicitly declared permissions. Authentication thus leads to initiation of an access decision, and not a sign of unbridled trust.

This generalization applies particularly to cloud-native orchestration environments where the functionality is typically distributed across microservices, APIs, event brokers, workloads in packages or containers, and outside services. The NIST guideline on cloud-native applications focuses on workload identity, policy enforcement, service-to-service

protection and directed communication between distributed components [2]. The architecture of the mechanisms through which the ideas can be implemented to the surroundings of network orchestration includes the mechanisms to differentiate legitimate and service contacts against malicious or malformed feedbacks. Having identity-aware communication, limited Scopes of service credentials, policy-based authorization and control-plane and execution isolation can help mitigate effects of individual component compromise [3] [4].

Security cannot however be limited by the capability to control who may invoke an orchestration service. Network automation is another problem that creates a question of whether an action that is requested is operationally safe and valid. Network intent might be too broad, contradictory or may cause havoc to the network even with an authenticated user [5]. Likewise, a legitimate automation workflow can respond to an incident in a way that would set undesirable adjustments in numerous gadgets. The platform thus needs a middle-ground validation, which checks intent, policy constraints, scope of target, dependencies and side effects before execution. This offers two options one being requesting an operation and the other one being an automatic execution of the operation [6].

The other concern is the processing of extensions. Conventional plug-in mechanisms may provide an easy way to create a feature, but may give inappropriate access into internal state of the application, shared databases, credentials, or execution resources [7]. The interfaces are extensible via versioned APIs, event-driven interfaces, capability-based permission and controlled perimeter boundaries with more secure architecture. These mechanisms enable extensions to not only evolve but also not to become implicit parts of the trusted computing base of the platform. This concept of least privilege can thus be applied to human users, or to software components or automated processes [8].

It is also important that the orchestration activities can leave huge changes in the infrastructure which will be auditable. An implementation safe platform should include record connecting the identity providing the request, the purpose being pursued, authorization decision, an extension or workflow involving, execution command, and state of a gadget. Evident Audit reports could be consolidated to help in incidents investigation, compliance testing, accountability of operations and recovery evaluation. At the same time, we can apply software provenance systems to know whether extensions and dependencies are the outcome of an approved source, and have preserved the intended integrity throughout the supply chain of the software [9] [10].

It is on this basis that this article suggests secure and scalable platform architecture of enterprise orchestration of networks. The architecture separates the core control plane, integration and execution space and integrates workload identity, policy-based authorization, intent validation, controlled secret management, software provenance, controlled extensibility, and tamper-evident auditing. The framework test is focused on permission, solitude, supply symptoms, audit entirety and non-fail facilities. The basic point is that extensibility and security do not necessarily conflict goals: extensibility can be a property of the architecture in the first place when all extensions are treated as

explicitly managed capabilities with limited authority, access to data and use of resources.

2. Zero-Trust Control Fabric and Capability-Isolated Orchestration Architecture

The architecture, shown below is constructed of a zero-trust control fabric in which identity, authorization, policy, execution and evidence are all treated as explicit and architectural functions rather than consensus properties of network location. The platform isolates components by their power and region of operation thereby minimizing impacts of component compromise and dissolving extensibility mechanisms that gain uncontrolled access to control. It has five major trust zones, i.e.: interaction, core control, integration, execution, and data. These compartments are connected through proven authenticated and policy-regulated interfaces and refined capability boundaries mediate the privileged operations.

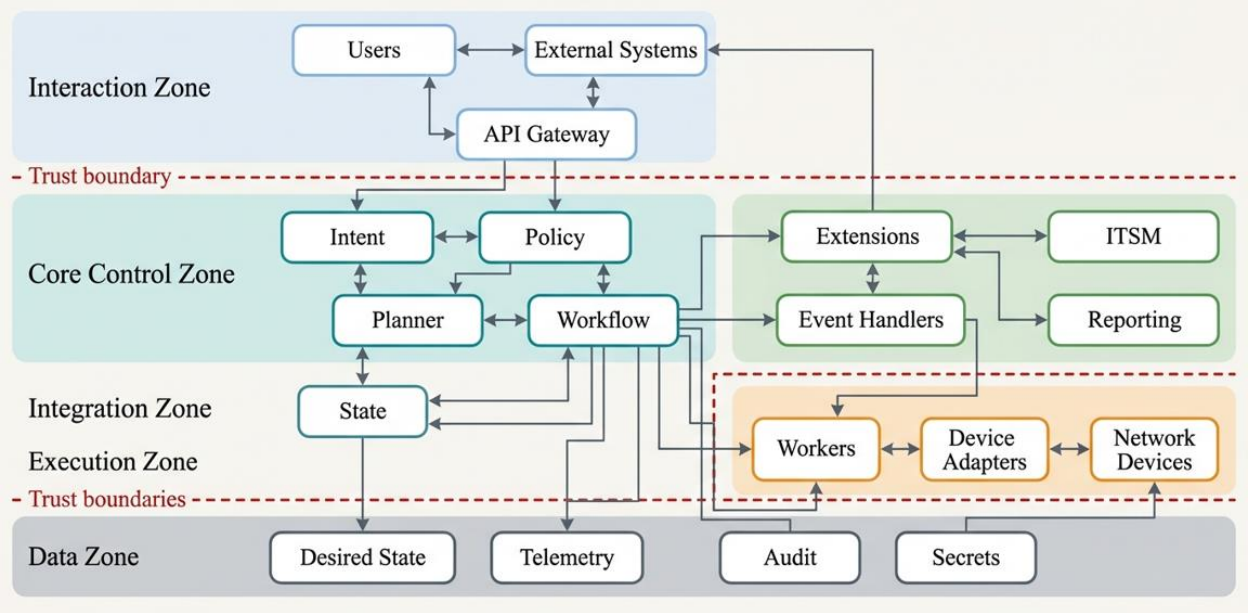

Figure 1- Zero-Trust Orchestration Platform Architecture

2.1. Trust-Zone Architecture

The interaction zone ends the connections of a human user and outside systems. The API gateway provides a protocol-level security entry point with authenticators verifying the identity of the caller, checking request format and request content quotas, rate rule, and making available an identity context to subsequent requests. Nonetheless, a gateway authentication is not deemed as adequate authorization. Resource scope, requested action, the operational environment, purpose as well as current policy must be taken into

consideration before denying access. This prevents a successful authentication of the caller in order to have an implicit access to infrastructure resources.

The core control zone has the most trusted logical services of the platform which includes; intent management, policy assessment, planning, workflow coordination and authoritative state management. It is a strictly narrow point of entry that is under the control of clearly defined interfaces. Core services do not add third-party code directly into their processes nor do they open their internal databases to outside extensions. There is a workload identity and permission set per service. So, topology service, a policy service or workflow component compromise does not give privileged access to other core services by default. The isolation gives the security boundaries of service levels even to the control plane itself.

The integration zone should provide controlled extensibility in event handlers, IT service-management connectors, reporting program, notification system and approved third party extensions. Integrations take events and subscribe to them, instead of reading internal state. Each integration possesses a workload, resource quota and defining policy-bounding communication boundary. Destinations that are outbound are explicitly listed as allow listed and the fields on sensitive events are removed, unless the fields can be sensitively used by the receiving integration because the business and security requirement of requirement of those fields is established. This model can allow the platform to have a broad ecosystem to work, but it cannot rely on everything as integration and make it a core of the core.

The implementation area consists of gadget adapters and individuals that translate approved orchestration plans into infrastructure actions. These sections can access only managed resources on a one-time basis and are not receptive to arbitrary user commands and integrations. Instead, work materials are signed and approved among the employees who contain absolute scope of operation and program limitations. The limits of network reachability are those of necessary targets and protocols. Once the execution is finished, workers hand over external operation identifier, separation of execution and all the evidences associated with it to the workflow service. This provides a narrow zone of control on decision making and side effects outside.

The data zone offers various protections to desired state protection, operating data protection, telemetry protection, audit evidence protection and secrets. These data classes will need varying access, encryption, retention, backup and recovery policies. Separation also eliminates what seem to be innocuous data service routes as secondary privilege-escalation vectors. One example of this is that the approved operational measures can be accessed by a reporting query but the configuration credentials and unsecured data regarding the client identification are not readable. Data classification is thus a component of the authorization architecture and not a storage issue.

2.2. Identity and Policy-Centric Authorization

Strong authentication mechanisms are used to authenticate human identities after which they are evaluated using role-based and scope-based authorization. Service identities are better to be dynamically issued and rotated automatically than being configured or shared

across numerous workloads. SPIFFE proposes a scheme of certifiably identities in heterogeneous settings [3]. Regardless of the adopted workload-identity mechanism (SPIFFE or another) the architecture must associate identity with workload attributes and lifecycle state with minimal reliance on long-lived shared credentials.

Decisions of authorization consider various dimensions as opposed to just making use of identity. The policy decision process considers the main principle of the action, action, resources, scope, purpose and contextual conditions of any request. As an example, a topology service can have permission to access the identity of devices and connectivity information but cannot be allowed to perform configuration modifications. Diagnostics may be required of an assurance service which cannot circumvent change-management policy. Similarly, a device adapter can be provided with a capability-limited token, allowing a specific operation to a specific target, temporarily at least. Effective determinations (government prevails the case), and denials including an act of the policy that was in place during the changes are recorded as audit evidence.

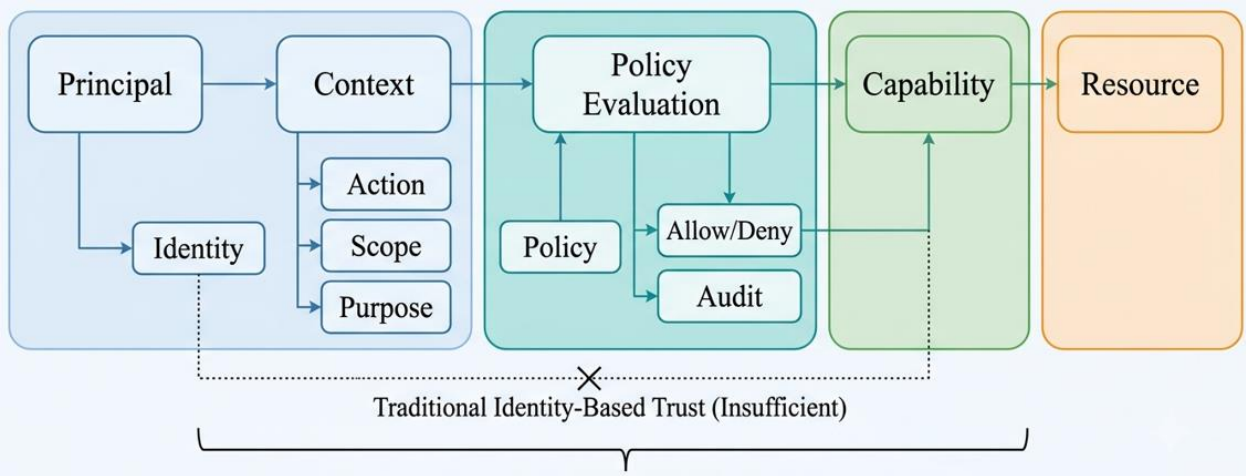

Figure 2- Capability-Bounded Trust Model

Network intent is also subjected to stringent action. The identification of a requester who is empowered to dictate the desired result, the constraints suggested must be consistent with the organizational policy and the requested scope should fall within delegated authority will be identified by admission control. The resultant implementation strategy is again assessed as translation of the high-level intent to device-specific operations possibly unveils high-impact actions that cannot be seen initially in the request. It is a two-step procedure, which seems intent authorization, and execution authorization and provides an additional check on unsafe automation.

2.3. Execution Security and Evidence-Bound Automation

A sufficient security context is included in each executable work item to specify why, what, where and when an operation was authorized. These, at the very least, are the accepted intent generation, plan identity, target scope, authorized type of operation, time expiration, and verification requirements. Work items that are stale, un-signed, malformed, or out-of-scope are rejected by execution workers. Idempotency keys also curb replayed requests that end up duplicating changes to infrastructure inadvertently.

The approval of step-up or some other approvals may be made in terms of the organization policy and constitute high impact operations. This allows the platform to distinguish more between the daily automation and those that have vast operational ramifications. The resulting control tool can incorporate automation of low risk processes and other points of control modification of critical services, security, or wide scales of infrastructure.

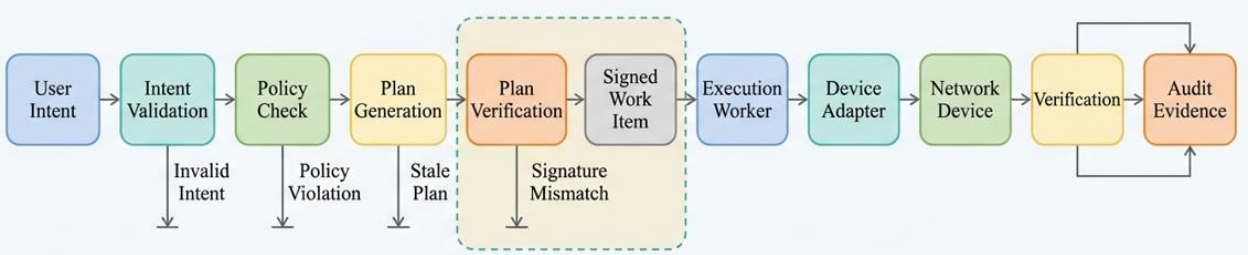

Figure 3- Secure Intent-to-Execution Pipeline

The contents of secrets are accessed during execution and reside out of workflow payloads, event streams, application logs, and regular audit history. Credentials are supposed to be short lived and only to the capability needed where appropriate infrastructure is in place. Old-fashioned statically credentials demand further seclusion, surveillance, and inspection. Even ongoing workflows should also be factored in routing processes so as not to invalidate legitimate operations unintentionally by replacing credential or lead to unreasoned recovery behavior.

Software integrity is another point of input responses. The platform metadatas (including extensions, image containers, and regular metadatas), deployment metadatas, and platform components can be verifiably provenance. SLSA framework provides a foundation to improve software-supply-chain integrity and develop provenance [4]. Releases and extensions should be appropriately associated with authorized sources and builds, and signed when necessary, scanned to known vulnerabilities, and verified before being accepted entry into privileged areas. The runtime policy should invalidate the capabilities of unauthorized images, packages or dependence to be executed within the sensitive control or execution environments.

2.4. Secure Extensibility Fabric

Versioned intent and integration APIs is the primary surface of extension. They offer secure contracts yet lack external factors dependant on internal execution particulars. The message event communicates consistent facts on a platform and ought to have a consistent schema, label of versions, and sensitivity. The webhooks have to be authenticated meanwhile they have to possess replay protection, limited number of retries and dead-lettering to ensure that when the external consumers fail, there is no impact on the control plane. The effectiveness of these patterns of extensions is evident in public northbound intent APIs and event-based integration as written on systems such as Cisco Catalyst Center [5].

Greater controls are placed on device adapters since they use those abstract orchestration requests into external side effects. The adapters should as well clearly indicate the models of the supported devices, the protocols, the behavior of the transactions, rollback capabilities and the network connectivity requirements. The unsupported inputs are to be discarded, permissions should be granted and structured execution evidence should be back by conformance validation. This brings about an adapter which is unrestricted plug in into an imposed capability provider.

Untrusted or independently written code in-process plug-ins, as a rule, is not a good fit since they are located in a similar host-execution environment. Where this cannot be mitigated through counter-measures, the component needs to be run in a sandbox where resources are not controllable, system calls are restricted, file system access restricted, and default network connectivity is turned off. It must interface with the platform along limited capability interface. The termination of extensions should then be separated too that an authoritative state cannot be hijacked by a failed or malicious part, obstruct an unrelated workflow or encroach on fundamental control facilities.

Generally, architecture proposes high degrees of extensibility as a feature that is controlled and not a form of implicit trust. The identity will determine who performs to act, the policy will be what to perform, what limits of ability will determine where and when it can be done, execution controls will allow external influences when it is permitted, and audit evidence will specify what occurred. This layered architecture allows enterprise orchestration platforms to increase in size using APIs, events, adapters, and automation services without necessarily depending on a separation between trusted control functions and possibly less-trusted extensions.

3. Evaluation Framework

The framework assessment will verify the capability that the proposed secure orchestration architecture has to preserve boundaries to authorization, component isolation, software integrity, execution safety and auditability under both non-adversarial normal and adversarial operation conditions. Instead of looking solely at successful performance of the service, the evaluation looks at how the unauthorized actions are

systematically avoided, failures are limited and adequate evidence is kept in order to re-create security-relevant incidents.

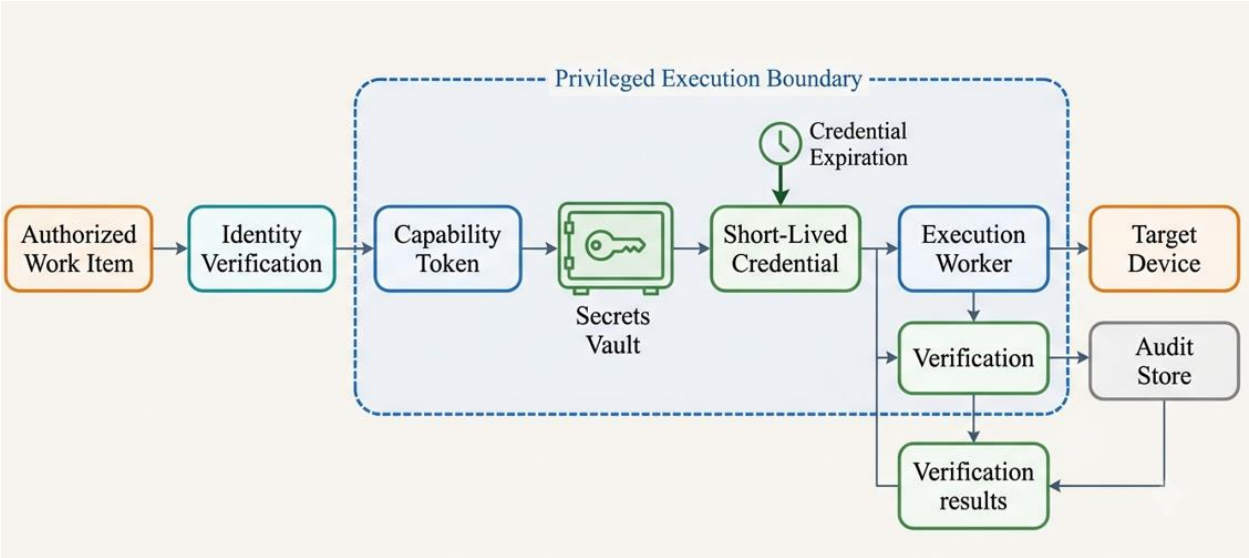

Figure 4- Protected Execution and Secret Lifecycle

3.1. Authorization and Access-Control Assessment

Security testing begins with a grid analysis involving a methodical grid of principals, activities, assets and situational circumstances. The matrix represents human and workload identities and assigns their authorized capabilities to the particular infrastructure resource, and extent of operations. The test cases intentionally strive to replicate issues of horizontal privilege escalation between equal services, vertical escalation between less-privileged and more-privileged roles, replaying tokens, stale plan executions, cross-tenant access, unintended event subscriptions, extracting secrets, and direct access to the database.

The metrics will be assessed using the percentage of operations being denied, legitimate denied, policy decisions latency and the completeness of audit records together with the operations. Capability-limited tokens and contextual authorization are given special consideration since access should not be granted beyond what is declared in resource and action -scope of the authorization. The policy-version recording is also reviewed to move forward based on the probability that an authorization decision can be ascribed to that particular policy up to the state at which it was made [11].

3.2. Isolation and Resource-Containment Assessment

The isolation testing rule is that one wants to find out whether there was a chance of bad performing, malfunctioning or intensive extension affecting core orchestration services. Every outbound extension is subjected to controlled CPU and memory exhaustion, excessive queue generation, repeated and unsuccessful API calls to create outbound

connections. The test indicates the capability of the quotas, admission controls, and communication policies to contain the extension without disrupting the major functions of control-plane.

Particular emphasis is put on the central availability of the admission and emergency-control services. Such functions cannot out exceed predetermined operational targets during periods when one of the elements of integration has exhausted resources. It is therefore an analysis considering both of the direct resource consumption as well as the indirect effects that encompasses saturation of queues and depletion of connections and over processing of events.

3.3. Software Supply-Chain and Recovery Assessment

Supply-chain assessment introduces artifacts that are out-of-compliance, or intentionally incompatible, like the case of unsigned packages, modified provenance metadata, malicious dependencies, using non-original images and configuration drift in place. The platform shall reject practically provenance artifacts which cannot meet provenance, integrity, or admission requirements before being admitted to privileged execution regions. Checking the accuracy of approved artifact against matching source, build and deployment metadata across the lifecycle is also checked by verification.

Recovery tests are focused on identity revocation and credential life cycle management. Workload identity is revoked, related sessions, tokens and that are on cache are alive. The platform is then tested with its predetermined expiration and revocation policies. The aim here is to ensure that the fact that revoked workloads cannot continue to exercise the former granted authority following a validated transition is checked.

3.4. Network Intent and Execution Safety

Network safety testing is the test of the overall direction of the desired motive to the operation of external infrastructure. Tests inject syntactically sound and perhaps malicious intents to know whether policy and scope approval could be in a position to isolate acceptable syntax and satisfactory operations. Additional kinds of scenarios change an execution plan when approved or in attempting to re- execute an older work item.

The site has to detect mismatches in generation, signature failures, expired authorization and scopes not in target before execution. The controls ensure that there is no chance of transferring one plan to a different or different operation. There is also idempotency and expiration to make sure that re-playing the external effects will not produce a duo of results.

3.5. Audit Reconstruction and Evidence Completeness

The last test is in determining whether any security-relevant operations are reconstructible based on tamper-evident evidence. The initiating caller, policy, authorized intent, new plan, performing worker, use of credential or use of ability, external operation identifier and final verification status should be built with a full rebuilding.

Measuring the completeness of the audit within the framework of performing purposefully representative works streams and coming to a conclusion whether every critical transition makes sufficient evidence without exposing to some sort of secrets that are safe or not. The evidence chain developed should be supporting incident investigation, operational accountability, and compliance analysis and ensuring that security policies of the platform were adhered to, as mandated in the implementation.

4. Resilient Trust Enforcement and Operational Continuity

Enterprise network security architecture must offer both a rigorous level of control and persistence and maintenance. Other form of operational risk is a security mechanism where the authorized operators are omitted because they are not authorized to respond to a network incident even though the mechanism is theoretically correct. Instead, it is to drive back authorization in extraordinary circumstances, but rather put in place and implement security controls that remain functional and mitigable in the event of service, communications routes or policies becoming unavailable.

When the authorization decisions are conditional to multiple distributed services, then it aggravates a lot of problems when they are synchronized. In the event of a major event, a failure of an identity provider, or a policy service or service mesh or an event broker (or any other dependency) can render unsanctioned execution of an application infeasible, even though the underlying network might be available. The specified architecture can respond to the given situation by assigning a strictly dedicated caching of previously analyzed authorization decisions. The entries in the cache should expire with a period, resource and action scope, as well as clearly be attributed to the identity and version of policy under which the decision was made. Caching should be limited in continuity thus, and not turn into a form of unrestricted permission.

Supplemental control is emergency procedures. Emergencies might be defined in the eyes of some individuals or services, which will require an emergency capability by design. Clear scope, duration and resource to be targeted and operational use must be approved should limit the scope of such procedures. An excellent grant should include an immediate and non-editable audit trail furnishing a name, use, context of authorization, resources that are affected and whether they are outstanding or not. Then it may be checked on the occasion of incidents to determine the relevance of the application of the emergency mechanism and to determine whether policy changes are necessary or not to be made after that.

It is noteworthy that no effort should be made to realize operational availability without performing identity verification, authorization or audit processes. These backups can result in temporary outage of service to get out of control as a security breach and difficult to rebuild an occurrence in future. Instead, redundancy, tightly confined decision caching, pre-authorized emergency workflow, local implementation of enforcement, and graceful degradation should result in resiliency.

The same is applicable to audit services. Through the possibilities of centralized audit storage inaccessibility in case of its momentary indispensability, the activities over the sensitive of security could not be silently continued without leaving any traces. The platform can use secure locals buffering or queuing so as to hold the security incidents until it can be drawn to a central location, with a capacity as well as retention reconnaissance.

Safe orchestration thus consists of a strong type of trust enforcement in contrast to unrestrained fail-open behavior. The aim is to ensure that there is the possibility of the continuance of critical operations to take place under controlled exceptional circumstances without affecting identity, authorization limits, accountability as well as evidence. This approach entails looking at availability and security as complementary architecture properties and not contrary goals.

5. Conclusion

To achieve secure enterprise network orchestration, the architectural model is needed whereby trust is assigned to each interaction explicitly and not due to the location in the network or deployment environment or due to membership of the component. This need will be taken into consideration in the proposed architecture, which will isolate the orchestration platform into particular areas of interaction, core control, integration, execution, and data. This segregation offers certain boundaries on trust and limits the infection of compromise between the elements that play diverse roles in activities.

Contextual authorization and facilities Workload identity provide the foundation to management of service-to-service passages and human entrance. It is possible to restrict any operation on its authorized strictly defined capability by evaluating the principal, action, resource, scope, purpose and conditions in the context of the situation through the platform. Assured execution further assures that it cannot be arbitrarily manipulated, rewound or re-executed approved work items as desired. Transient credentials and revoked secret access contain the amount of disclosed privileged access to infrastructure and software provenance mechanisms increase the security of the artifacts that have been deployed to the sensitive execution environment to breach or denial of service.

Security is also shown by the architecture to not mean doing away with extensibility. Controlled device adapters, versioned APIs, and event-based interfaces provide handy forms of interoperation with external services and technologies, which are heterogeneous networks and leave internal databases and full access to control-plane these capabilities hidden. The extensions can be implemented on a distinct workload identity and quota, communication boundary, and capability scope. Would-be extensions, which are unavoidable, can as well be isolated by sandboxing and resource constraints.

Realization of security and operation responsibility is an additional value added. The intent validation, plan verification, execution controls and tamper-evident audit evidence are a tracked connection of a request by an authorized request and the ensuing infrastructure response. This argument aids in the investigation of security, the

examination of conformity, and verifying accomplishment without relying on implicit trust.

Overall, the suggested architecture provides the foundations of credible extensibility grounded on the capability-limited trust. Its principle commitment is that it is a rule that every extension is granted only as much power, knowledge and means as required to accomplish the stated purpose. The development of enterprise orchestration platforms can be achieved without impairing the control-plane integrity, accountability, and operational resilience through separation of concerns, explicit identity, contextual policy enforcement, supported execution, provenance validation, and regulated integration surface.

References

1. S. Rose et al., *Zero Trust Architecture*, NIST SP 800-207, Aug. 2020. https://doi.org/10.6028/NIST.SP.800-207
2. R. Chandramouli and Z. Butcher, *A Zero Trust Architecture Model for Access Control in Cloud Native Applications in Multi-Cloud Environments*, NIST SP 800-207A, Sept. 2023. https://doi.org/10.6028/NIST.SP.800-207A
3. SPIFFE Project, *Secure Production Identity Framework for Everyone Specification*. https://github.com/spiffe/spiffe/blob/main/standards/SPIFFE.md
4. SLSA Project, *Supply-chain Levels for Software Artifacts Specification*. https://slsa.dev/
5. Cisco Systems, *Introduction to Cisco Catalyst Center API*. https://developer.cisco.com/docs/catalyst-center/
6. R. Chandramouli, *Security Strategies for Microservices-Based Application Systems*, NIST SP 800-204, Aug. 2019. https://doi.org/10.6028/NIST.SP.800-204
7. S. S. Mwanje, "Intent-driven network and service management: Definitions, modeling and implementation," *ITU Journal of Future and Evolving Technologies*, vol. 3, no. 3, pp. 555–569, Oct. 2022, doi: 10.52953/AOIA2456.
8. A. Clemm, M. F. Zhani, and R. Boutaba, "Network management 2030: Operations and control of Network 2030 services," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 721–750, 2020, doi: 10.1007/s10922-020-09517-0.
9. L. Pang, "A survey on intent-driven networks," *IEEE Access*, vol. 8, pp. 22,862–22,873, 2020, doi: 10.1109/ACCESS.2020.2969208.
10. H. Yang, "Automatic guarantee scheme for intent-driven network slicing and reconfiguration," *Journal of Network and Computer Applications*, vol. 190, Art.no. 103163, Sept. 2021, doi: 10.1016/j.jnca.2021.103163.
11. F. Bannour, S. Dumbrava, and D. Lu, "A flexible GraphQL northbound API for intent-based SDN applications," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Apr. 2022, doi: 10.1109/NOMS54207.2022.9789785.