Learn more at https://www.axiomatics.com

- Title
  - Axiomatics: Harnessing Authorization & Gen AI for Enhanced Customer Experiences
- Abstract
  - Digitization of life is leading to an increasing number of apps, data & users. This leads to new opportunities and comes with security risks. While authentication is solved, we need to tackle authorization in a transparent and scalable way. Policy-driven authorization helps you build an effective access control strategy to share data with the right users in the right context and build new customer experiences. This session will show how AI can help implement policy-driven authorization effectively and how authorization can help enable secure AI use cases such as RAG and agentic AI.

About Axiomatics

Axiomatics is the originator and leading provider of runtime, fine-grained authorization delivered with attribute-based access control for applications, Big Data, API Gateways, and microservices.

Some of world's most notable enterprises and government agencies depend on Axiomatics' enterprise proven, flexible, and open authorization solution to share sensitive, valuable and regulated digital assets – but only to authorized users and in the right context.

Source gallica.bnf.fr / BnF

This is the Tower of London, perhaps one of the most secure buildings in London. And yet, scores of tourists walk its hallways and battlements. How does one get in? By “authenticating” at the front door. Tourists will buy a ticket, employees will have a pass. Guards or staff will “authenticate” the ticket or the badge.

Identity & Context Attributes

Access

Date

Type

Photo

Name

Expiry

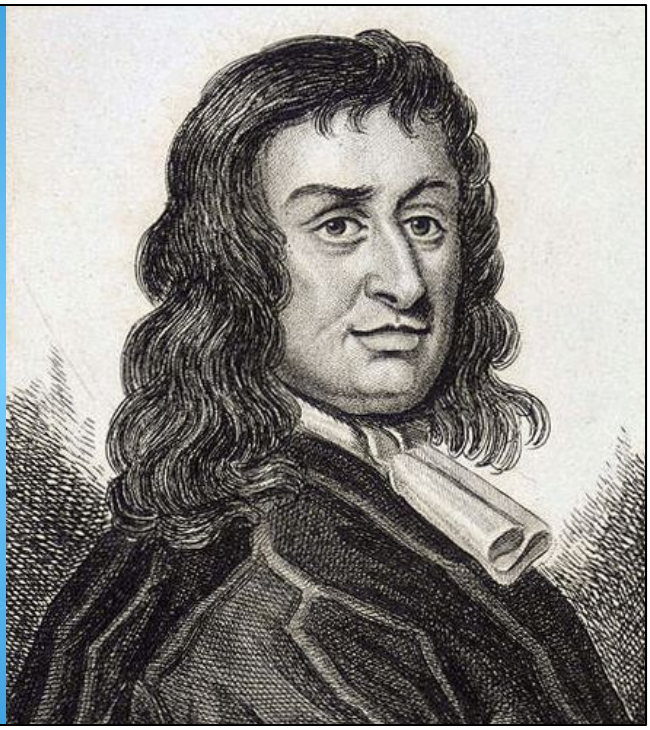
Role

AXIOMATICS

Now of course, based on the type of ‘token’, the user will be granted access to different experiences. The token can include different attributes or claims that further describe the bearer of the token and what they represent.

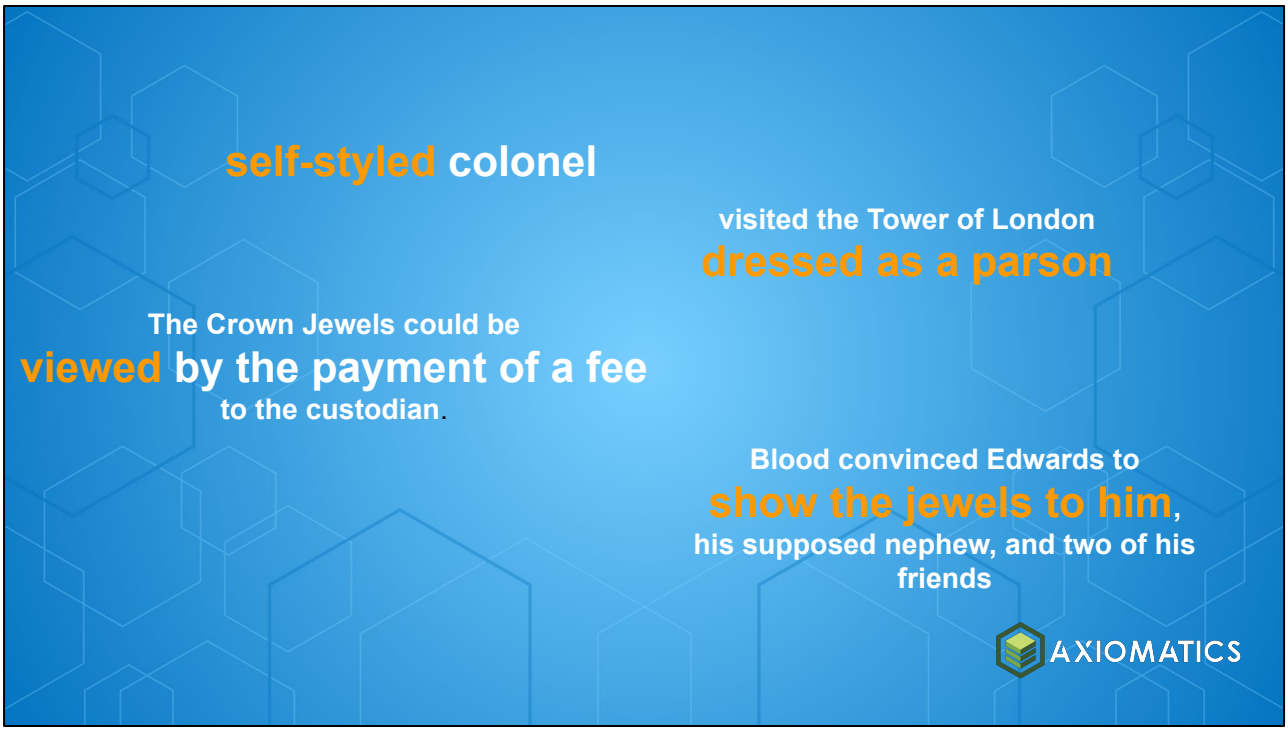
In the year 1637…

Thomas Blood (1618 – 1680)



In 1637, the Tower of London, a fortress of supposed security, was breached. Thomas Blood, a self-styled colonel, dared to challenge the very might of the crown. Disguised as a parson, he weaved a web of deceit, charming his way into the Jewel House. With cunning and treachery, he struck, attempting to steal the priceless Crown Jewels. The Imperial State Crown was flattened, the Sceptre broken, the Orb concealed, a brazen act of defiance. But fate intervened, and the alarm was raised. Blood's audacious plot was foiled, his dreams of glory shattered. Yet, even in capture, he declared it a "gallant attempt," a testament to his audacious ambition. The jewels were recovered, but the tale of Blood's daring heist echoes through the ages, a reminder that even the most secure bastions can be vulnerable.

For Thomas Blood, the goal was self-evident: the crown jewels which could then be dilapidated and sold for a hefty sum. Think about your enterprise: what are your crown jewels? Is it the secret recipe to your most popular product? Is it your customer data? Your algorithms? Do you keep an inventory of what matters most to you? How are those secured? Have you considered the insider threat? You need to get a better handle on your jewels. The Royal Family has successfully guarded theirs since the 1637 theft.



How did Thomas Blood pull it off? It's a mix of failed authentication and absent access control. On one hand, his "attributes" were taken for granted. The fact he was a parson was not challenged. Now of course passkeys weren't prevalent in XVIIth c. London but still guards could have verified Blood's identity more closely.
But beyond that, the fact a visitor would be able to weave their way through to the jewels like Tom Cruise weaves his way past the most technically advanced gates shows that any system needs proper defense in depth and the principle of "never trust always verify". Blood's story illustrates the acute need for Zero Trust and fine-grained, runtime & continuous authorization.

How does this relate to me?

As an attendee at Gartner IAM, why should I care?
Well because authentication is largely a solved challenge and that the next frontier is fine-grained access control.

Data, apps, & users: the crown jewels of today's enterprise

AXIOMATICS

1. We are in an era of exponential growth in data, users, and digital services.
2. Our lives are increasingly digitized, from banking to healthcare to social interactions.
3. This digitization creates vast new opportunities for innovation and connection.
4. The increase in data, users, and services also introduces significant security and access challenges.
5. Enterprises must now protect a much larger and more complex digital landscape.
6. Effectively managing access and security in this environment is crucial for success.
7. These "crown jewels" of data, apps, and users are the assets we need to safeguard.

8. Photo by Alexander Grey on Unsplash
9. Photo by Anna Dziubinska on Unsplash

Traditional access control mechanisms are not enough

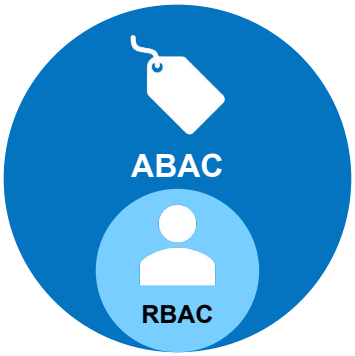
A **self-styled colonel** should not simply get access to your **Crown Jewels**.

Active colonels assigned to the Tower and the King's protection **can access** the Jewels.

Roles (and other identity attributes) alone are not enough: we need **risk scores**.

Contextual data e.g. **time** & day, **location**, and IP should be taken into account.

Move from **role-based access control (RBAC)** to **attribute-based access control (ABAC)**.



AXIOMATICS

You're currently viewing Slide 9, which touches on this very topic! Let's break down why role-based access control (RBAC) often falls short and why attribute-based access control (ABAC) is a more robust solution, as highlighted in your presentation.

Why RBAC is Not Enough:

RBAC grants access based on a user's *role*. While simple and widely used, it has limitations:

- **Granularity Issues:** RBAC struggles with fine-grained access control. If you need to grant different levels of access to users within the same role, RBAC becomes cumbersome. You might end up creating numerous roles, leading to role explosion and management headaches.
- **Context Blindness:** RBAC doesn't consider contextual factors. It doesn't matter if it's 3 AM or 3 PM, or if the user is accessing from a secure network or a public Wi-Fi. If they have the role, they have the access. This lack of context can lead to security vulnerabilities.
- **Limited Flexibility:** Changing access requires modifying role assignments. This can be time-consuming and error-prone, especially in large organizations with complex structures.
- **Doesn't Handle Dynamic Situations Well:** In scenarios where access needs to change rapidly based on events or conditions, RBAC is often too static to adapt effectively.

Why Move to ABAC:

ABAC addresses the shortcomings of RBAC by using *attributes* to make access decisions. Here's why it's a better approach:

- **Fine-Grained Control:** ABAC allows for highly granular access control. You can define policies that consider a wide range of attributes, including user attributes (role, department, clearance), resource attributes (data sensitivity, type), and environmental attributes (time, location, device).
- **Context-Aware Access:** ABAC can factor in the context of the access request. This enables dynamic authorization decisions based on real-time conditions, enhancing security.
- **Flexibility and Adaptability:** Policies in ABAC can be easily updated and modified without changing user roles. This makes it much more flexible and adaptable to changing business needs.
- **Improved Security:** By using a combination of attributes and context, ABAC provides a more secure approach to access control, reducing the risk of unauthorized access.
- **Zero Trust Alignment:** ABAC aligns well with the Zero Trust principle of "never trust, always verify." Access is constantly evaluated based on policies and context, regardless of the user's role.

As seen on this slide:

- "A self-styled colonel should not simply get access to your Crown Jewels." (RBAC might grant access based on a fake role.)
- "Active colonels assigned to the Tower and the King’s protection can access the Jewels." (ABAC can verify specific assignments and attributes.)
- "Roles (and other identity attributes) alone are not enough: we need risk scores." (ABAC can incorporate risk scores as attributes.)
- "Contextual data e.g., time & day, location, and IP should be taken into account." (ABAC excels at considering contextual data.)

In summary, while RBAC provides a basic level of access control, ABAC offers a more dynamic, flexible, and secure solution that can handle the complexities of modern IT environments and the growing need for fine-grained authorization.

Policy-driven authorization

Basics

- Everything is attributes
- Policies combine attributes together
- Attributes can represent contextual data
- Policies are decoupled from data
- Retrieve your data from wherever it lives

Example

- Tourists can visit the Tower of London from 9am to 5pm
- The Yeomen Warders can enter and exit the Tower at any time
- Employees of the Tower can view their own payslips.

Policy-based access control (PBAC) is an authorization model that dynamically grants or denies access to resources based on a set of policies. These policies evaluate various attributes of the user, the resource, the environment, and the action being requested.

Here's a breakdown:

What is PBAC?

- **Policies as the Core:** PBAC revolves around policies that define the rules for access. These policies are typically expressed in a language that can be understood by the authorization system.
- **Attribute-driven:** Decisions are made based on attributes. These attributes can include:
    - **User Attributes:** Role, department, security clearance, etc.
    - **Resource Attributes:** Type of data, sensitivity level, location, etc.
    - **Environment Attributes:** Time of day, location of access, IP address, etc.
    - **Action Attributes:** Read, write, delete, etc.
- **Dynamic Decisions:** PBAC allows for real-time, dynamic authorization decisions. Access can be granted or denied based on the current context and attributes.
- **Decoupled from Applications:** Authorization logic is separated from

- application code, making it easier to manage and update policies without changing the applications themselves.

Benefits of PBAC:

- **Fine-Grained Control:** PBAC enables highly granular control over access to resources. You can create very specific policies to address complex access requirements.
- **Flexibility and Adaptability:** Policies can be easily updated and modified to reflect changing business needs or security requirements. This makes PBAC more adaptable than traditional role-based access control (RBAC).
- **Context-Aware Access:** PBAC can consider contextual factors when making authorization decisions. This allows for more intelligent and secure access control.
- **Improved Security:** By enforcing precise access policies, PBAC helps to reduce the risk of unauthorized access and data breaches.
- **Simplified Management:** Centralized policy management simplifies administration and auditing. Policies can be managed in one place, making it easier to ensure consistency and compliance.
- **Zero Trust Alignment:** PBAC supports the Zero Trust principle of "never trust, always verify" by continuously evaluating access requests based on policies and context.
- **Scalability:** PBAC can scale to handle large numbers of users, resources, and complex access requirements.
- **Auditability:** PBAC systems often provide detailed logs of authorization decisions, which can be used for auditing and compliance purposes.

In essence, PBAC provides a more robust, flexible, and secure approach to access control compared to traditional methods.

Policy-driven authorization benefits

- Runtime, continuous enforcement (Zero Trust)
- Context-aware & Relationship-based
- Decoupled from the protected application
- Write once, use everywhere
- Consistent authorization across all apps, APIs, and services
- Easy to audit
- Simplifies the need for roles, role engineering, and user access recertification

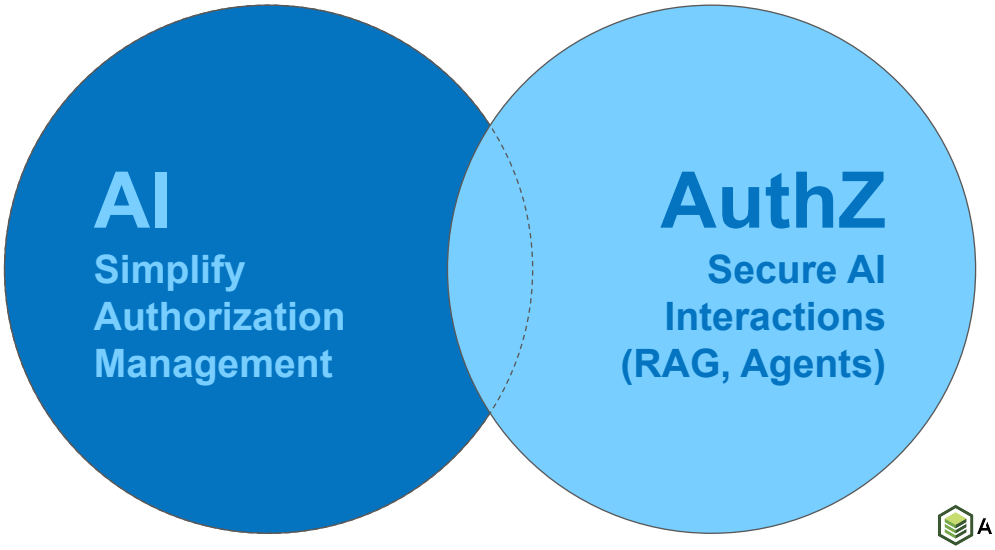
AXIOMATICS

Slide 11 Notes: Policy-driven authorization benefits

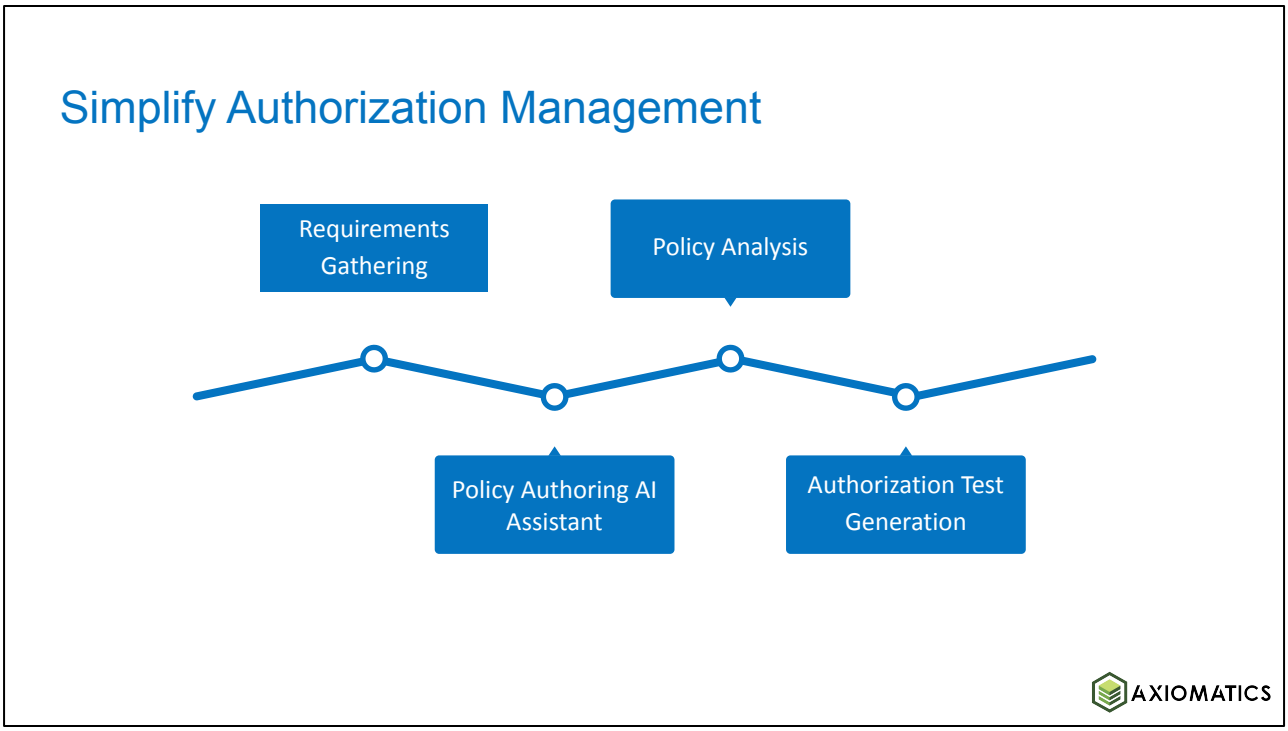
- **Runtime, continuous enforcement (Zero Trust):** Authorization decisions are made in real-time, continuously verifying access. Aligns with Zero Trust principle of "never trust, always verify."
- **Context-aware & Relationship-based:** Takes into account various contextual factors (time, location, etc.) and relationships between entities when making authorization decisions.
- **Decoupled from the protected application:** Authorization logic is separate from the application code, making it easier to manage and update.
- **Write once, use everywhere:** Policies can be reused across different applications and systems, promoting consistency.
- **Consistent authorization across all apps, APIs, and services:** Ensures uniform access control across the entire organization.
- **Easy to audit:** Centralized policy management simplifies auditing and compliance efforts.
- **Simplifies the need for roles, role engineering, and user access recertification:** Reduces complexity compared to traditional role-based access control (RBAC).

What about AI?

The Venn of AI & Authorization



AXIOMATICS



Policy Companion is an AI-based tool that lowers the entry point for authoring effective attribute-based access control (ABAC) policies and the ability to interpret existing policy code, to anyone who can read English.

Learn to author policies without being an expert
Policy authoring can now be started by anyone, without extensive coding experience or knowledge of fine-grained access control.
Policy Companion leverages generative AI to evaluate requirements and provide specific recommendations, making it easy for anyone to quickly understand the concepts around authoring effective policies.
Bridge natural language requirements with machine readable code
There are often different personas, who speak different languages, involved in authoring and creating authorization policies. For example, business personas speak in more natural language, where developers better understand more structured language that leads to machine readable code.
With Policy Companion, as business users generate requirements, the actual machine-readable code is also created for the developer.

Microsoft Copilot significantly enhances the process of authoring Abbreviated Language for Authorization (ALFA) policies directly within a developer's Integrated Development Environment (IDE), such as VS Code. Leveraging its AI capabilities, Copilot can suggest and generate ALFA code based on natural language descriptions or existing policy snippets. This seamless integration streamlines policy creation by reducing manual coding and minimizing errors. Developers can quickly translate

business requirements into precise authorization rules, receive real-time feedback, and explore different policy structures, all within their familiar coding environment. This accelerates development cycles, improves policy accuracy, and empowers developers to efficiently manage fine-grained access control for their applications.

Policy Companion puts everyone on the same page and bridges the gap between developer, security, and business users.

AI can help determine & structure requirements

Example

I am building a ride-sharing app. Riders should be able to book trips. Drivers should be able to choose customers.

AI Outcome

Suggested authorization: riders can view their own rides; a driver cannot view the payment profile of a rider.

AI can help translate requirements into policy

Requirement

Accountants can view financial records if they are in the same department as the user.

Because ALFA, the Abbreviated Language for Authorization, is a standard, Microsoft Copilot and other AIs already understand it without specific training.

Authorization Policy (ALFA)

```
namespace copilot{
    // Policy to control access to financial records
    policy accessFinancialRecords {
        apply denyOverrides

        // Rule to allow accountants to view financial records if they are
        rule allowAccountantsView {
            target clause accessedResource == "financial-record"
            permit
            condition userType == "accountant" && accountantDepartment == 
        }

        // Rule to allow patients to view their own financial records
        rule allowPatientsView {
            target clause accessedResource == "financial-record"
            permit
            condition userType == "patient" && userId == patientId
        }

        // Rule to deny access to non-financial staff
        rule denyNonFinancialStaff {
            target clause accessedResource == "financial-record"
```

AXIOMATICS

AI can help analyze what a policy does

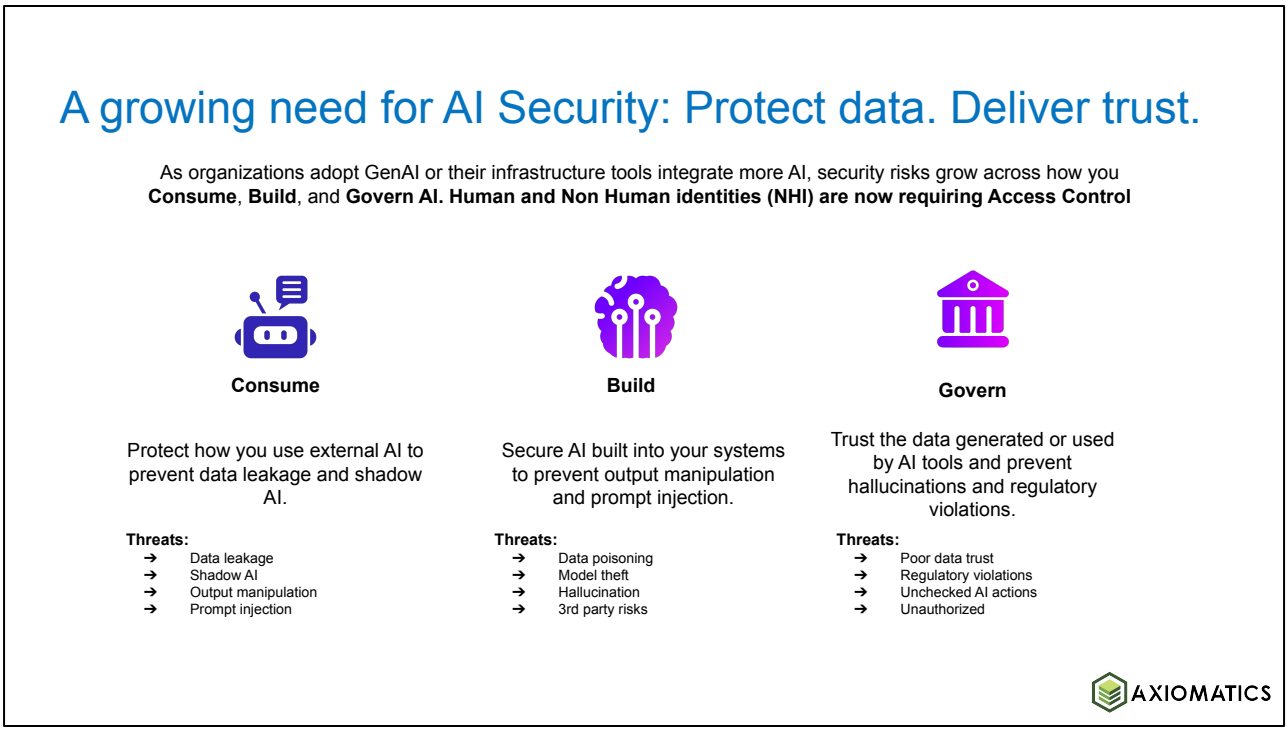
```
namespace copilot{
  policyset records{
    target clause Attributes.objectType == "record"
    apply firstApplicable
    policy viewRecords{
      target clause Attributes.actionId == "view"
      apply firstApplicable
      rule managers{
        target clause example.userRole == "manager"
        permit
      }
    }
  }
}
```

Explanation:

1. **Namespace**: copilot
   - This defines the namespace for the policy, which is copilot.
2. **Policy Set**: records
   - **Target Clause**: Attributes.objectType == "record"
     - This policy set applies to objects of type "record".
   - **Combining Algorithm**: apply firstApplicable
     - the first applicable policy will be applied.
3. **Policy**: viewRecords
   - **Target Clause**: Attributes.actionId == "view"
     - This policy applies to actions with the ID "view".
4. **Rule**: managers
   - **Target Clause**: example.userRole == "manager"
     - This rule applies to users with the role "manager".
   - **Effect**: permit
     - This rule permits the action if the target clause conditions are met.

Summary: This policy allows users with the role "manager" to view records. The policy set `records` applies to objects of type "record", and within this set, the policy `viewRecords` applies to action "view".

AXIOMATICS



Slide 18 Notes: A growing need for AI Security: Protect data. Deliver trust.

- **AI Security Risks:** AI adoption increases security risks across consumption, building, and governance.
- **Human and Non-Human Identities (NHI):** Access control is now needed for both human and AI identities.
- **Consume:** Protect external AI use to prevent data leaks and shadow AI.
- **Build:** Secure AI in systems to stop output manipulation and prompt injection.
- **Govern:** Ensure AI data is trusted and prevent hallucinations and regulatory violations.
- **Threats:** Include data leakage, shadow AI, output manipulation, prompt injection, data poisoning, model theft, hallucinations, regulatory violations, and unauthorized actions.

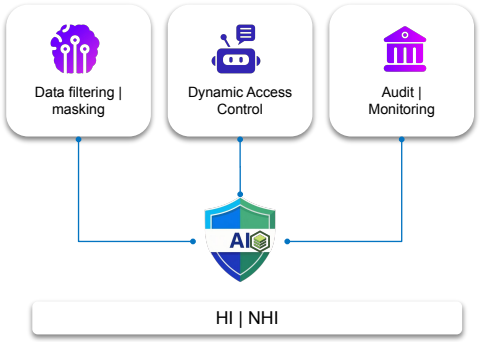
De-risk AI Security | Leverage PBAC

Axiomatics secures AI with a dynamic, least privilege access framework aligned to Gartner's AI Security model. Axiomatics extends identity-first, to both human and non-human (AI) identities — protecting data and AI interactions end to end.

Axiomatics AI-driven use cases

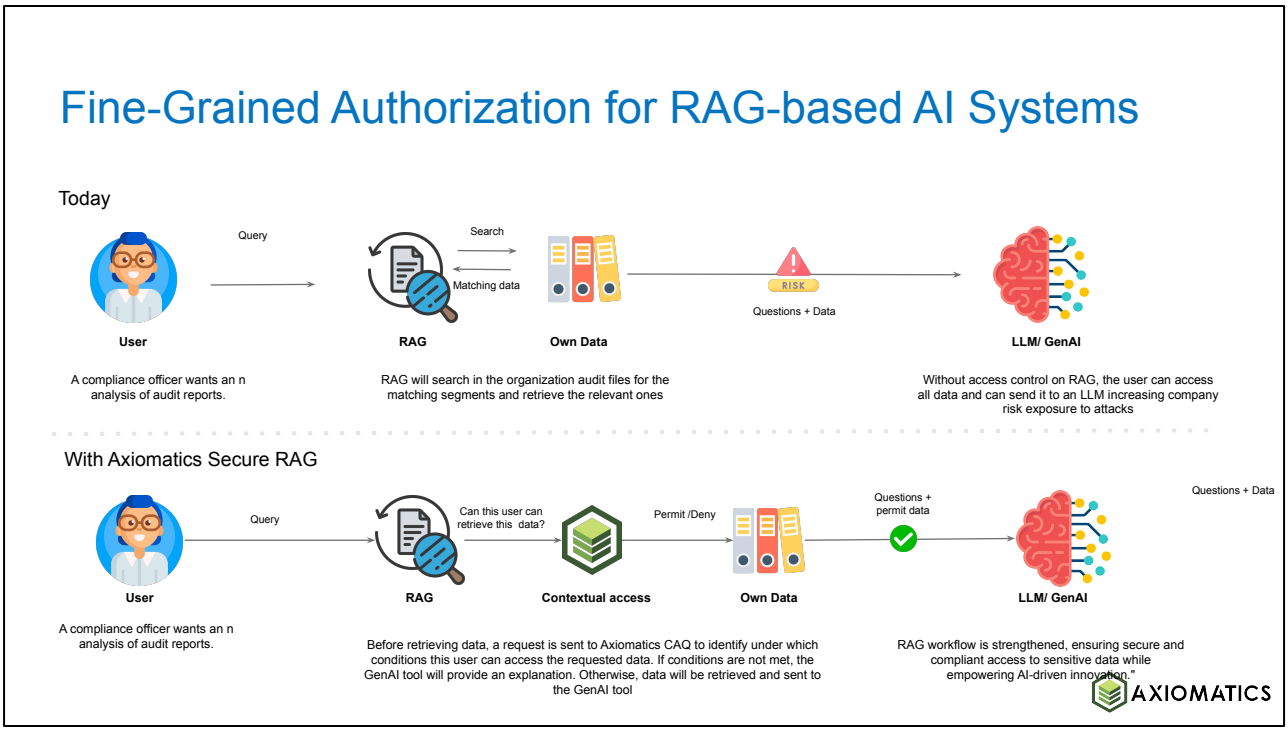
1. **AI Dynamic Access Control**
   How do we stop unauthorized users (HI /NHI) from using internal AI services?
2. **Data filtering for AI agents**
   How do we prevent AI tools from accessing sensitive data or user to share sentive data.
3. **Secure RAG**
   How do we ensure AI only retrieves appropriate data for response generation?
4. **Monitor AI usage**
   How do we ensure AI outputs comply with security policies?
   How do we govern AI usage across multiple systems?

Axiomatics AI Security framework



AXIOMATICS

- **Transparency:** Understand PBAC ensure clear and enforceable policies make AI decisions more accountable.
- **Control:** Organizations retain full control over AI interactions, ensuring that models operate within ethical and security boundaries.
- **User Confidence:** Employees, partners, and customers trust AI systems when they know access is securely managed and monitored.



RAG allows companies to connect their data with LLMs, enabling artificial intelligence opportunities for businesses that are more trustworthy, pertinent, and timely [source]
With policy-driven authorization:

1. Enforce policies to ensure the user can invoke the AI agent
2. Create a filter/query plan that defines the constraints that must be applied when fetching underlying data
3. The data provided to the LLM is therefore entitled data.
4. The result of the LLM is contextual, relevant, and compliant with the authZ.

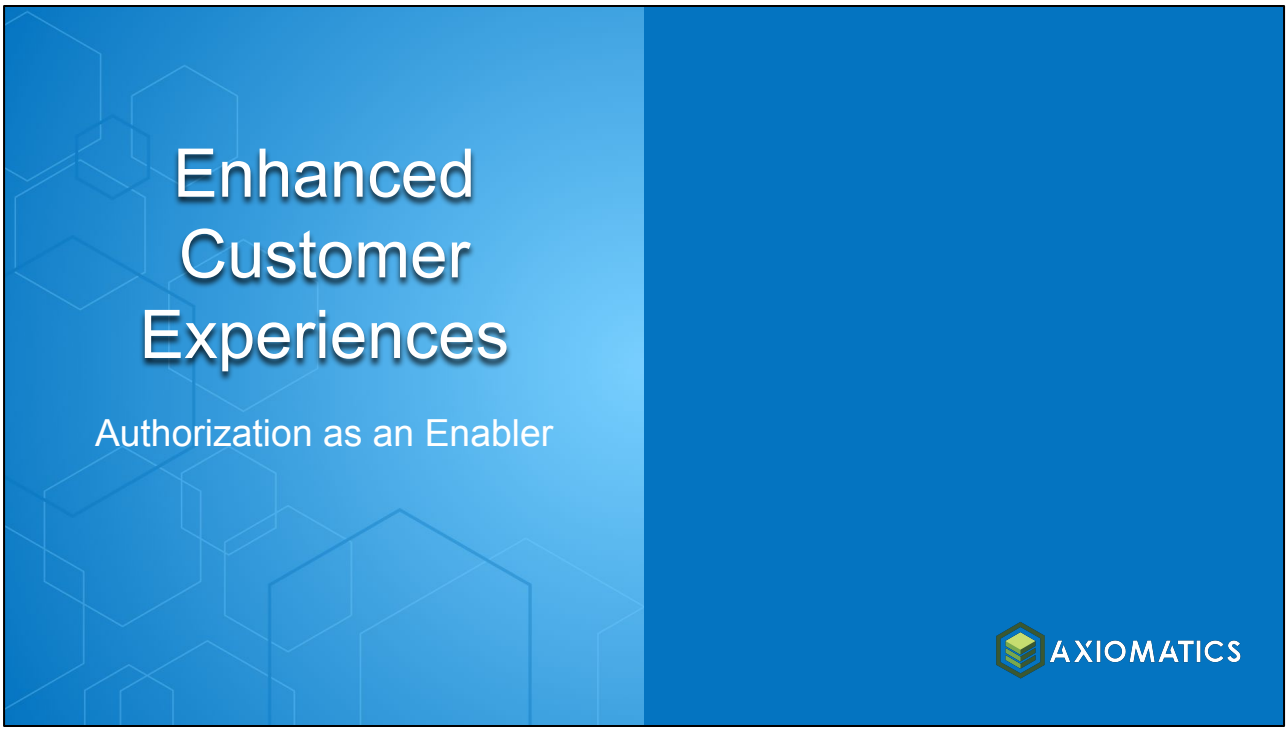
Today (Without Axiomatics Secure RAG):

- **User:** A compliance officer wants an analysis of audit reports.
- **RAG:** Searches the organization's audit files for matching segments and retrieves them.
- **LLM/GenAI:** Receives the retrieved data and answers the query.
- **Risk:** Without access control on RAG, the user can access all data, increasing company risk exposure to attacks and data breaches.

With Axiomatics Secure RAG:

- **User:** A compliance officer wants an analysis of audit reports.
- **RAG:** Before retrieving data, a request is sent to Axiomatics CAQ to determine under which conditions the user can access the requested data.
- **Axiomatics CAQ:** Evaluates the request and decides to Permit or Deny

- access based on policies and context.
- **Contextual Access:** If permitted, the relevant data is retrieved and sent to the GenAI tool. If denied, an explanation is provided.
- **LLM/GenAI:** Receives the permitted data and answers the query.
- **Strengthened RAG Workflow:** Ensures secure and compliant access to sensitive data, empowering AI-driven innovation.
- **Key Benefit:** Prevents unauthorized access and data leakage by implementing fine-grained authorization before data retrieval.



* Fine-grained authorization is often perceived as solely a tool for compliance and security, creating a sense of fear, uncertainty, and doubt (FUD).
* This perception limits fine-grained authorization to merely locking down systems and restricting access, which is a narrow view.
* In reality, fine-grained authorization enables the creation of new business opportunities and innovative consumer interactions.
* With its precise control, fine-grained authorization unlocks use cases that were previously deemed impossible or too risky.
* By allowing for granular control over data and access, organizations can explore novel ways to engage with customers and partners.

A story of a digital life

Put data ownership back into the right hands

- Banking data (Open Banking)
- Health data (FHIR)
- Insurance data

My story

- Medical results from Chicago shared with my doctor in Canada

Allow sharing of data (under the right circumstances) to enable

- New business opportunities
- Research (medical)
- Law enforcement

AXIOMATICS

Slide 23 Notes: A story of a digital life

- **Put data ownership back into the right hands:** Emphasize the importance of individuals controlling their own data.
- **Examples:**
  - **Banking data (Open Banking):** Users can selectively share banking information with third-party services.
  - **Health data (FHIR):** Secure and controlled sharing of health records.
  - **Insurance data:** Individuals manage access to their insurance information.
- **My story:** Illustrate with a personal example: Medical results from Chicago shared with my doctor in Canada, highlighting secure cross-border data sharing.
- **Allow sharing of data (under the right circumstances) to enable:**
  - **New business opportunities:** Creating innovative services through data sharing agreements.
  - **Research (medical):** Facilitating medical research with controlled data access.
  - **Law enforcement:** Enabling data sharing for investigations while respecting privacy.

What about the Crown Jewels?

Today you can visit the Tower of London and admire the jewels from afar.

You cannot touch or handle the jewels

Come see us Booth 310

Want to learn more? Visit Axiomatics at booth 310 or contact us to speak with our team about our authorization solution.

Additional resources

- State of Authorization: Playbook Edition
- Policy Companion — an AI-based tool for policy authoring
- Axiomatics Deployment Methodology