# NON-DETERMINISTIC AND RISK BASED SECURITY SERVICES

Srinivas Rao Doddi[1] and Akshay Krishna Kotamraju[2]

[1]Department of Information Technology, University of Los Angeles, Los Angeles, California, USA
[2]Founder Non-profit, Think Cosmos, Saratoga, California, USA

## *ABSTRACT*

*This paper presents a comprehensive framework to address these challenges. Understanding various social engineering tactics is crucial for effective prevention and detection. Trust based models in entities enable many business objectives that may include speed to market, scalability, decentralization etc However, they also increase the attack surface due to "loose boundaries" between enforcement points or corresponding resource authorization servers and the service orchestration layer. The control points of enforcement are generally static across a spectrum of threat vectors such as Identity, Fraud, Authentication, Authorization, Cyber security and physical security etc. in this paper we propose a "Trust based security framework aka " Interdiction Services" that fundamentally is non-deterministic and risk based. This paper presents a converged security framework towards a comprehensive prevention and detection controls mechanism. The paper proposes a converged security framework that allows various parties from fraud, cyber, and physical security to collaborate but operate independently through a common framework of Interdiction Services*

## *KEYWORDS*

*Security, Access, Authentication, Authorization, Assurance*

## 1. INTRODUCTION

Zero trust security models have dependencies with various layers and the policies of enforcement between the service layers could be specific to each domain layer, experience layer, data layer etc and therefore lead to security lapses. Challenges at different layers-The different layers of zero trust security spanning domains such as workforce, device management, network, telemetry, data etc have unique products specific to each layer. Any deviation from those boundaries in terms of implementing those products would be constrained by the lack of orchestration and telemetry with adequate policy oversight and enforcement controls. In addition, enabling web services as atomic as possible are a key aspect to consider. Many architectures supporting web services generally decompose web applications into modular components to automate deployments and achieve scalability. However, the flipside of these benefits is that these could lead to increase attack surfaces, complexity in security policy enforcement. Configuration issues with legacy tools etc. It is also important to recognize that due to policy fragmentation, tools etc. the assets aren't fully managed in a Zero Trust construct. [1] These in turn result into security gaps from poor planning and have the potential to cause disruption. Some of the threats or challenges include but not limited to Insider threats challenges wherein malicious actors with privileges can exploit resources or weakness in policies or processes can result into a security risk. It is an important point to highlight that the applications and the associated services are prone to "malicious attacks "such as client-based attacks that may include "Request smuggling" and in few instances they could be subject to "Token misconfigurations". Additionally, Microservice-based attacks exploit vulnerabilities in cases when a component or a module become compromised and as a result could disrupt other modules or associated services of the application .[1] All these make token introspections and trust challenges even more important and therefore a valid assertion is required that may be recursive. These assertions include identity, authentication, authorization etc to check for threat vectors such as compromised credentials. Authorization and trusted identity challenges include Trust broker services that connect applications and users are vulnerable to failure and are generally targets for attack. Another aspect to keep track of or monitor would be local physical devices that can be attacked ,have data exfiltration through associated services.[2] Zero-trust admin account credentials are attractive targets as well. These while have their own benefits on the flip side lead to increase attack surfaces, large number of components leads to complex security policy enforcement, Configuration issues with legacy tools etc. Therefore, it is important to plug any policy leaks or gaps to ensure that all assets are managed in a “zero-trust” construct [3]. These in turn result into security gaps from poor planning and have the potential to cause disruption.

## 2. LAYERED SECURITY FRAME WORK TO MITIGATE SECURITY RISKS

We recommend a layered security approach to enhance resource authorization limitations or gaps by ensuring that the framework allows for context with path-awareness policies . This methodology allows to recursively enable AuthN/Authentication services or enforcements via sidecars.[4] The layered security framework wea re proposing also allows applications to rely on authorizations that the user accesses the relevant application [5]. We propose the OAuth2 framework for authorization at the endpoints that are enabled via proxies.[6] The applications could be additionally considered or designed as atomic micro-services with corresponding endpoints. This allows for control enforcement of protection of relevant endpoints and would be repetitive or continuous checks via relevant polices. Also, these services can then be made a generic pattern by enabling client based OAuth2 libraries within each of the micro services so that these can now be deemed protected endpoints. We also recommend maintaining versioning across microservices and that also allows for different language models that may have different libraries The framework also has a sidecar pattern that helps address or alleviate some of the above stated limitations, concerns and in addition provides consistency across all these micro services while at the same time protect the relevant endpoints with OAuth2 workflow. The pattern also demonstrates and allows configuring the sidecar thereby generating, validating relevant token int eh format of a JSON web token aka JWT . This allows for low code changes for relevant micro service and in many instances a no code change to maintain and the corresponding microservices.[6] In a sidecar pattern, it provides an opportunity to offload the authorization and authentication with the sidecar. The proposed pattern of the sidecar envoy allows the incoming connections to flow through the application after the successful checks related to both authorization and authentication. In a multi-tenant or a cloud native ecosystem the sidecar pattern enables functionalities associated with the corresponding domain or application could be defined under a separate process to provide encapsulation and make them atomic but local at the same time thus preserving the integrity aspects.[6] The assumption being that the sidecar and the application are residing in a trusted encapsulated environment.

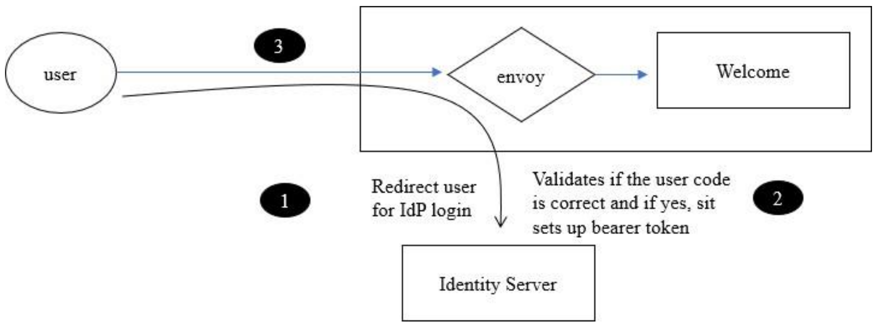

user
3
envoy
Welcome
1
Redirect user for IdP login
Validates if the user code is correct and if yes, sit sets up bearer token
2
Identity Server

Figure 1. A generic sidecar pattern for IAM

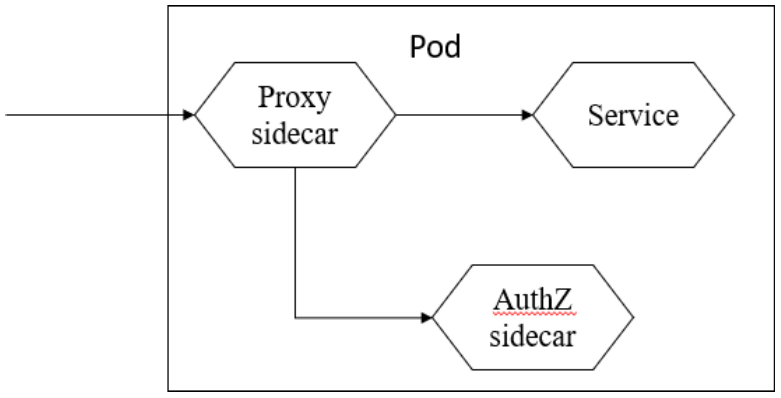


Figure 2. A generic sidecar proxy pattern for IAM

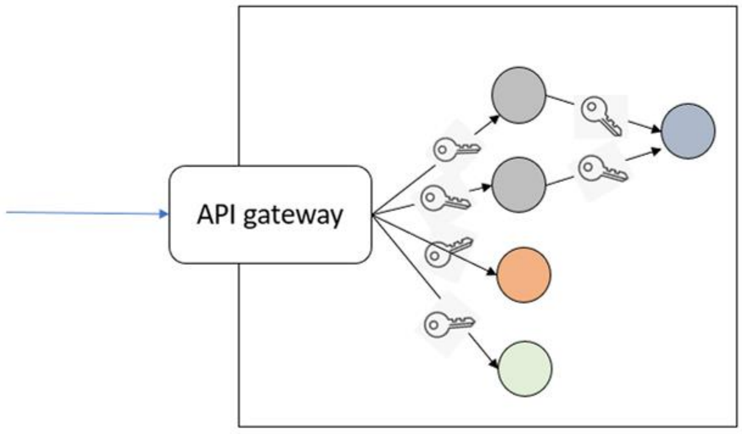


Figure 3. JWT in their incoming request and outgoing requests.

## 2.1. Microservices and Trusted Execution Challenges

Our pattern also allows for various micro-services to be able to interact based or defined and managed trust . Each such layer across the communication channels in turn are required to introspect policy configurations when required. [6] The predictable nature of the risk services create additional risk if the perimeter or firewall configurations meet adequate threshold for them to be trusted. This creates scenarios wherein the derived trust may not be sufficient and creates ambiguity in the context of zero trust for applications. The scope of the service or application being evaluated are often provided with identity and policies are built over this identity to ascertain the connection. This requires both the layers below the service and higher layers to be able to attest the boundary conditions. We leverage OAuth2 in helping achieve zero trust, thus, conforms to best practice. Our framework has patterns that systemically can make an endpoint attestation based on signatures identified. To achieve this a sidecar ensures that the applications can communicate only via the sidecar, and all the endpoints are automatically secured[6] Our framework also enables uniformity leading to clear visibility into telemetry and observability aspects leveraging service mesh. It also allows for configuring the sidecar and not write any relevant code thus maintaining the integrity. The application in such a scenario needs to enable the relevant business logic, and additional peripheral concerns could be migrated to the sidecar. The side car can be plug-gable and it can also provide support for multiple application languages ,frameworks. The framework we propose also allows for native OAuth2 integrations via Envoy and applications use external authorization framework that are in turn based on relevant OPA (Open Policy Agent) so that appropriate authorizations can be enforcement or allowed. [6]

## 2.2. Oauth Grants and Authorized Access Challenges

We also consider an event or a scenario wherein unauthenticated/unauthorized access is intercepted via the sidecar, and user gets redirected to relevant identity service/IdP that in turn forces an authentication challenge. The outcome of the event if results in a “successful authentication” would then be vi aa browser redirect with an implicit OIDC endpoint callback again gets scanned by the sidecar. This process via sidecars facilitates obtaining corresponding identity tokens and associated services redirect agents to requested page the applications that could be web based or app based. The layered security construct also forces a refreshed token thru the sidecar whenever a token expiration or validation is in question and this process may be repetitive. This allows for a bearer of the token to not only authenticate successfully but also access an end point that is in scope of authorization. It also can allow for customizations with further assertions of the JWT. An example of the JWT would now be a layer that could entitlements based such as an RBAC or ABAC based JWT filter on the envoy, and this in turn allows for passing of the decrypted JWT to the back-end application, for additional introspection if the application requires as governed by local policies.[6]

## 2.3. List Based Access Control and Privileges

The microservice service should be made context aware so it applies relevant authorizations specific to it. The list-based policy or LBAC in our framework needs to be at service level, since the context requires service’s private data. However, this can also be centralized through a generic pattern wherein either an RBAC( role-based access) or ABAC (attribute-based authorization). This way all such services can be intercepted, introspected and decisions can be enforced on what could be authorized or denied and this can in turn be achieved vi a local policy engine or could be federated globally.[7]

## 2.4. Zero Trust Simulations and Authorizations

Before putting zero-trust implementations into production, we recommend user trials and security evaluations. It allows the users in enabling an experience by deploying these types of systems, tools and manage security teams experience and strengthen a security operation centre aka SOC responding to incidents. [8] This methodology also allows to gather feedback from all users to improve future implementations. People as attack surface extension & their risk mitigation and zero trust needs to be a "collective responsibility" within an organization. Multiple degrees of controls need to be enabled through a policy-based approach that covers data, network, application etc domains. The scaling of these controls has to be a gradual increase so that these don't disrupt business continuity, the lack of which could again create siloed approach in tackling "trust' between. Benefits – wherein a non-decentralized policy management allows policy or rules to relevant databases that is restricted. This would be similar to Principle or least privilege or single responsibility as it relates to authorization. Additionally, some of the challenges related to LBAC involves addressing data checks on services that are upstream. The framework allows for the authorization service to be context aware and apply appropriate authorization rules and this includes both upstream and time with domain and experience layer criteria [7].

## 2.5. Authorization at the Microservice Level

Each service is in charge of enforcing all its own authorization rules. It also allows for Role Based Access Control (RBAC), based services to assess authorization for relevant requests, with appropriate roles information. And the approach we suggest wis to fetch the authorization Service and also validate that with suer roles with each request [7]

## 2.6. Token Based Authentication

Traditional approaches authenticate users at start of the request but don't account for complex topology and the dynamic nature of microservices. Therefore, it is important that clients use an authentication service to acquire an access token in line with their privilege es. Any additional requests must include this token and to be introspected by services for subsequent authorization,

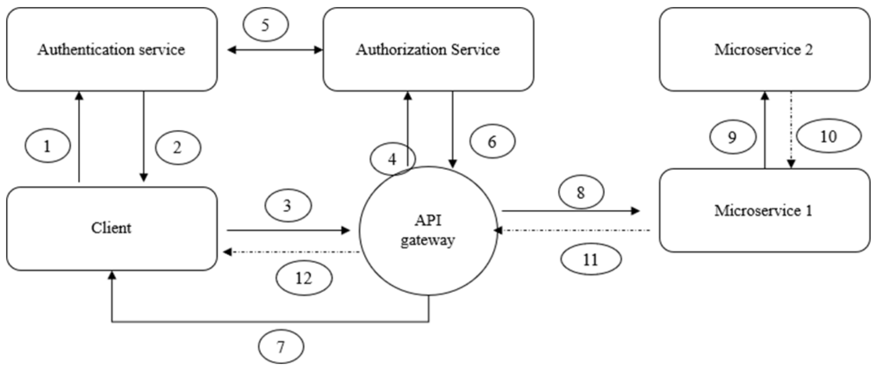


Figure 4. Client uses an AuthN service to acquire an access token baser on privilege.

## 2.7. Proposed Design Patterns of Interdiction Services Framework

We recommend leveraging sidecars so it restricts with reduced attack surface the trusted computing base. Since microservices are prone to attacks a component may be compromised which in turn has the ability to inflict damage and cause disruption and that may include applications. To address this issue, we propose a recursive authentication and authorization mechanism with each and that our sidecar pattern for a smooth execution and maintaining integrity of each service providing security in a matrix deployment at large scale .[7] In addition to this, embedding path information tokens within requests allow sidecars to validate whether incoming or outgoing requests meet certain security policies. Standardize authorizations and create holistic view of the security information model, enable authz decisions closer to enforcement points, Scalable distributed environment, Managed and composable services, zero code authz model that any can use anywhere enforcement, Extensible to self-serve complex policy logic etc [7]

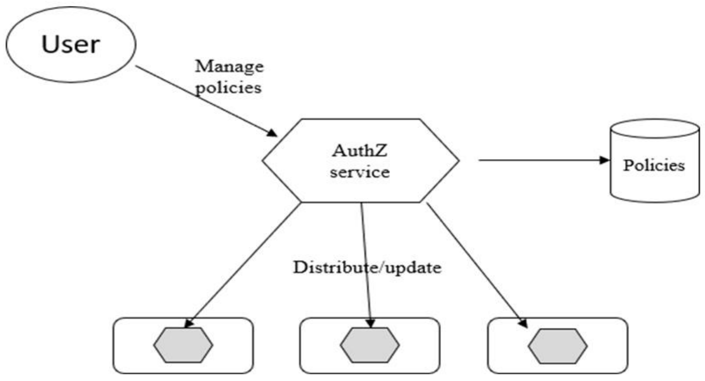


Figure 5. Pods with authorization sidecars

### 2.8. Microservices Attack Vector  Trusted

Client based attacks could be of request smuggling or token misconfiguration type. In addition, microservice based attacks could be malicious as well.

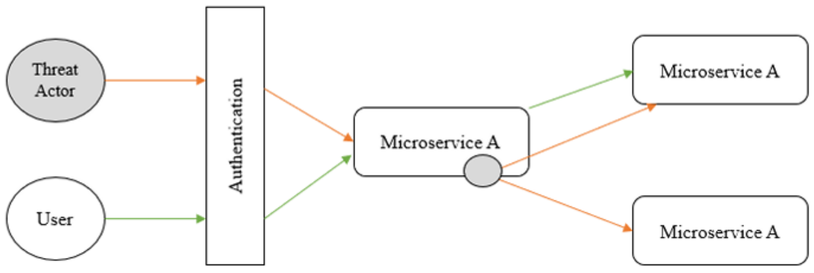


Figure 6. Microservices vulnerable to attacks where one component becomes compromised.

### 2.9. Managing Polices and their Enforcement

Rules/policies may be managed in a central service, but then distributed to every authorization sidecar. open-source policy engine called OPA. We also recommend writing or coding policies through either JSON or Rego format, and this allows a distributed package throughout infrastructure. These policies in turn need to be polling in real time so the information/policy refresh is latest and that each of these are made locally available to corresponding pods. This allows of scalability, high availability, security and ensure s a zero-trust ecosystem. We also further recommend configuring the sidecar /Envoy as a proxy which then can introspect the relevant JWTs for authentication . Finally, all such configurations can be encoded with open policy agent aka OPA so that relevant authorizations can be enforced and applied at each container. [7][15]

### 2.10. Benefits of Path Aware Security

Our approach of utilizing hotlists that are aware of the paths allow to detect unusual requests and would detect compromised services.[7] Provide path-aware security for microservices. Perform recursive checks through sidecars using client and path information. Such a sidecar can then be used to apply RBAC for each of the microservices. Sidecar may have access to the authorization rules/policies. This way each pod checks via proxies the status of relevant authorizations requests

,ensures polices are applied, introspects and then forwards to corresponding resources access. [7][18]This ensures that the microservices are exempted from implementing RBAC. Authorization is enforced at the microservice level instead of at the edge, complying with the Zero Trust Architecture and making the corresponding pod more independent.

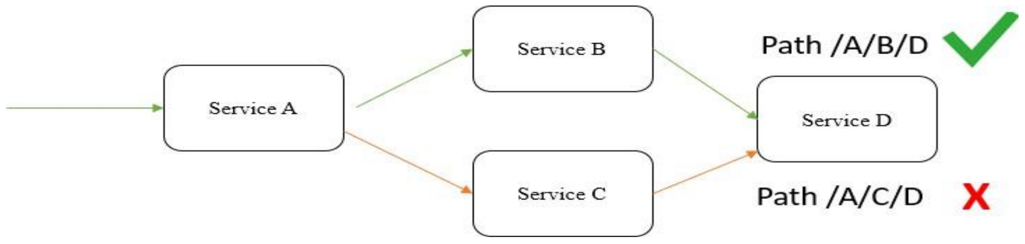


Figure 7. Path aware lists detect abnormal flows

## 2.11. Proposed Design of Interdiction Services Framework

We propose to deploy sidecars to reduce trusted computing base. Ingress proxy should have misconfiguration checks along with path inspections. Egress proxies to block outgoing requests to unauthorized services. Also, they need to append signed sidecar token to request header path fields. Such that sidecar A || sidecar B[8]

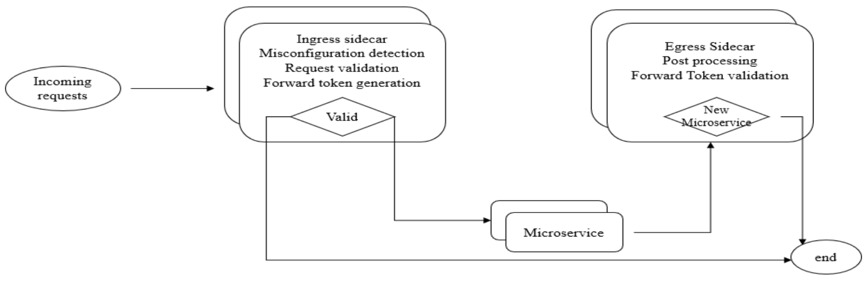


Figure 8. Ingress proxy and egress proxy to reduce trusted computing base

## 2.12. Handling Request Flows

Authorization request to have 2 tokens signed by Authorization and gateway servers.

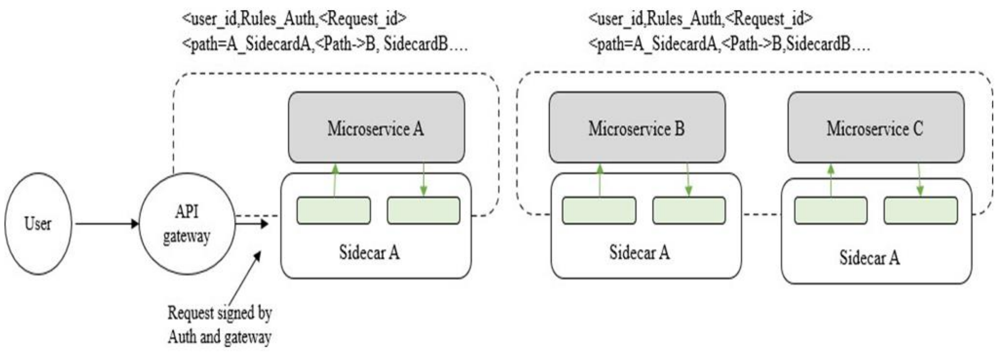
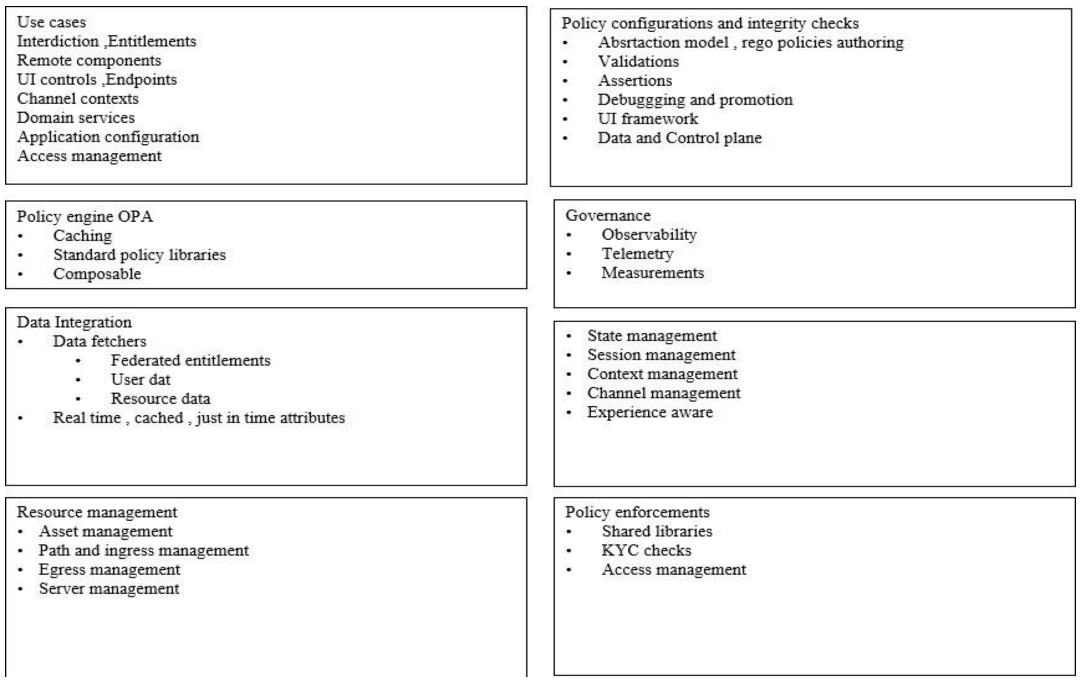

<user_id,Rules_Auth,<Request_id>
<path=A_SidecardA,<Path->B, SidecardB....
<user_id,Rules_Auth,<Request_id>
<path=A_SidecardA,<Path->B,SidecardB....
Microservice A
Microservice B
Microservice C
User
API gateway
Sidecar A
Sidecar A
Sidecar A
Request signed by Auth and gateway

Figure 9. Ingress proxy and egress proxy to reduce trusted computing base

## 2.13. Target State Components



Use cases
Interdiction ,Entitlements
Remote components
UI controls ,Endpoints
Channel contexts
Domain services
Application configuration
Access management

Policy configurations and integrity checks
- Absrtaction model , rego policies authoring
- Validations
- Assertions
- Debuggging and promotion
- UI framework
- Data and Control plane

Policy engine OPA
- Caching
- Standard policy libraries
- Composable

Governance
- Observability
- Telemetry
- Measurements

Data Integration
- Data fetchers
  - Federated entitlements
  - User dat
  - Resource data
- Real time , cached , just in time attributes

- State management
- Session management
- Context management
- Channel management
- Experience aware

Resource management
- Asset management
- Path and ingress management
- Egress management
- Server management

Policy enforcements
- Shared libraries
- KYC checks
- Access management

Figure 10. Target state design components

## 2.14. Design Flows– High Level Modular Framework

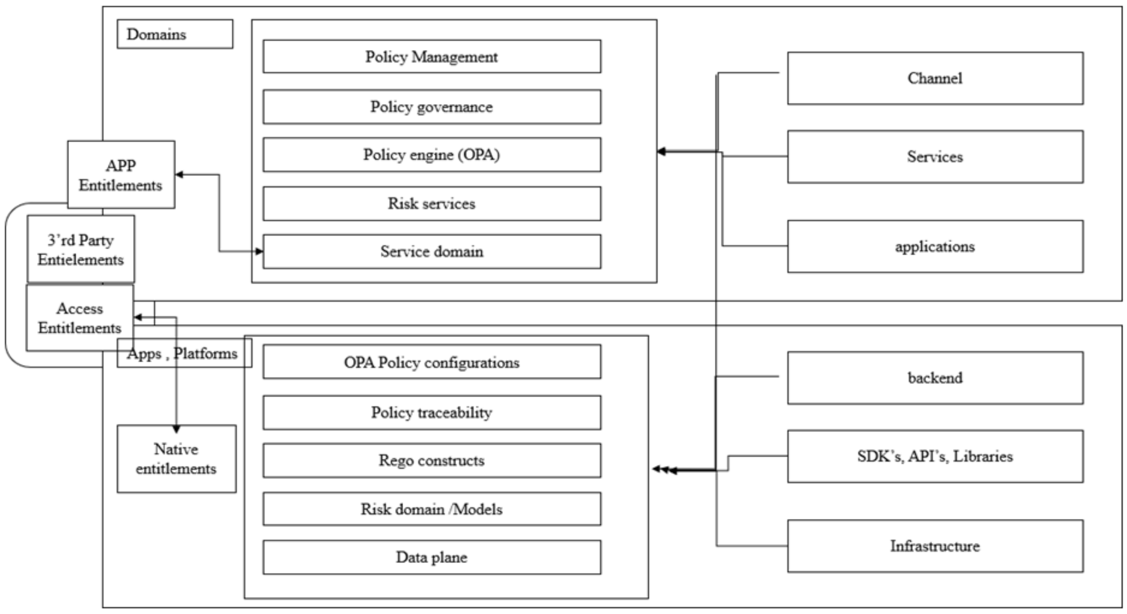


Figure 11. AuthZ modular view

### 2.15. Policy Management and Delineation Between Domains and Platform Services

The below figure namely Fig 12 ,shows the various aspects being considered when a sidecar proxy is enabled for policy evaluation run time.

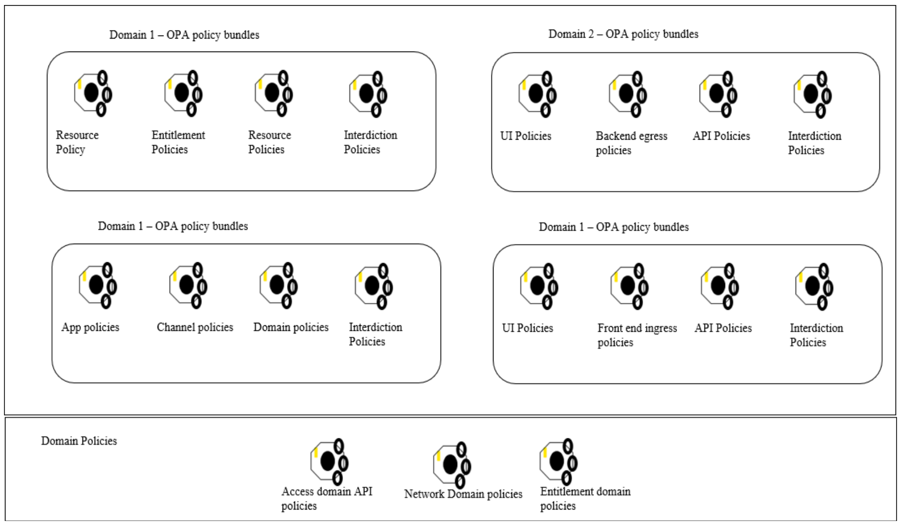


Figure 12. Policy management via sidecar and OPA

### 2.16. User Session and Authorization Interactions with Context

The channels would produce context and the below Fig 13 shows the session and behaviour aspects from channel as important factors in ultimately deciding authoring request and enforcing corresponding polices.[9]

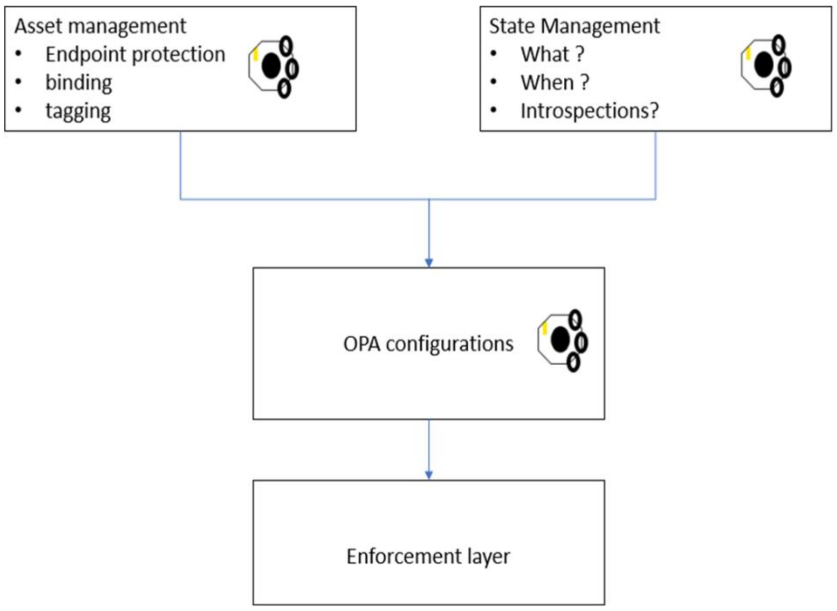
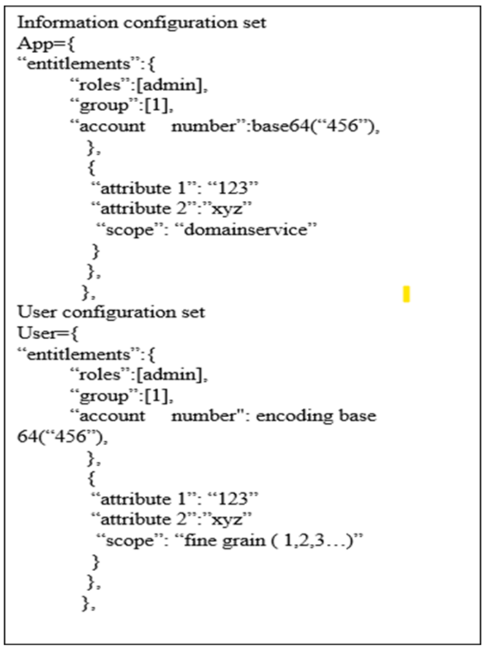


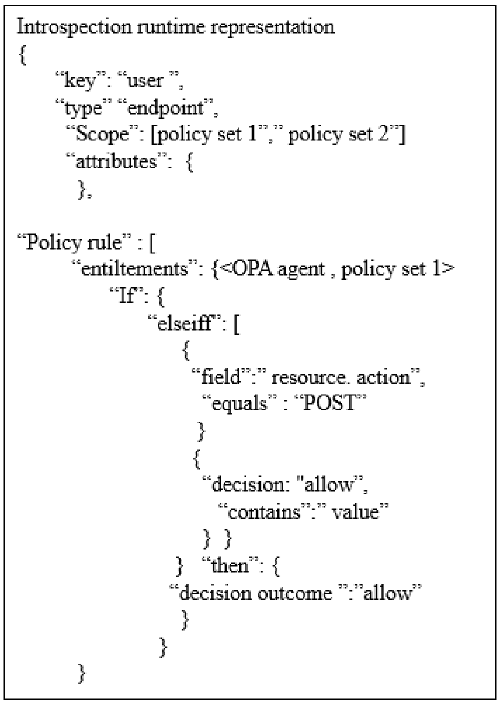
Figure13.AuthZ flow

## 2.17. Sample Illustrative View of Declarative Framework[10]



```
Information configuration set
App={
"entitlements":{
        "roles":[admin],
        "group":[1],
        "account    number":base64("456"),
          },
          {
           "attribute 1": "123"
           "attribute 2":"xyz"
            "scope": "domainservice"
           }
          },
         },
User configuration set
User={
"entitlements":{
        "roles":[admin],
        "group":[1],
        "account    number": encoding base
64("456"),
          },
          {
           "attribute 1": "123"
           "attribute 2":"xyz"
            "scope": "fine grain ( 1,2,3…)"
           }
          },
         },
```

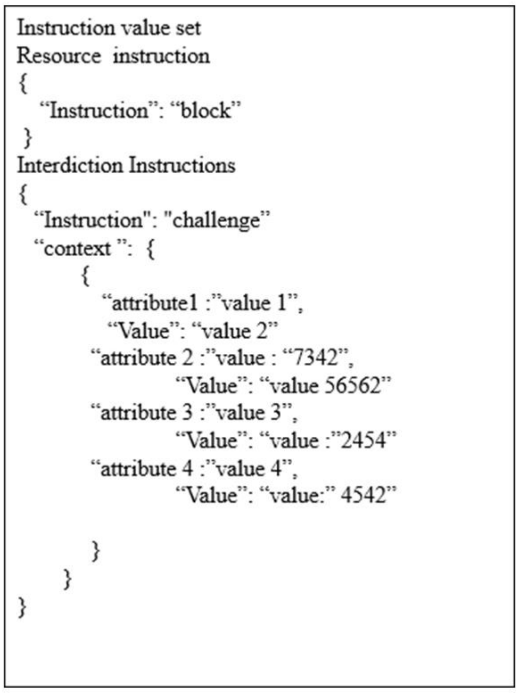
Figure 14. user, resource and policy declarations



```
Introspection runtime representation
{
     "key": "user ",
     "type" "endpoint",
      "Scope": [policy set 1"," policy set 2"]
      "attributes":  {
        },

"Policy rule" : [
        "entiltements": {<OPA agent , policy set 1>
             "If": {
                  "elseiff": [
                       {
                        "field":" resource. action",
                         "equals" : "POST"
                         }
                        {
                         "decision: "allow",
                            "contains":" value"
                         } }
                     }   "then": {
                    "decision outcome ":"allow"
                       }
                    }
        }
```

Figure 15. user, resource and policy declarations



```
Instruction value set
Resource  instruction
{
   "Instruction": "block"
 }
Interdiction Instructions
{
  "Instruction": "challenge"
  "context ": {
         {
            "attribute1 :"value 1",
             "Value": "value 2"
         "attribute 2 :"value : "7342",
                    "Value": "value 56562"
         "attribute 3 :"value 3",
                    "Value": "value :"2454"
         "attribute 4 :"value 4",
                    "Value": "value:" 4542"

            }
        }
}
```

Figure 16 user, resource and policy declarations

## 2.18. Authorization Flow

The Fig 15 shows the dependencies and orchestration needed to keep things in sync from a request response to policy enforcement and also for enrichment of services.

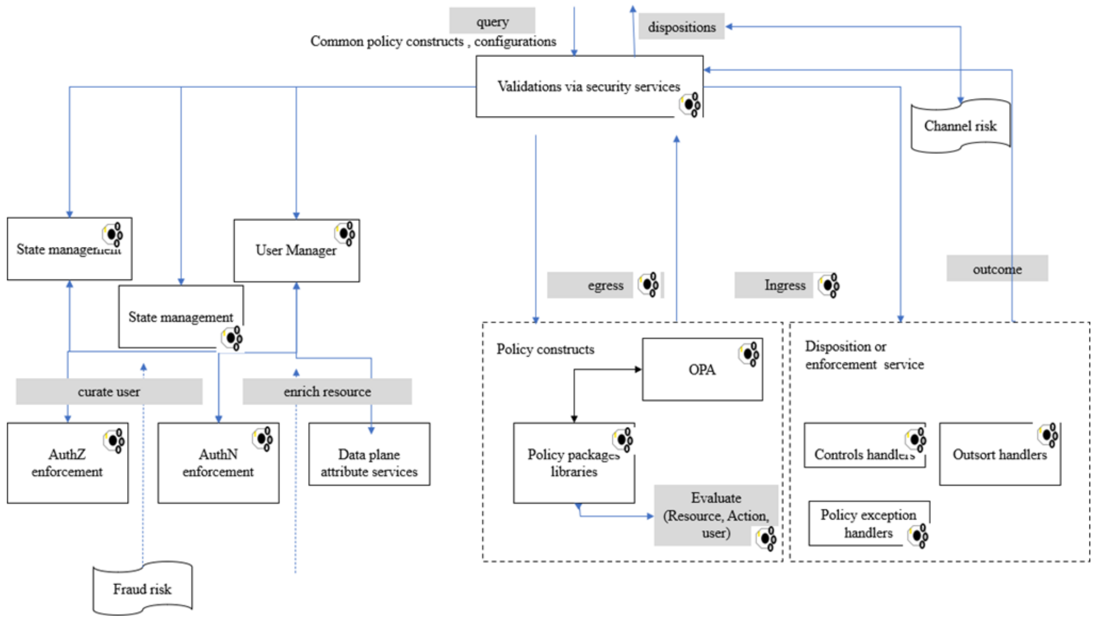

query
Common policy constructs , configurations
dispositions
Validations via security services
Channel risk
State management
User Manager
State management
egress
Ingress
outcome
curate user
enrich resource
Policy constructs
OPA
Disposition or enforcement service
AuthZ enforcement
AuthN enforcement
Data plane attribute services
Policy packages libraries
Controls handlers
Outsort handlers
Evaluate (Resource, Action, user)
Policy exception handlers
Fraud risk

Figure 17. AuthZ request /response flow

## 2.19. Policy Evaluation Context

The evaluation of policy needs to go further beyond just being a set of static rules as the context not only complements the risk signals in aggregating the scores but also allows for proper normalization to avoid false positives.

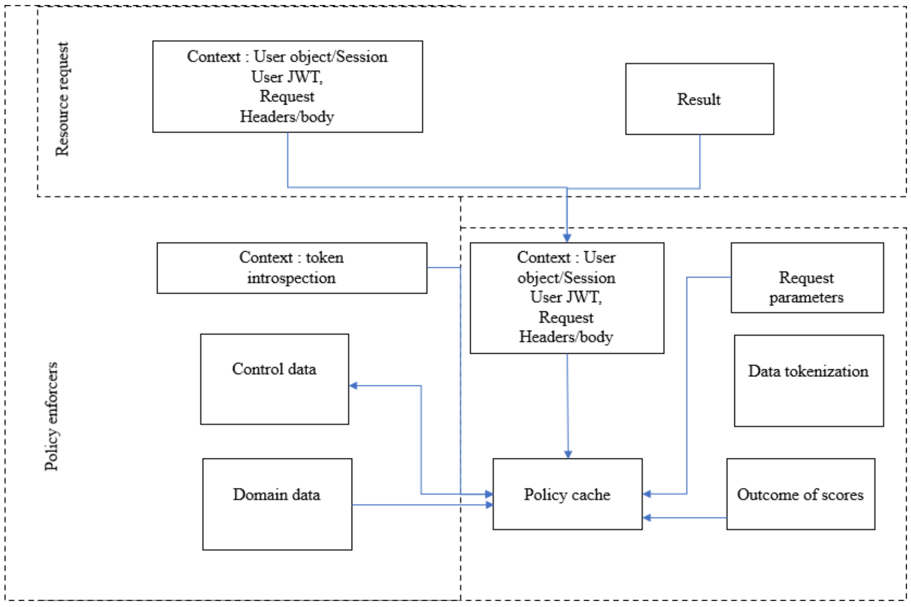

Resource request
Context : User object/Session User JWT, Request Headers/body
Result
Policy enforcers
Context : token introspection
Context : User object/Session User JWT, Request Headers/body
Request parameters
Control data
Data tokenization
Domain data
Policy cache
Outcome of scores

Table 18. context aware policy evaluation

## 2.20. Overall Authorization Framework

The below Fig 17 shows the overall authorization framework and also highlights the delineation between the platform services and application domains.

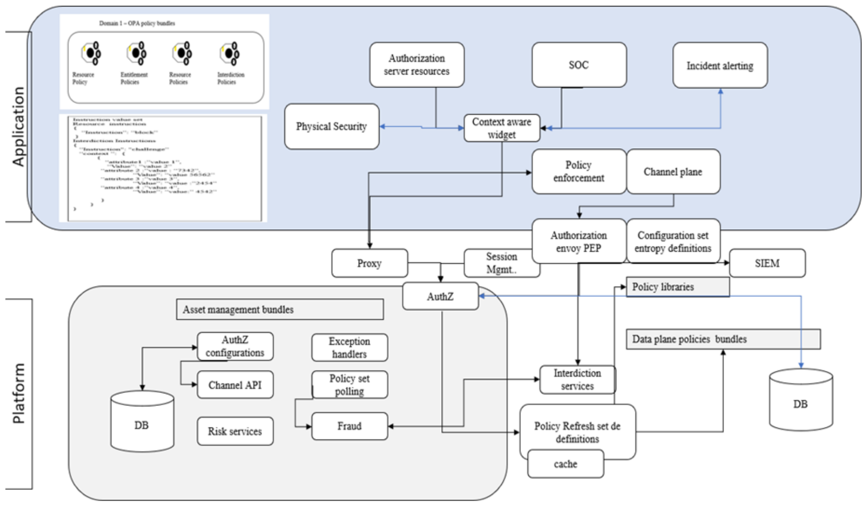


Figure 19.Authorization framework

### 2.21. Risk Based Interdiction Services in Decentralized Trusted Framework

The channels would emit events and they are being introspected in real time by respective OPA agents without the need to invoke any API calls. The risk is then calculated via OPA rules run time and a policy enforcement happens within the channel and with the context , which is key to understand the user behaviour.[9] Each of the key decision engine along with its system of record would act independently and this allows the orchestration layer to handle with clear demarcation between ingress and egress layers. The framework allows for policy collision and avoid conflicts via a hierarchical rule set definition[10][11[12] This can further be strengthened by the infrastructure deployments of red/Green and North/South traffic patterns that are configured for gateways and service mesh appropriately.

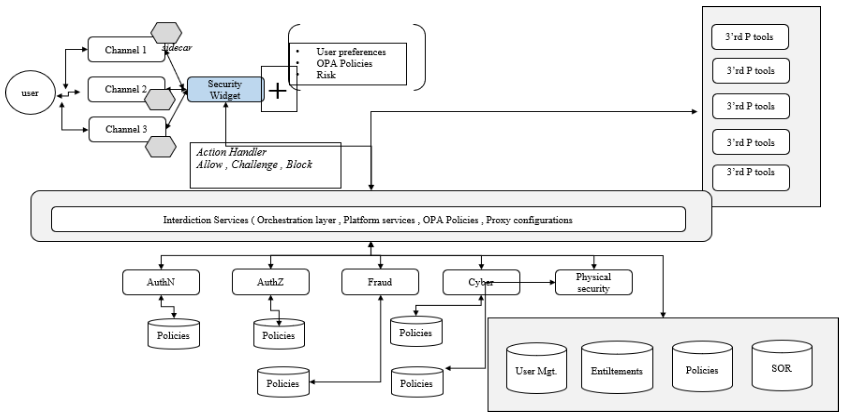


Figure 20. Channel and platform trust and decentralized risk-based services

## 3. CONCLUSIONS

We have presented many patterns with an end-end perspective that includes channels, platforms and Application planes when it comes to trusted access management and risk services. Our patterns have shown to asses channel and platform risk in real time with decentralized approach via sidecars. A comprehensive frame work of interdiction services as presented allows to asses risk, manage user experience, and also allows to handle threats as a result of conflict or malicious activity. The authorization patterns laid out allow to not only detect any suspicious or abnormal activity, they also prevent, deter, detect and allow to threats appropriately.

## ACKNOWLEDGEMENTS

The authors would like to thank everyone, just everyone!

## REFERENCES

[1] https://www.sacmat.org/2023/resource/slides/3_1_1_MeadowsCatherine.pdf
[2] https://www.sacmat.org/2023/resource/slides/3_1_1_MeadowsCatherine.pdf
[3] https://nordlayer.com/learn/zero-trust/benefits/
[4] https://www.sacmat.org/2023/resource/slides/3_1_1_MeadowsCatherine.pdf
[5] https://www.sacmat.org/2023/resource/slides/3_1_1_MeadowsCatherine.pdf
[6] https://www.jpmorgan.com/technology/technology-blog/protecting-web-applications-via-envoy-oauth2-filter
[7] https://www.softensity.com/blog/authentication-authorization-in-a-microservices-architecture- part-3/
[8] https://www.techtarget.com/searchsecurity/tip/Top-risks-of-deploying-zero-trust-cybersecurity-model

[9] http://archive.uva-aias.net/uploaded_files/publications/WP99-Hollanders,Bersem.pdf

[10] X. Zhu and Y. Badr, “Identity Management Systems for the Internet of Things: A Survey Towards Blockchain Solutions, ”Sensors (Basel, Switzerland), vol.18, no.12, pp.1–18,2018.

[11] R. Taylor, D.Baron, and D.Schmidt, “Theworldin 2025 – Predictions for the next ten years, ” 2015 10th International Microsystems, Packaging, Assembly and Circuits Technology Conference, IMPACT 2015 - Proceedings, pp. 192–195, 2015.

[12] P. Handy, “Introducing Masked Authenticated Messaging, ”2017. [Online]. Available: https://blog.iota.org/introducing-maskedauthenticated-messaging-e55c1822d50e

[13] A. Gruner, A. Muhle, T. Gayvoronskaya, and C. Meinel, “A Quantifiable Trust Model for Blockchain-Based Identity Management, ”in 2018 IEEE International Conference on Internet of Things (iThings), Green Computing and Communications (GreenCom), Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData), no. September. IEEE, 7 2018, pp. 1475–1482. [Online]. Available: https://ieeexplore.ieee.org/document/8726703/

[14] F. Chin-Chen Chang and C.-Y. Lee, "A secure single sign-on mechanism for distributed computer networks", *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, vol.59, no.1, 2012.

[15] B. M. CA Gunter and D Liebovitz, "Experience-based access management: A life-cycle framework for identity and access management systems", IEEE security and privacy, 2011.

[16] J. Jensen, "Federated identity management-we built it; why won't they come?", *IEEE Computer and Reliability Societies*, 2013

[17] Conceptual Foundations for Forming a Configuration Management Subsystem of a Telecommunications Network; A.K. Kanaev; E.V.Login; K.A.Pudovkina; 2024 International Russian Smart Industry Conference (SmartIndustryCon)

[18] User identity and Access Management trends in IT infrastructure – an overview. ManavA. Thakur, Rahul Gaikwad. 2015 International Conference on Pervasive Computing (ICPC)

## AUTHOR

**Srinivas Rao** is a technology professional currently at university of Los Angeles , California.

**Akshay Krishna Kotamraju** is founder of a non profit organization “think cosmos” aimed at helping students learn more about Astronomy& Programming.