CHAPTER 4

Transitioning to Modern Authorisation Management

Alex Olivier
CPO and Co-founder at Cerbos

Navigating the ever-evolving landscape of authorisation management poses considerable challenges for companies. The dynamic nature of authorisation requirements, ranging from the addition of new roles to adapting to shifts in data security needs, demands a delicate and imperative approach. Legacy systems entrenched with hard-coded authorisation logic present hurdles in terms of scalability, developer experience, and responsiveness to changing business demands.

For regulated industries aiming for compliance benchmarks like ISO27001, SOC2, or PCI, the fragmented nature of authorisation audit logs can pose significant challenges during audits or security incidents. In this environment, the inability to efficiently handle complex roles and permissions can impede a company's growth potential.

Transitioning to modern authorisation, or decoupling authorisation management from core code, emerges as a strategic solution to the challenges posed by legacy systems. The paradigm shift involves centralising authorisation logic in audited, version-controlled policies, separating it from the application code. This decoupled authorisation approach brings about technical advantages, such as independent evolution, versioning, and testing of authorisation logic. Standardising decisions through a Policy Decision Point (PDP) not only enhances efficiency but also ensures a consistent audit trail across various components of the application architecture. The organisational impact is equally profound, as this approach fosters transparency, empowering a broader range of stakeholders to comprehend and modify the permissions structure without extensive coding knowledge.

The Challenges of Legacy Authorisation Management

Without fail, your authorisation requirements will change. Whether it is adding a new role, completely reworking your access model to suit a new product packaging framework, or enforcing a change in data security needs—this work is delicate, laborious and imperative to get right. If the thought of this immediately sounds like a three-month scope of work, or touching a ball of mud then it is a sign the authorisation mechanisms in an application need modernisation.

Consider a system where this logic has been deeply ingrained and hard-coded into application logic, this is going to be a time-consuming, expensive and

fragile process. Developers will need to go back into code bases which have gone long untouched, convert business requirements into code, update tests and redeploy the application stack. Changes to the authorisation logic become progressively more difficult to implement, often creating bottlenecks in business operations. For more heterogeneous microservice architectures, the business requirements may have to be translated into many different programming languages and frameworks, each with its idiosyncrasies and testing hoops to jump through.

In regulated industries, or for anyone looking to obtain and retain compliance such as ISO27001, SOC2 or PCI, being on top of the authorisation happening at every step of the request chain is critical. Ensuring that every microservice, function and worker job not only checks whether an action is authorised but also makes a concrete, consistent and verifiable audit log of the actions is a hard requirement. Approaches where each component creates and maintains its own authorisation audit logs, result in a fragmented set of formats, storage, and retrieval mechanisms. When it comes to demonstrating your approach to access control to your auditors or security, you may find yourself searching through disparate application logs and myriad request traces. The pain of trying to understand how actions were allowed is only amplified when being done under pressure due to a breach or data security incident.

Moreover, the inability to effectively manage complex roles and permissions can limit a company's growth potential. It can prevent businesses from selling to larger corporations, hinder operations in regulated industries, and slow down product innovation.

There is a better way to approach this which makes a system manageable, scalable and most importantly compliant.

The Case for Modern Decoupled Authorisation

Decoupled authorisation is a new approach which enables application permissions and access controls to be defined centrally and enforced locally for each application. The core promise of decoupled authorisation is to extract all the authorisation logic out of the application code base and define it in an audited, version-controlled and human-readable set of policies which can be stored as configuration rather than application code.

These policies are loaded into a Policy Decision Point (PDP) which is deployed alongside your application and then wherever permissions need to be checked, rather than hardcoding the logic, an API is made out to the PDP with all the relevant context, and a simple ALLOW or DENY decision is returned. The application code is now a simple boolean check of this response to decide whether to do the requested action or not.

This approach has two major technical advantages:

1. Authorisation logic is now independent of the application code. It can be evolved, versioned and tested in isolation. Whenever a change is made, all the PDPs receive the updated policies and immediately serve decisions based upon them without any need to touch the application code, saving time and deployment risk.

2. All decisions, regardless of where in the application architecture—frontend, backend, async jobs—are going through a standardised PDP service which will make the correct decision based on the policy but also create a standardised audit log of the decision made checking many compliance boxes.

A side effect of decoupling the logic out of the application code is more organisational. This approach allows both business stakeholders and non-expert engineers to understand the permissions structure

as policies are now written as configuration definitions rather than application code. Whilst there are some domain-specific language concepts to learn for any policy format, the effort to learn is far less than understanding the entire application code base structure and architectural decisions. This transparency empowers a broader range of stakeholders to make changes to the authorisation logic without needing extensive coding knowledge or risking the stability of the core application.

Without Cerbos

```
apiVersion: api.cerbos.dev/v1
resourcePolicy:
    resource: contact
    version: default
    rules:
      # admins can do all actions
      - actions: ['*']
        effect: EFFECT_ALLOW
        roles:
          - admin

      # user role can create and update contacts
      # if they are a member of the sales group
      - actions: ['create', 'update']
        effect: EFFECT_ALLOW
        roles:
          - user
        condition:
            match:
                expr: ('sales' in request.principal.attr.groups)

      #user role can delete a contact if they are one of:
      # - the owner of the contact
      # - member of the sales-manager group
      # - member of the product group
      - actions: ['delete']
        effect: EFFECT_ALLOW
        roles:
          - user
        condition:
            match:
                any:
                    of:
                        - expr: request.resource.attr.owner == request.principal.id
                        - expr: ('sales-manager' in request.principal.attr.groups)
                        - expr: ('product' in request. principal.attr.groups)
```

With Cerbos

```
if (await cerbos.isAllowed({ principal: user, resource, action: "edit" })) {
  //allowed
}
```

Addressing the Challenges in Shifting to Modern Authorisation

Decoupling authorisation means relying on a third party, so you need to consider the potential risks that come with it so that you can manage them well.

Firstly, vendor lock-in can feel like a loss of control since you can rely only on the features of the solution you've chosen. The language of your policies and their configuration structure will be imposed by the solution you use. If you find that a feature is missing, you may struggle to achieve your goals, and moving to another solution may be complex. It's therefore of paramount importance to choose your solution well.

That means making sure you select a solution that can support your workload and is highly available. Otherwise, the authorisation component may become a bottleneck or, worse, the single point of failure. For example, you may want to evaluate the scalability of the solution and its ability to handle your traffic load, including periods of increased traffic or sudden spikes in demand. You will need to evaluate outage risks and how to mitigate them.

Lastly, using an authorisation server means adding a call to another application, which means added latency to your application. So make sure the solution you select is performant. If you opt for a SaaS solution, consider the provider's geographic proximity to your application. The better alternative is, of course, to have a solution that can be deployed in your infrastructure.

Decoupling Authorisation: Best Practices

From our experience at Cerbos, 90% of the effort involved with migrating to a more modern authorisation solution isn't a technical one, but rather a business analysis one. Authorisation has stakeholders from across the organisation—be it security teams, customer support, product development, sales and DevOps. Getting all these parties around the table and agreeing on what the access model should be for an application is key.

We always recommend starting with a simple spreadsheet with a column for each user role, and all the actions as rows and simply fill in the boxes for which role can do each action. This quickly highlights the complexity of the permissions as the answer is not always a simple ALLOW or DENY but rather a conditional one.

Once everyone is on the same page, the actual implementation can seem rather trivial, but there are several aspects to consider. It's important to select a solution that meets your application's requirements and can scale with your needs. Consider factors such as ease of integration, performance, flexibility, and support for different access control models. Ideally, the solution should seamlessly integrate with your existing infrastructure, including identity providers, data stores, and messaging systems, to avoid introducing new vulnerabilities or dependencies.

There are various challenges that a development team may face when implementing decoupled authorisation:

Ensuring Policies Are Enforced Accurately and Efficiently

An authorisation system must be robust, accurate, and perform well under load. It's essential to validate policies before deployment and ensure that they are applied consistently across the application. For example, a development team for a large e-commerce platform must ensure that the decoupled authorisation system can handle a high volume of user requests during peak shopping times to protect sensitive user data and prevent unauthorised access to restricted resources such as payment information or order management.

Maintaining Policy Consistency Across Multiple Services and Microservices

Maintaining policy consistency across multiple services and microservices can become challenging as applications scale and evolve. Developers must ensure that authorisation rules are synchronised and that changes propagate correctly throughout the system. For example, a content streaming platform may use microservices for user management, content catalogue, and billing. As new features and services are added, the development team must ensure that authorisation policies remain consistent across all microservices.

Dealing With Policy Conflicts and Resolutions

Increasing the number of policies also increases the potential for conflicts. Developers must be able to detect and resolve these conflicts to ensure that the correct access permissions are applied in all scenarios. For example, in a healthcare application, a doctor may have access to their own patients' records, but access to records of patients outside their care may be restricted to emergency situations. In this case, the development team must implement a mechanism to detect and resolve policy conflicts, ensuring that the doctor can access critical information in emergencies but is restricted from accessing other patients' records in non-emergency situations.

Standardise Communication Across the Business

Communicating effectively will allow your business to implement authorisation quickly; use clear, concise policies that are easy to understand and maintain. Using comments and descriptive names for variables, functions, and classes helps provide context and improve readability. Periodically review and update access control policies to ensure they remain accurate and reflect the current state of your application; this helps to prevent stale or outdated policies from causing security issues.

Create Robust Testing

Before deploying policies to production environments, extensively test them. Automated testing tools and techniques like unit tests and integration tests can help validate policy behavior and ensure it meets requirements. In addition, implement robust logging and monitoring solutions, such as the ELK stack (Elasticsearch, Logstash, and Kibana) or Splunk, to track authorisation activity. Use this data to detect potential issues, analyse trends, and provide evidence for compliance audits.

Decoupled authorisation is essential to modern software development, offering flexibility, scalability, and improved security. However, a business must understand and address the technical challenges of decoupled authorisation to create robust and maintainable applications. Plan carefully and implement a solution that works for you and your organisation; this will help you adapt to changing requirements and regulations more effectively.

Conclusion

Having security concerns means your application stack likely has several layers of authorisation from the network level down to the resource levels and through the application itself. The issue with this approach to authorisation is its complexity and lack of visibility. For instance, at the application level, developers may not realise that they're writing the same code repeatedly to check who can do what.

This is where the idea of decoupling authorisation logic from the main application comes into play. It allows developers to outsource repetitive tasks and offers increased visibility on your policies. This approach reduces the overhead of managing authorisation logic and allows you to focus on implementing business logic. Decoupling authorisation also enables policy frameworks to be tested independently, increases visibility, and provides comprehensive audit trails.

Decoupling authorisation is a powerful tool that can help organisations improve their security and reliability. However, you need to carefully evaluate potential risks before selecting your solution provider. This process includes considering some downsides, such as being aware of vendor lock-in and the need to select a solution that is both performant and highly available.

About Cerbos

If you are interested in a decoupled authorisation solution, consider Cerbos to easily implement and manage fine-grained access control in your applications.

Flagsmith

Get in Touch

Flagsmith is an open source feature flag software that lets developers release with confidence. We work with banks and financial institutions across the world to help them transition to modern feature management and software development, offering on-premise deployments, security features, and technical support to cover your needs. We also partner with OpenFeature to support open standards and prevent vendor lock-in.

"Our development speed and velocity have increased. Mainly, though, I just feel good about releases. I know when I ship something to production it's going to be safe and I won't have to do a thousand tests to make sure I don't miss something. When things are behind a feature flag, I know what is and isn't enabled in production and I have the visibility I need."

Vontobel

Globally active investment firm with Swiss roots

Flagsmith.com