Cybersecurity MSc: Identification, Authentication and Authorization 2021-22

Project (1st part): Management of players' reputation with OAuth2.0

May 18, 2022 Due date: Jun 23, 2022

Changelog

- v1.1 - Changed the delivery date.
- v1.0 - Initial version.

1 Introduction

Nowadays, many Service Providers can have interest in providing services to end-users without having to know exactly who they are. Instead, they just need to know a few trustworthy attributes that are fundamental for providing the service, without know nothing else besides that. For that, they can resort to Identity Providers that feed them with those attributes upon end-users' access requests. And for that one can use the OAuth2.0 framework.

2 Homework

The work consists on build a system for managing and using the reputation of players in online table games. The system will include 4 entities:

- The User Agent (UA). The UA is used by the players to participate in online games. It allows players to participate in several types of games. Players should use a UA to interact with a Table Matchmaker in order to be selected for a match.

  For the sake of simplicity, you do not have to implement the games; you only have to start them and immediately reach a random outcome, where some win and some lose; therefore, you do not have to deal with game interfaces.

  For testing purposes, you can create bot players with autonomous UAs, which require minimal or no input at all to participate in many games.

- The Reputation Manager (RM). The RM is the keeper of the users' reputation relatively to online games. The RM keeps an entry for each registered user that reflects their past actions.

  The reputation will be used to organize games with other players, upon which the reputation is updated given the outcome of the game.

- The Tables' Matchmaker (TM). The TM is the entity responsible for organizing online games with candidate players. The TM does not know the identity of the players, only their reputation. Therefore, it must use only their reputation, and the players preferences, to decide which players it should join for a game that the players asked to play.

  We could have a TM for each type of game (e.g. one for chess, one for checkers/draughts, one for the Hearts card game, etc.) or one only for all of them. For simplicity, you can choose this last alternative.

- The Match Manager (MM). The MM is responsible for controlling the execution of a game match, avoiding and detecting cheating and determining the winner, the loser, or a ranking sequence (e.g. first the main winner, last the main loser). Furthermore, the MM will report players' abuses. All these outcomes should be conveyed to the TM.

  As referred before, games do not need to be implemented, so you can use random outcomes. In that process, you should include outcomes that include cheating an quitting.

  Since games are not effectively run, you can implement the MM as a part (function or module) of the TM.

2.1 The user's reputation

The users' reputation should contain two attributes per game type: skill and behaviour.

Skill should yield the quality of the user as a player. Highly skilled users are the ones that win the most, while low skilled users are the ones the usually loose.

Behavior should yield the attitude of a person as a player. A cheater or a quitter should have a lower behavior reputation, while honest and committed players should have a higher behavior reputation.

Both reputations indicators should be keep by the RM as a scalar value (integer or real numbers), starting at zero. The skill indicators should be increased upon wins and decreased upon losses. The behavior indicator should be increased upon a correct, finished match, and decreased upon being caught cheating or quitting a game. In this last case, the skill and behavior indicators of the other players in the same match are not updated.

Users' reputation on the RM are updated by a TM upon an outcome indication given by a MM. You can implement basic update policies (e.g. in a 4-player cards game you can give 1 point to the winner, 1 to the loser and 0 to the other two that stayed in between) or you can use some sophisticated calculation given the skill reputations (e.g. use the skill reputation of the players in the same match to adapt the wins and loses).

The reputation indicators maintained by the RM are never directly exposed to others than their owners. Besides those, reputation indicators are exposed to TMs in a coarse-grained form (e.g. level L out of N, with $L = 1$ being the highest one). This provides some anonymity to players, which can choose the N from a maximum allowed by the RM (e.g. 10). N defines the number of bins that were used to rank players; each bin must have a population as close as possible to P/N players, where P is the number of all registered players that have played before a given game.

Players can also chose their reputation preferences for guiding the TM in selecting their match peers. In particular, can indicate a preference for the skill reputations of them (e.g. I prefer to play against players with higher, equal or lower skills than me) and the behavior reputation (e.g. I prefer not to play against players behaving worse than me). For players using different values for N you should assume that a preference condition holds if there at least one possibility for it to be true. For instance, if player A with a skill reputation of 1/10 prefers to play with more skilled players, is is possible to match they with player B with a skill reputation of 1/3. And vice-versa. But not with C if its skill reputation is 2/3.

It should be possible for players not to indicate preferences regarding skill or behavior of others to play with.

2.2 Tables' Matchmaker as an OAuth2.0 client application

The TM if the central point that a player chooses to play a game. The player selects a game and the TM fetches the player's skill indicators for that game from the RM with the help of the player. That operation should use OAuth2.0; thus, in the OAuth2.0 model, the TM is a client application.

The reputation indicators must be collected for each game participation. A player asks the TM to play a given game, the TM collects their reputation indicators (with OAuth2.0), the player is placed in a

table managed by an MM, the match ends successfully or not, the MM reports the match outcome to the TM and the TM updates the players' skill indicators accordingly (also with OAuth2.0). In this cycle, the TM should be able to use exactly twice an access token for manipulating a player's reputation: once to get it, another to update it.

2.3 Reputation Manager (RM) authentication service

The RM must implement some sort of authentication for players, in order to allow them to authorize an OAuth2.0 client (the TM) to get an authorization grant for fetching an access token for accessing a user's player reputation. For this project, implement a basic username / password based protocol; it will be changed in the 2nd part of this project.

Also, since the TM will need an authorization grant to each game request of each player, implement some sort of session for players, in order to reduce the number of required user authentications. If using a browser to support the UA execution, you can use HTTP cookies for that.

For the sake of simplicity, do not make special arrangements to protect the authentication exchange or the session exploitation. Namely, do not use HTTPS for interacting with the RM (it facilitates debugging). Such protection will be defined and implemented in the following 2nd part of this project.

3 Project development and delivery

This project is to be implemented by groups of 2 students. The project can be coded in any language but must use OAuth2.0.

Send your code to the course teachers through Elearning (a submission link will be provided). Include a report, with no more than 30 pages, describing the system implemented. Such description must include the data structures stored, the structure of the messages exchanged and the message flows, the interfaces used and their parameters, some relevant implementation details (not complete copies of the code!) and the results achieved.

In order to facilitate the testing of your system, create bots to play with.

The report must state the percentage of effort devoted by each group member to the project.

4 Evaluation

This 1st part of the project will be evaluated as follows:

- OAuth2.0 setup and use: 30%;
- RM design and implementation: 20%;
- TM and MM design and implementation: 20%;
- UA design and implementation: 10%;
- Written report, with complete explanations of the strategies followed and the results achieved: 20%