AQUA: An Efficient Solver for the User Authorization Query Problem

Alessandro Armando  
alessandro.armando@unige.it  
DIBRIS, Università di Genova  
Genova, Italia

Giorgia A. Gazzarata  
giorgia.gazzarata@unige.it  
DIBRIS, Università di Genova  
Genova, Italia

Fatih Turkmen  
f.turkmen@rug.nl  
University of Groningen  
Groningen, Netherlands

ABSTRACT

We present AQUA, a solver for the User Authorization Query (UAQ) problem in Role-Based Access Control (RBAC). The UAQ problem amounts to determining a set of roles granting a given set of permissions, satisfying a collection of authorisation constraints (most notably Dynamic Mutually-Exclusive Roles, DMER) and achieving some optimization objective, i.e. seeking min/max/any number of roles to activate and/or permissions to grant. AQUA supports the enforcement of a wide class of DMER constraints as well as several types of optimization objectives (namely, min/max/any number of roles to activate, min/max/any number of permissions to grant, and a combinations thereof). In this paper, we demonstrate the use of AQUA over a running example while providing certain implementation details including the architecture.

ACM Reference Format:

1 INTRODUCTION

Role-based access control (RBAC) has proven to be one of the most successful access control models thanks to its intuitiveness, simplicity and standard support. Today, many authorization systems support at least one form of RBAC among several variants the standard [7] provides. One of the key issues in RBAC is, given a set of requested permissions, finding an optimal set of roles in a given session that will both cover the requested permissions and satisfy a collection of Dynamic Mutually-Exclusive Roles (DMER) constraints. This problem, known as the User Authorization Query (UAQ), has been well studied in the literature, however there is little tooling (e.g. solver, benchmark generator) support to study its different aspects.

In this paper we present AQUA, a solver for the UAQ problem that combines (i) a reduction of various types of DMER constraints (in a given state) to an equivalent set of single-session DMER constraints, (ii) a reduction of the UAQ Problem to the Weighted Maximum Satisfiability problem (Weighted MAX-SAT), and (iii) the application of state-of-the-art Weighted MAX-SAT solvers to tackle the problem. To the best of our knowledge AQUA is the only solver that simultaneously offers all the above features.

1.1 Running Example

Consider a data visualization application displaying plots about a pandemic and RBAC is employed to control access to (sensitive) plots based on user’s privileges. Table 1 summarizes the RBAC policy in terms of PA relation. In the RBAC policy, there is a single operation, “view”, that can be performed on the plots and thus the plots uniquely identify the permissions. We assume that there is a single user (alice) and a single session (s1) in the system for the simplicity of discussion.

<table>
<thead>
<tr>
<th>Roles</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>{p1, p2}</td>
</tr>
<tr>
<td>Guest</td>
<td>{p1, p2}</td>
</tr>
<tr>
<td>Researcher</td>
<td>{p3, p4, p5}</td>
</tr>
<tr>
<td>BoardMember</td>
<td>{p3, p4}</td>
</tr>
</tbody>
</table>

Table 1: PA for the Visualization Application

The system has a single DMER constraint:

\[ \text{DMER}(\text{Researcher}, \text{BoardMember}), 2 \]

that makes sure that a user cannot assume the permissions of a researcher and a board member at the same time.

Besides some required plots, \{p1, p2\}, accessible to some basic roles (i.e. Employee, Guest) the application must also display a maximal set of plots, \{p3, p4, p5\}, accessible only to more privileged roles (i.e. Researcher, BoardMember). The intuition is the application should maximize the amount of information conveyed to the user, i.e. provide a coherent overview from different sources and combine them in a meaningful way, while complying with the authorization constraints.

This amounts to the following UAQ query:

\[ q = (s1, \{p1, p2\}, \{p1, p2, p3, p4, p5\}, \max) \]

Given the \( \max \) objective, the query prioritizes availability of the permissions over the safety of the system.

2 IMPLEMENTATION DETAILS

AQUA has been mostly implemented in C with various functionalities written in Java or implemented as shell scripts. The implementation of core architectural components (see Figure 1) constitutes of
some -2K lines of code. The source code of AQUA is freely available\(^1\) as part of a larger project presented in [2] and [3] along with further details on the performance. In what follows, we provide details about the implementation and demonstrate the use of AQUA over the running example.

### 2.1 Architecture

AQUA receives a UAQ instance and returns a set of roles. Figure 1 presents the overall architecture. There are four components:

- **Format Translator**: AQUA has a native input format for UAQ specifications, however it can also work with different formats. The component *Format Translator* is responsible for converting input specifications written in different formats to AQUA’s native format. The current implementation supports the notation used in [6].
- **Instance Parser**: The component *Instance Parser* parses the input specifications and generates an internal representation for the query and the accompanying RBAC model.
- **MaxSAT Encoder**: Given the internal representation, the component *MaxSAT Encoder* generates the corresponding MaxSAT formulae in WCNF format [4]. Among others, these formulae include clauses for RBAC relations (UA, PA), optimization objective and DMER constraints. Note that, AQUA employs the inequality encoding presented in [8] for efficiently handling DMER constraints, however it can also be used without this encoding.
- **Solver Interface**: AQUA can work with different MaxSAT solvers to tackle the final MaxSAT instance and the component *Solver Interface* enables the easy integration of MaxSAT solvers. This component is mainly for providing syntactical adjustments to the inputs sent to the solver, invokes the solver with the input and extracts the relevant information from its console output.

![Figure 1: AQUA Architecture](image)

**Figure 1: AQUA Architecture**

### 2.2 Execution

Before running AQUA, a few conditions need to be satisfied a priori:

- The selected MaxSAT solvers (AQUA supports various solvers natively including Loandra [5] and [1]) needs to be installed and the relevant environment variable needs to be set.
- If the cardinality encoding from [8] is used then the relevant executables need to be installed.

After these requirements are satisfied, AQUA can be invoked against a given UAQ specification (denoted as *UAQIns*) as follows:

\[ \text{./solve_uaq_smart.exe} \rightarrow \text{MaxSATSolver \textit{UAQIns}} \]

Assuming that the solver Loandra has been chosen, AQUA will show the roles (Figure 2) to be activated along with some additional information from the MaxSAT solver. It can also provide certain debug information related to errors in the input specification or misconfigurations regarding the MaxSAT solver integration.

![Figure 2: Output from AQUA](image)

**Figure 2: Output from AQUA**

As can be noted from output, the solution obtained from the solver (Line 5) and the permissions associated with the selected roles are also printed (Line 6).

### 3 CONCLUSIONS

We presented AQUA, a solver for the UAQ problem that implements a reduction of the UAQ Problem to Weighted MAX-SAT and can thus leverage state-of-the-art Weighted MaxSAT solvers taken off-the-shelf to efficiently solve UAQ problems. AQUA supports the enforcement of a wide class of DMER constraints and several types of optimization objectives. It is freely available and can be further extended with new optimization strategies and objectives. Moreover, thanks to the flexible architecture, new MaxSAT solvers can be easily integrated for efficiency purposes and/or better features.

### REFERENCES


