



Onix AutoACL: An AI-powered approach to adaptive security in Chrome Enterprise Premium



Table of Contents



▶ Abstract	03
▶ Introduction	04
Business value and justification	04
Enhanced security	04
Increased efficiency	04
Enhanced business agility	05
Cost optimization	05
▶ AutoACL architecture	06
Core components	06
Data flow	06
▶ Implementation details	07
Data preprocessing and training	07
Integration with Google services	07
Open-source tool integration	07
▶ Security considerations	08
▶ Future directions	08
▶ Conclusion	08
▶ Technical addendum: Implementing AutoACL in Chrome Enterprise Premium	09





Abstract

This white paper provides an overview of the capabilities, technical design, and implementation of AutoACL, an innovative solution initially developed by Google, but enhanced by Onix to help organizations improve security for their Chrome Enterprise Premium (CEP) deployments. AutoACL uses Google's PaLM2 large language model to improve access control management within CEP.

This document explores the underlying technologies, integration with Google Cloud and Chrome services, and utilization of open-source tools to provide guidance to organizations seeking to enhance their security posture through AI-driven access control in CEP.





Introduction

Traditional access control mechanisms, particularly Access Control Lists (ACLs) using role-based access controls (RBAC) or static permissions present complexity, management, scalability, and overhead challenges for organizations. Manually configuring and maintaining ACLs for a large number of users, devices, and resources for resource targets whose access needs to be adjusted based on business conditions and/or market and organizational changes can be error prone and time-consuming. This increases business, technical, and operational risk and frustrates end-users and non-employees such as contractors.

AutoACL addresses these challenges directly by enabling the definition and modification of security policies using natural language, simplifying access control management and promoting least privilege principles which allow the organization to scale without having to worry about legacy access challenges.



Business value and justification

Deploying and using AutoACL for CEP can offer significant upside to medium and large organizations. By streamlining and automating access control management, AutoACL delivers benefits that positively impact security posture, operational efficiency, and overall business agility. Simplifying security is a win for any organization.

Organizations that use AutoACL may see the benefits outlined in this document (note: it assumes the organization has built and uses buttoned-down role-based access controls to support their operations).




Enhanced security



- **Reduced human error:** AutoACL minimizes the risk of human error associated with manual ACL configuration, leading to more accurate and reliable access control, with faster response times for end-user customers.
- **Least privilege enforcement:** AutoACL simplifies the implementation of least privilege principles, ensuring that users have only the necessary access permissions to perform their job functions.
- **Improved compliance:** AutoACL helps organizations meet regulatory compliance requirements by providing a clear and auditable access control framework. Framework goes into the tool and the ACL output will reference this information to ensure compliant access is provided.



Increased efficiency

- 
- **Reduced administrative overhead:** AutoACL automates ACL management tasks, freeing up IT staff to focus on strategic initiatives and top business priorities.
 - **Improved productivity:** Streamlined, lean processes will enhance employee and contractor productivity by minimizing access-related delays and frustrations. Onboard and offboard faster and handle organizational changes quickly.



Enhanced business agility



- **Scalability:** AutoACL's architecture is designed to scale with organizational growth, accommodating increasing numbers of users, devices, and resources.
- **Adaptability:** AI-driven access control enables dynamic adaptation to evolving business requirements and threat landscapes.
- **Innovation:** By reducing the complexity of access control management, AutoACL empowers organizations to focus on innovation and digital transformation initiatives.



Cost optimization

- **Reduced operational costs:** Automation and efficiency gains lead to lower operational costs associated with access control management.
- **Reduced security incident costs:** Enhanced security posture can lower the risk and costs of security breaches and data leaks.



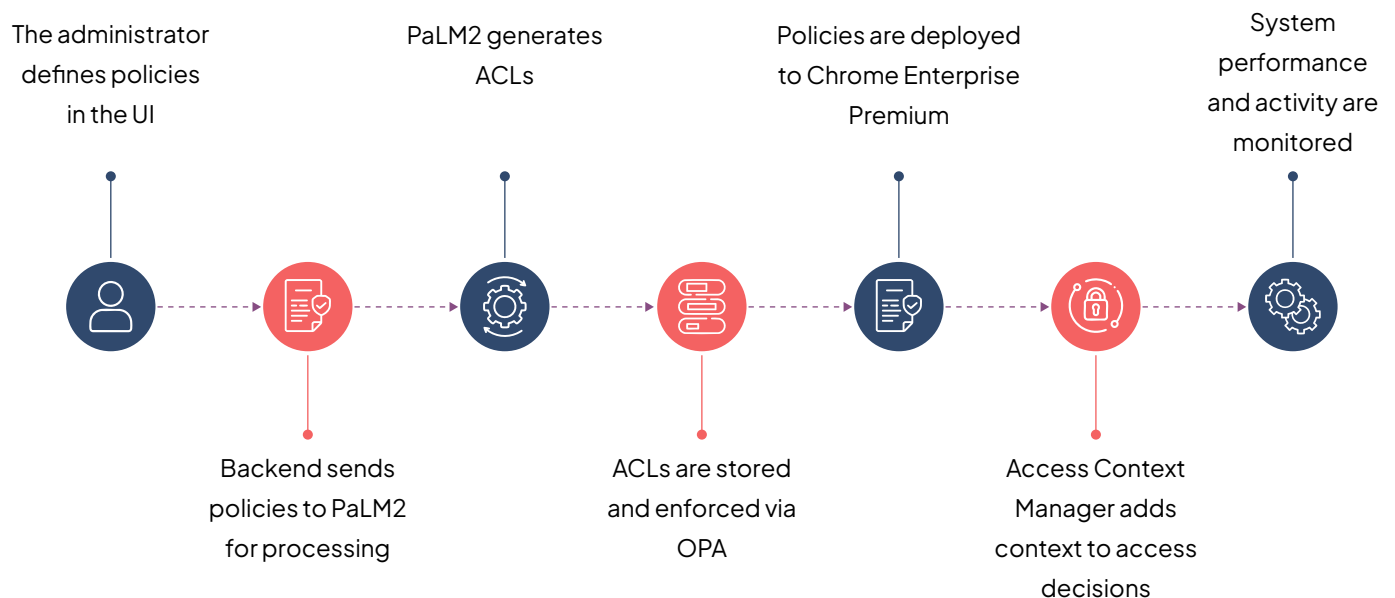
AutoACL architecture

AutoACL relies on Google Cloud products and services as well as open-source tools to provide a flexible, robust, and scalable access control solution within Chrome Enterprise Premium.

Core components

- **User Interface:** A web interface where administrators define policies in natural language.
- **Backend Service:** Handles policy processing, interacts with PaLM2 (via Vertex AI), and generates ACLs.
- **PaLM2:** Google's language model interprets natural language and generates ACLs.
- **Cloud Storage:** Stores training data, model checkpoints, and generated ACLs.
- **Cloud Run:** Serverless environment for deploying and scaling the backend service.
- **OPA:** Open-source policy engine for fine-grained access control enforcement.
- **Chrome Enterprise Premium:** Integrates for policy deployment and enforcement on Chrome devices.
- **Access Context Manager:** Provides contextual information (location, device state) for access decisions.
- **Monitoring and Logging:** Tracks performance and activity for operational insight.

Data flow



Implementation details

Data preprocessing and training

- **Dataset creation:** A comprehensive dataset of access control policies and corresponding ACL entries is curated from various sources, including existing security policies, best practices, and real-world scenarios.
- **Data preprocessing:** The dataset is preprocessed to clean, normalize, and format the data for optimal training of the PaLM2 model.
- **Model training:** PaLM2 is fine-tuned on the preprocessed dataset using Google's advanced machine learning infrastructure. This fine-tuning process enables the model to effectively understand and generate ACLs from natural language instructions.

Integration with Google services

- **Google Cloud Storage:** Google Cloud Storage is used to store the training data, model checkpoints, and generated ACLs. This ensures data durability, scalability, and secure access control.
- **Cloud Run:** Cloud Run is employed to deploy and manage the AutoACL application, providing a serverless platform for handling access control requests with automatic scaling and cost optimization.
- **Cloud Logging and Monitoring:** Cloud Logging and Monitoring services are integrated to track AutoACL's performance, identify potential issues, and ensure operational efficiency.

Open-source tool integration

- **Open Policy Agent (OPA):** This open-source tool provides a general-purpose policy engine used for policy evaluation and enforcement. This allows for flexible and granular control over access permissions.





Security considerations

AutoACL incorporates several security measures to mitigate potential risks associated with AI-powered access control:

Prompt engineering: Careful prompt engineering techniques are employed to guide the PaLM2 model towards generating secure and compliant ACLs.

Input sanitization: All user inputs are sanitized to prevent malicious code injection and other security threats.

Model explainability: Techniques for model explainability are employed to understand the reasoning behind the generated ACLs, ensuring transparency and accountability.

Auditing and logging: Comprehensive auditing and logging mechanisms are implemented to track all access control activities and facilitate security investigations.


Future directions

AutoACL is an evolving solution with ongoing development and enhancements. Future directions include:

Enhanced natural language understanding: Improving the model's ability to understand complex and nuanced natural language instructions for access control.

Contextual awareness: Incorporating contextual information, such as user location, device posture, and time of day, into access control decisions.

Automated policy optimization: Developing AI-driven techniques to automatically optimize access control policies based on usage patterns and security best practices.




Integration with threat intelligence: Integrating threat intelligence feeds to proactively adapt access control policies based on emerging threats.

Conclusion

AutoACL is a giant leap forward in access control management using AI. It helps simplify policy definition, reduce errors, and improve security within Chrome Enterprise Premium. By combining Google's cutting-edge technologies with open-source tools, AutoACL provides a robust and scalable solution for organizations seeking to implement adaptive and efficient access control, reduce the support burden, and enable their end-users.

Even as a tool to audit or check the validity of access settings based on what an analyst or administrator observes in the environment, AutoACL can provide assurance that the proper access controls are in place very quickly. AutoACL could act as a second line of defense and also be used by auditors to identify weaknesses and control issues for access management reducing the amount of work required.

As AI continues to advance, enhancements to AutoACL will play a crucial role in enabling organizations to achieve a secure and productive digital workplace while minimizing security headaches for their support teams and end-users.





Technical addendum: Implementing AutoACL in Chrome Enterprise Premium

This addendum provides a step-by-step guide for CEP administrators to implement AutoACL in their organization.

Step 1: Accessing PaLM2


- Vertex AI: PaLM2 can be accessed through Vertex AI, Google Cloud's machine learning platform
 - Sign up for a Google Cloud account and create a project. [Link to Google Cloud signup](#)
 - Enable the Vertex AI API in your project. [Link to Vertex AI documentation](#)
 - Explore PaLM 2 models available within Vertex AI. [Link to PaLM 2 models](#)



Step 2: Building the AutoACL interface

- Frontend development: Develop a user-friendly web interface for security administrators to input natural language access control policies
 - Utilize a frontend framework like React or Angular to build the interface
 - Design input fields for specifying subjects, objects, and actions in natural language
 - Include validation to ensure complete and accurate policy definitions
- Backend development: Develop a backend service to handle policy processing and ACL generation
 - Use a backend framework like Node.js or Python (Flask/Django)
 - Implement API calls to interact with the PaLM2 model through Vertex AI
 - Integrate with Google Cloud Storage to store policies and generated ACLs

Step 3: Configuring open-source tools

- Open Policy Agent (OPA):
 - Download and install OPA. [Link to OPA documentation](#)
 - Define policy rules in Rego, OPA's policy language, to enforce access control based on the generated ACLs
 - Integrate OPA with your application to evaluate access requests against the policies
 - OWASP ZAP:
 - Download and install OWASP ZAP. [Link to OWASP ZAP website](#)
 - Configure ZAP to perform automated security scans on the AutoACL application and generated ACLs
 - Analyze scan results to identify and address potential vulnerabilities
- 

- SonarQube:
 - Download and install SonarQube. [Link to SonarQube website](#)
 - Configure SonarQube to analyze the AutoACL codebase for code quality and security issues
 - Integrate SonarQube with your development pipeline for continuous code quality assessment

Step 4: Deploying on Cloud Run

- Containerize the application: Package the AutoACL application and its dependencies into a Docker container
- Deploy to Cloud Run: Deploy the container to Cloud Run. [Link to Cloud Run documentation](#)
- Configure service: Set up environment variables, scaling options, and security settings for the Cloud Run service

Step 5: Integrating with Chrome Enterprise Premium

- API integration: Utilize the Chrome Enterprise Premium API to programmatically manage user access and device policies. [Link to Chrome Enterprise Premium API documentation](#)
- Policy enforcement: Enforce the generated ACLs by configuring user and device policies within Chrome Enterprise Premium
- Access Context Manager: Leverage Access Context Manager to define contextual conditions for access control, such as user location, device posture, and time of day. [Link to Access Context Manager documentation](#)

Step 6: Monitoring and maintenance

- Cloud logging: Monitor AutoACL logs for errors, performance issues, and security events
- Cloud monitoring: Set up monitoring dashboards to track key metrics, such as request latency, error rates, and resource utilization
- Regular updates: Regularly update the PaLM2 model and AutoACL application to benefit from the latest improvements and security patches. Example Code Snippets:





Interacting with PaLM2 through Vertex AI (Python):

Python

```
Userent

from google.cloud import aiplatform
•
•
• aiplatform.init(project="your-project-id", location="us-central1")
•
• model = aiplatform.Model("text-bison@001")
•
• prompt = "Allow engineers in the development team to access the code repository."
•
• response = model.predict(prompt=prompt)
•
• print(response.predictions[0].text)
```



Use code with caution.

Defining OPA Policy (Rego):

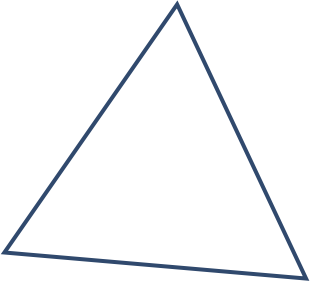
Code snippet

```
Userent

• package example
•
•
• allow {
•   input.user.role == "engineer"
•   input.user.team == "development"
•   input.resource == "code repository"
• }
```



Use code with caution.



Enforcing Policy with Chrome Enterprise Premium API (Python):

Python

Usent

```
• from googleapiclient.discovery import build
•
• service = build('chromemanagement', 'v1')
•
• policy = {
•   'policySchema': 'chrome.users.policies',
•   'policyValue': {
•     'URLBlacklist': ['*.example.com']
•   }
• }
•
• request = service.customers().policies().orgunits().create(
•   customer='my_customer',
•   orgUnitId='my_org_unit',
•   body=policy
• )
•
• response = request.execute()
•
• print(response)
```

Use code with caution.

Note: These code snippets are for illustrative purposes and may require modifications based on your specific implementation.

This detailed guide, combined with the comprehensive information in the white paper, empowers CEP administrators to effectively implement and manage AutoACL, enhancing their organization's security posture with AI-driven access control.

Get in touch



onixnet.com



connect@onixnet.com