Formal Model for Permission Matrices and Access Policy Chains

Samuel Imboden

April 2025

1 Introduction

This paper is mainly for myself to write my thoughts down.

1.1 The Problem

Given the task of designing the access policies for a zero trust-based network application, I ran into a massive problem. The way this tool defines what an application is, combined with the simple policy evaluation engine, forces me to write many policies, each with duplicated logic.
Explain the rule engine...

1.2 Proposed Solution

To help with designing the policies, we can first define a more complex permission structure that makes sense from an operational standpoint. This structure can then be converted through a series of mathematical formulas into a set of rules that works for the application.

1.3 Process Overview

The complete process of defining and simplifying permission-based access control can be described in the following steps:

1. **The process starts with knowing your environment:**
   Define set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ for users and set $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ for applications.

2. **Define group-based permissions using a matrix:**
   Define groups $\mathcal{G}_U \subseteq \mathcal{P}(\mathcal{U} \cup \mathcal{G}_U)$ of users and,
   if applicable, groups of applications $\mathcal{G}_A \subseteq \mathcal{P}(\mathcal{A} \cup \mathcal{G}_A)$.

   Then define a permission matrix $M : \mathcal{G}_U \times \mathcal{G}_A \to \{0, 1\}$.

3. **Flatten the matrix to unique users and applications:**
   Recursively expand each group using flattening functions to get sets of atomic users and applications.

4. **Convert the flat matrix to a Boolean formula:**
   Construct a disjunctive normal form (DNF) formula:

$$\Phi(u, a) = \bigvee_{(i,j) \text{ where } P[i][j]=1} (U_i \wedge A_j)$$

5. **Simplify the Boolean formula:**
   Use Boolean minimization techniques (e.g., Quine-McCluskey or Espresso) to derive an equivalent minimal formula $\Phi'(u, a)$.

6. **Generate access policy chain from simplified formula:**
   Convert each term in Φ' into a conditional rule $R_k = (\text{condition}_k(u, a), \text{action}_k)$, forming an ordered policy chain.

2 Definitions

2.1 Permission Matrix

Let:

- $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of all users
- $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ be the set of all applications
- $\mathcal{G}_U \subseteq \mathcal{P}(\mathcal{U} \cup \mathcal{G}_U)$ be the set of user groups, possibly nested
- $\mathcal{G}_A \subseteq \mathcal{P}(\mathcal{A} \cup \mathcal{G}_A)$ be the set of application groups, possibly nested
- $P \colon \mathcal{G}_U \times \mathcal{G}_A \longrightarrow \{0, 1\}$ be the permission matrix

2.2 Group Flattening

We define the flattening functions recursively:

$$\text{flatten}_U(G) = \bigcup_{x \in G} \begin{cases} \{x\}, & \text{if } x \in \mathcal{U} \\ \text{flatten}_U(x), & \text{if } x \in \mathcal{G}_U \end{cases}$$

$$\text{flatten}_A(G) = \bigcup_{x \in G} \begin{cases} \{x\}, & \text{if } x \in \mathcal{A} \\ \text{flatten}_A(x), & \text{if } x \in \mathcal{G}_A \end{cases}$$

2.3 Expanded Permission Function

Define the expanded access relation:

$$(u, a) \in P \iff \exists G_U \in \mathcal{G}_U, G_A \in \mathcal{G}_A$$
$$\text{such that}$$
$$P[G_U][G_A] = 1 \wedge u \in \text{flatten}_U(G_U) \wedge a \in \text{flatten}_A(G_A)$$

2.4 Access Policy Chain

Policy definition:

$$R_k = (\text{condition}_k(u, a), \text{action}_k) \quad \text{with action}_k \in \{\text{allow}, \text{deny}\}$$

Access decision function:

$$\text{Access}(u, a) = \begin{cases} \text{action}_k, & \text{for the first } k \text{ such that condition}_k(u, a) = \text{true} \\ \text{deny}, & \text{if no such } k \text{ exists} \end{cases}$$