Security Target for Online Game Architecture

David Eskilsson
Linköping University
daves801@student.liu.se

Abstract

This document is a security target as specified by the Common Criteria, aimed to describe the target of evaluation in terms of a general description as well as assumptions about the environment and possible perceived threats. From this information is derived certain security objectives which are then further refined to security requirements. Thereafter, a set of assurance requirements are outlined, which together comprise a specific assurance level. Finally, a rationale is included to show that the threats outlined have been met by the objectives, and that each objective is refined to the appropriate set of requirements.

1. Introduction

This section contains information about the document, such as an overview of the contents, a note on the organisation of the document, and a list of the limitations imposed on the report due to time constraints.

1.1. Identification

Title: Security Target for Online Game Architecture
Keywords: online game architecture security, client-server structure

1.2. Overview

The Common Criteria (CC) specify a set of functional and assurance requirements which help to find and address security issues in a development project. The Security Target for Online Game Architecture (hereafter referred to as STOGA) outlines the perceived threats and security requirements on a client-server game architecture under development, and aims to outline these in such a way that a reasonably secure system can be developed.

1.3. Organisation

This document is developed as part of the course TDDC03 Information Security, given at Linköping University. It is not intended to have any commercial application whatsoever.

The purpose of writing this document has been to give the author some insight into the application of the Common Criteria, and as such, the document should under no circumstances be used as any kind of reference for future work.

1.4. Limitations

Due to time constraints in the course given, certain parts have been left out, or have not been fully developed. Most notably, the chapters TOE Summary Specification and PP Claims have been left out due to time constraints, and not all applicable assumptions, threats, objectives and requirements have been included, again due to time constraints in the course.

2. Target of evaluation description

The following section describes the general functionality and usage of the target of evaluation (TOE).

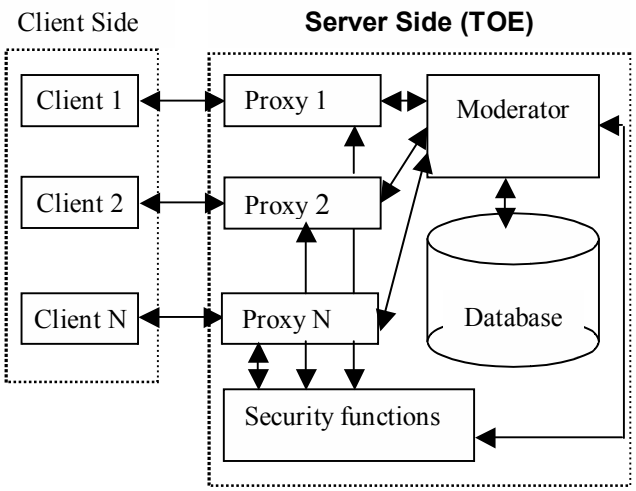

Figure 2.1 - General architecture of the TOE

2.1. General description

The TOE described is a semi-massively multiplayer online strategy game architecture, meant to be used across some kind of wide network, primarily the Internet (although intranets are certainly an alternative). It is implemented in the programming language Java.

The product is implemented as a client/server solution where the server can host a number of sessions (hereafter referred to as *games*), and each of those is associated with a number of players through clients (there is a one-to-one mapping between players and clients). The main point of this architecture is then to allow a player to play his or her turn at their leisure, and if an action should be required of the player when he/she is not present, some kind of default action will be taken.

The general architecture of the system can be seen in figure 2.1.

2.1.1. The server. The server part of the system has three major components, which are described below.

The database - The database is the information repository for the system, and contains all actual game-related information, which has been created in a modular fashion and is almost completely interchangeable and independent of the architecture implementation. This component is only operated on by the moderator (see below), and as such is a very separate part of the system, for security purposes.

The moderator - The moderator is the heart of the system execution, and has mainly two functions:

1. Establish and maintain a connection to the database, and perform database access functions. It also updates the database according to specific game rules on a regular basis.
2. Send information and receive input to and from the player proxy classes, to be incorporated in the database.

Player proxy - Each client has a player proxy assigned to it, which has two functions:

1. Establish and maintain an RMI connection to the client to which it is assigned, and handle any problems and issues which arise from this.
2. Incorporate an artificial intelligence function which determines the course of action if one should be requested or required while the client to which this proxy is assigned is not online or active.

2.1.2. The client. The client uses a fairly thin architecture, which is comprised of a fairly complex graphical user interface and a module which handles the connection to the server. Put simply, the client only has two functions: To display the information received from the server in a way which is visually appealing to the player and to make available actions to the player and refer these to the server. The client is not part of the TOE, and as such is only included for reference purposes.

3. TOE security environment

This section outlines the assumptions made about the TOE environment for security purposes. It also specifies the assets which require protection in the TOE, as well as any threats to these that have been identified. Furthermore, the different organisational policies which have to be enforced are specified.

3.1. Assumptions

The assumptions made about the security environment are listed below.

3.1.1. Physical assumptions. These assumptions relate to direct physical environment and operation of the system.

A.1.1 The software which comprises the server part of the TOE and the hardware it executes on are assumed to be located within controlled access facilities which will prevent unauthorised physical access.

A.1.2 The software critical to the server side secure execution of the TOE is assumed to be physically protected from potentially hostile *outsiders*.

3.1.2. Personnel assumptions. These assumptions relate to the personnel managing the TOE.

A.2.1 It is assumed that there is *at least one* administrator of the TOE. This person is assumed to have a high level of knowledge of the system on which the TOE is run and be completely trusted not to abuse his/her privileges.

A.2.2 Attackers are assumed to be willing only to spend time and/or money in proportion to the possible gain of an attack (see A.3.1 below).

3.1.3. Information assumptions. These assumptions relate to the kind of information which is stored in the TOE.

A.3.1 The data stored in the system, due to its nature, whether inside or outside of the database repository, is assumed to be of relatively low monetary value.

A.3.2 The database and its security functions are assumed to be part of the TCB, and as such it is assumed that any data entered in the database are for all intents and purposes secure.

3.1.4. Connectivity assumptions. These assumptions relate to the network and connectivity aspects of the TOE.

A.4.1 All connections to peripheral devices used by the server side of the TOE are assumed to reside within the controlled and secure access facilities (see A.1.1 above).

A.4.2 The connection between the server and the client(s) in the TOE is assumed to be the only connection open to the area outside the controlled and secure access facilities (See A1.1 and A.4.1 above).

3.2. Threats

The perceived threats to the TOE are listed below, sorted by security aspect compromised (confidentiality, integrity and availability).

3.2.1. Threats to confidentiality. Outlined below are the threats to the confidentiality aspect of the TOE.

T.1.1 A compromise of the personal or game related information stored in the database part of the TOE may result from a player performing actions he/she is authorised to perform, intentionally or not.

T.1.2 A player may access information (game related or otherwise) without having permission from the owner of said information.

T.1.3 A player may, intentionally or accidentally, observe game related and other information that he/she is not cleared to see.
Example: A player learns – through direct observation of results or other methods – of rules for the game mechanics, giving the player an unfair advantage.

T.1.4 An unauthorised entity may eavesdrop on, or otherwise overhear, information which may be sensitive to one or more players.
Example: By intercepting packages being sent across the network, an attacker may discover information about a player that should be kept secret.

3.2.2. Threats to integrity. Outlined below are the threats to the integrity aspect of the TOE.

T.2.1 The integrity of the information stored and being changed in the database may be compromised if a player connects to the TOE at an inappropriate time.
Example: A player connects to the server while the moderator is applying game rules to the database, and sends actions to the server, which in turn interferes with the moderation of the database.

T.2.2 The information being sent between the client and the player proxy – in either direction – may be altered by an outside entity.
Example: A person may intercept a package being sent, and either prevent it from arriving at its destination, corrupt the contents or change the contents so that the client-server communication is disrupted.

T.2.3 The system will use *at least once* semantics, ie. any information sent between client and server will always arrive *at least once*, but possibly more than once. This creates the problem of detecting duplicate actions, so that the integrity of the intent of the player is not jeapardised.
Example: A player performs the action of giving another player a certain amount of resources. Due to poor connectivity, the action is resent several times, and as a result arrives several times at the server, and the server interprets this as the player giving a larger sum of resources to the other player.

T.2.4 An entity may gain access to a player's account, either by hacking into the TOE, or by somehow gaining access to the player's login information.
Note: This is a threat to confidentiality, integrity *and* availability, but is placed here due to the integrity aspect being considered the most important.
Example: A person steals the password for a certain player and logs into the system with that information, and proceeds to sabotage for that player.

T.2.5 The TOE might suffer from some kind of general fault or crash, either as the result of an attack, or through a bug or accident, which would be a threat to both integrity and availability, the former being considered more important.
Example: A hacker makes the system halt through some kind of attack.

3.2.2. Threats to availability. Outlined below are the threats to the availability aspect of the TOE.

T.3.1 A player, through his/her client, consumes network bandwidth, processor power and memory from the server when being connected, which can compromise the ability of other player clients connecting to and performing actions on the server.

T.3.2 An entity may use some kind of Denial of Service attack (DoS) to compromise the availability of the server to other players.

Example: An ill-meaning person might create a program which repeatedly pings or other sends other messages to the server, to such an extent that the server becomes unavailable to authorized players.

T.3.3 The database will be using some kind of locking scheme to assure that concurrent access to the data is handled in a proper way. This can compromise the availability of the data for write and read operations.
Example: A player may be unable to perform actions while the server is updating the data in the database according to the rules.

3.3. Organisational security policies

The organisational security policies which have been outlined for the TOE are listed below.

P.1.1 Only those users who have been authorised to access the TOE may do so.

P.1.2 There will be at least one person who will assume the role of administrator, and have the privileges to manage the TOE and all player information (see assumption A.2.1 above).

4. Security objectives

This section outlines the intended response to the security problem outlined in the previous chapter (see chapter 3 – TOE security environment). It is intended to bridge the gap between the assumptions and threats outlined in the previous chapter, and the formal requirements outlined in chapter 5.

4.1. Security objectives for the TOE

These security objectives specify what measures are taken by the TOE to counter the threats specified in chapter 3.2, taking into account the assumptions of chapter 3.1. The objectives identified are divided into three broad categories (preventative, detective and corrective).

4.1.1. Preventative objectives. The following objectives are such that aim to prevent or limit the ways in which a threat can be carried out.

O.1.1 The TOE will make available functions such that an administrator can configure and manage the server, database and the security functions, and will also ensure that only the authorised administrator can access those functions.

O.1.2 The TOE will uniquely identify all clients connecting, and will authenticate the claimed identity before granting the client access to the TOE.

O.1.3 The TOE will encrypt the information being sent to the client, to make it more difficult for unauthorised entities to intercept and modify said information.

O.1.4 The TOE will only accept a certain number of connection requests from one source per minute.

O.1.5 The TOE will maintain a pseudonym for each registered player, and will only make available references by that pseudonym to other players. The personal information of all players will be hidden from all but the administrator.

O.1.6 The TOE will implement mechanisms for detection and handling of duplicated messages received.

O.1.7 The TOE will assign different priorities to different actions on behalf of the players, such that – in case of resource shortage – players can still perform their most important actions.

O.1.8 The TOE will ensure that a given user can only be connected through *one* session at a time. The player will also automatically be logged out after a certain amount of time has passed.

O.1.9 The TOE will enforce certain access and information policies that will make it more difficult or impossible for players to gain more information than the designer of the game rules intend.

4.1.2. Detective objectives. These objectives aim to provide means to detect and monitor the events involved in a threat.

O.2.1 The TOE will create a log of the events sent to and from the client program connected to each server proxy, which will be stored for a certain amount of time on the server. This log will be accessible only to the administrator of the TOE.

O.2.2 The TOE will make available information to the player about any failed logins since the last successful login, and also information about where the last successful login was made from.

4.1.3. Corrective objectives. The following objectives provide the TOE with the ability to return to a safe state in the case of damage inflicted in connection to a threat.

O.3.1 The TOE will make available functions to return the information stored in the database to a previous state, without ending up in an inconsistent state.

O.3.2 Upon restart of the TOE after a malfunction, the TOE will aim to return to a functioning state, but failing that, the TOE will enter an administrative

state where the administrator can find the error and correct it.

4.2. Security objectives for the environment

These security objectives are such objectives that are to be satisifed by the environment of the TOE. These objectives aim to counter the threats which are not addressed by the TOE. Due to time constraints, no such security objectives have been formulated.

5. IT security requirements

This chapter specifies the security requirements for the TOE, as taken from the Common Criteria. Certain operations are defined in the Common Criteria, and completed such operations are denoted through italicized text.

5.1. TOE security functional requirements

Below follow the functional requirements for the TOE, sorted by the requirement type.

5.1.1. Security audit (FAU).

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions.

b) All auditable events for the *not specified* level of audit; and

c) The events specified in table 5.1.

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the ST, *the information in the appropriate details column.*

Table 5.1. Auditable events.

Component	Event for audit
FAU_GEN.1	Start-up and shutdown of the audit functions.
FAU_GEN.2	None.
FAU_SAR.1	None.
FDP_ACC.1	None.
FDP_ACF.1	None.
FDP_IFC.1	None.
FDP_IFF.1	None.
FDP_ROL.2	The rollback of the database.
FIA_AFL.1	The failure to login.
FIA_ATD.1	None.
FIA_UAU.1	None.
FIA_UAU.7	None.
FIA_UID.1	None.
FIA_USB.1	None.
FMT_MOF.1	Any of the specified actions.
FMT_MSA.1	Any changes to the attributes.
FMT_MSA.3	Any change to the default values.
FMT_MTD.1	None.
FMT_SMR.1	None.
FPR_PSE.2	None.
FPT_AMT.1	The tests being run.
FPT_ITT.1	None.
FPT_RCV.2	Restart of the system or entrance into maintenance mode.
FPT_RPL.1	The event being removed.
FPT_STM.1	Changes to the time.
FPT_TST.1	The tests being run.
FRU_PRS.1	None.
FRU_RSA.1	Any attempts to exceed the quota.
FTA_MCS.1	Any attempts to exceed the session limit
FTA_SSL.3	Timeouts.
FTA_TAH.1	None.

FAU_GEN.2.1 The TSF shall be able to associate each auditable event with the identity of the user that caused the event.

FAU_SAR.1.1 The TSF shall provide *the administrator* with the capability to read *all information* from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

5.1.2. User data protection (FDP).

FDP_ACC.1.1 The TSF shall enforce *the access control policy* on *the clients, database and any defined client actions pertaining to the database*.

FDP_ACF.1.1 The TSF shall enforce the *access control policy* to objects based on *the identified and authorised identity of the client in question.*

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: *The subject must be identified and authorised and the operation must be made according to the rules specified in the game system.*

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: *No additional rules.*

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the *failure of the subject to identify and/or authorise itself.*

FDP_IFC.1.1 The TSF shall enforce *the information flow control policy* on *the clients, the information repository (database) and any information flow between the database and clients.*

FDP_IFF.1.1 The TSF shall enforce the *information flow control policy* based on the following types of subject and information security attributes: *The information flow control policy must at least*

a) function in conjunction with the access control policy
b) be able to determine the rights of subjects to access certain kinds of information, as given in the specified game rules
c) allow for the identification of information flow sources

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: *The identity of the subject must be known and confirmed.*

FDP_IFF.1.3 The TSF shall enforce the *information flow control policy in that information flow between the TOE and any trusted clients takes precedence over any information flow between the TOE and untrusted clients.*

FDP_IFF.1.4 The TSF shall provide the *ability to specify the specific rules for information flow control in addition to those stated above as required by the specific implemented game rules.*

FDP_IFF.1.5 The TSF shall explicitly authorise an information flow based on the following rules: *An administrator connecting to the TOE through some kind of trusted path.*

FDP_IFF.1.6 The TSF shall explicitly deny an information flow based on the following rules: *in accordance with the security policy setting to deny information flow to and from an untrusted client.*

FDP_ROL.2.1 The TSF shall enforce *the access control policy and the information control flow policy* to permit the rollback of all the operations on the *database*.

FDP_ROL.2.2 The TSF shall permit operations to be rolled back within the *last week of operations performed.*

5.1.3. Identification and authentication (FIA).

FIA_AFL.1.1 The TSF shall detect when *3* unsuccessful authentication attempts occur related to *the initial client-server authentication procedure.*

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall *deny further login attempts from that address for a configurable amount of time.*

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:

a) User identifier
b) Authentication data

FIA_UAU.1.1 The TSF shall allow *invocation of server status information services* on behalf of the user before the user is authenticated.

Note: Server status information services are functions which return the current status of the server (such as whether it is accepting login requests or not).

FIA_UAU.1.2 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

FIA_UAU.7.1 The TSF shall provide only *obscured feedback* to the user while the authentication is in progress.

Note: Obscured feedback means that the user is provided no visible feedback while entering the required authentication information.

FIA_UID.1.1 The TSF shall allow *invocation of server status information services* on behalf of the user to be performed before the user is identified.

Note: Server status information services are functions which return the current status of the server (such as whether it is accepting login requests or not).

FIA_UID.1.2 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

FIA_USB.1.1 The TSF shall associate the appropriate user security attributes with subjects acting on behalf of that user.

Note: The subject mentioned above is the server proxy associated with the user.

5.1.4. Security management (FMT).

FMT_MOF.1.1 The TSF shall restrict the ability to *determine the behaviour of* the functions

a) *Deleting user accounts*
b) *Arbitrary modification of accounts*
c) *Configuring the server options*
d) *Modifying the game state*

to *the administrator.*

FMT_MSA.1.1 The TSF shall enforce the *access control policy and information flow control policy* to restrict the ability to *modify* the security attributes *user identity and identification information* to *the user in question and the administrator.*

FMT_MSA.3.1 The TSF shall enforce the *access control policy and information flow control policy* to provide *permissive* default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the *administrator* to specify alternative initial values to override the default values when an object or information is created.

FMT_MTD.1.1 The TSF shall restrict the ability to *query and modify* the *the user information* to *the administrator and to the player in possession of said information.*

FMT_SMR.1.1 The TSF shall maintain the roles

a) *Player*
b) *Administrator*

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

5.1.5. Privacy (FPR).

FPR_PSE.2.1 The TSF shall ensure that *the players* are unable to determine the real user name bound to *the players and the administrators.*

FPR_PSE.2.2 The TSF shall be able to provide *one* alias of the real user name to *the players.*

FPR_PSE.2.3 The TSF shall *accept the alias from the user* and verify that it conforms to the *specified rules governing names and aliases for the actual game system.*

FPR_PSE.2.4 The TSF shall provide *the administrator* a capability to determine the user identity based on the provided alias only *for communication purposes in case of any need for external communication.*

5.1.6. Protection of the TSF (FPT).

FPT_AMT.1.1 The TSF shall run a suite of tests *at the request of an authorised user* to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

FPT_ITT.1.1 The TSF shall protect TSF data from *disclosure and modification* when it is transmitted between separate parts of the TOE.

FPT_RCV.2.1 When automated recovery from a failure or service disconuity is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

FPT_RCV.2.2 For *network failures and certain software failures*, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RPL.1.1 The TSF shall detect replay for the following entities: *Players*.

FPT_RPL.1.2 The TSF shall perform *removal of the last part of replay* when replay is detected.

FPT_STM.1.1 The TSF shall be able to provide reliable time stamps for its own use.

FPT_TST.1.1 The TSF shall run a suite of self tests *periodically during normal operation* to

demonstrate the correct operation of the TSF.

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of the TSF data.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of stored executable code.

5.1.7. Resource utilisation (FRU).

FRU_PRS.1.1 The TSF shall assign a priority to each subject in the TSF.

FRU_PRS.1.2 The TSF shall ensure that each access to *the database* shall be mediated on the basis of the subjects assigned priority.

FRU_RSA.1.1 The TSF shall enforce maximum quotas of the following resources:

a) bandwidth

b) database access

that *players* can use *simultaneously*.

5.1.8. TOE access (FTA).

FTA_MCS.1.1 The TSF shall restrict the maximum number of concurrent sessions that belong to the same user.

FTA_MCS.1.2 The TSF shall enforce, by default, a limit of *one* session per user.

FTA_SSL.3.1 The TSF shall terminate an interactive session after a *30 minute period of user inactivity*.

FTA_TAH.1.1 Upon successful session establishment, the TSF shall display the *date, time and location* of the last successful session establishment to the user.

FTA_TAH.1.2 Upon successful session establishment, the TSF shall display the *date, time and location* of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment.

FTA_TAH.1.3 The TSF shall not erase the access history information from the user interface without giving the user an opportunity to review the information.

Note: The actual user interface is not actually within the control of the TOE, and therefore this last capability is not particularly relevant.

5.2. TOE security assurance requirements

Below follow the assurance requirements for the TOE, sorted by the requirement type. Requirements ending with D are aimed at the developer, those ending with C refer to the actual document in question, and the ones ending with E are aimed at the evaluator.

The assurance requirements below compose EAL2.

5.2.1. Configuration management (ACM).

ACM_CAP.2.1D The developer shall provide a reference for the TOE.

ACM_CAP.2.2D The developer shall use a CM system.

ACM_CAP.2.3D The developer shall provide CM documentation.

ACM_CAP.2.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.2.2C The TOE shall be labelled with its reference.

ACM_CAP.2.3C The CM documentation shall include a configuration list.

ACM_CAP.2.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.2.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.2.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2. Delivery and operation (ADO).

ADO_DEL.1.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.1.2D The developer shall use the delivery procedures.

ADO_DEL.1.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.1D The developer shall document procedures necessary for the secure installation, generation and start-up of the TOE.

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation and start-up of the TOE.

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.2E The evaluator shall determine that the installation, generation and start-up procedures result in a secure configuration.

5.2.2. Development (ADV).

ADV_FSP.1.1D The developer shall provide a functional specification.

ADV_FSP.1.1C The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.1.2C The functional specification shall be internally consistent.

ADV_FSP.1.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate.

ADV_FSP.1.4C The functional specification shall completely represent the TSF.

ADV_FSP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

ADV_HLD.1.1D The developer shall provide the high-level design of the TSF.

ADV_HLD.1.1C The presentation of the high-level design shall be informal.

ADV_HLD.1.2C The high-level design shall be internally consistent.

ADV_HLD.1.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.1.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.1.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware or software.

ADV_HLD.1.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.1.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.1.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3. Guidance documents (AGD).

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the control of the administrator,

	indicating secure values as appropriate.
AGD_ADM.1.6C	The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.
AGD_ADM.1.7C	The administrator guidance shall be consistent with all other documentation supplied for evaluation.
AGD_ADM.1.8C	The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.
AGD_ADM.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AGD_USR.1.1D	The developer shall provide user guidance.
AGD_USR.1.1C	The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.
AGD_USR.1.2C	The user guidance shall describe the use of user-accessible security functions provided by the TOE.
AGD_USR.1.3C	The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.
AGD_USR.1.4C	The user guidance shall clearly present all user responsibilities necessary for the secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.
AGD_USR.1.5C	The user guidance shall be consistent with all other documentation supplied for evaluation.
AGD_USR.1.6C	The user guidance shall describe all security requirements for the IT environment that are relevant to the user.
AGD_USR.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4. Tests (ATE).

ATE_COV.1.1D	The developer shall provide evidence of the test coverage.
ATE_COV.1.1C	The evidence of the test coverage shall show the correspondence between the test identified in the test documentation and the TSF as described in the functional specification.
ATE_COV.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ATE_FUN.1.1D	The developer shall test the TSF and document the results.
ATE_FUN.1.2D	The developer shall provide test documentation.
ATE_FUN.1.1C	The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.
ATE_FUN.1.2C	The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.
ATE_FUN.1.3C	The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.
ATE_FUN.1.4C	The expected test results shall show the anticipated outputs from a successful execution of the tests.
ATE_FUN.1.5C	The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.
ATE_FUN.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.2.1D	The developer shall provide the TOE for testing.
ATE_IND.2.1C	The TOE shall be suitable for testing.
ATE_IND.2.2C	The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.
ATE_IND.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.2.6. Vulnerability assessment (AVA).

AVA_SOF.1.1D The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

AVA_SOF.1.1C For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the ST.

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the ST.

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that the strength claims are correct.

AVA_VLA.1.1D The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

AVA_VLA.1.2D The developer shall document the disposition of obvious vulnerabilities.

AVA_VLA.1.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.1.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

5.3. Security requirements for the environment

No environmental security requirements have at this time been identified.

6. TOE summary specification

Due to time constraints and the lack of progress of the implemented system, the TOE summary specification will be presented in a later project.

7. Rationale

This chapter aims to show that:

a) All security needs, as defined by the threats and organisational policies, are dealt with through the use of appropriate security objectives.

b) All security objectives are suitably met by the identified IT security requirements, which in turn are met by the specified functional and assurance requirements.

7.1. Security objectives rationale

This section maps threats to security objectives, and states why the outlined security objectives address the issues specified in the threats in an appropriate way. For the actual mapping, see table 7.1.

Table 7.1. Threats to objectives mapping.

Threat/policy	Security Objective
T.1.1	O.1.5
T.1.2	O.1.5
T.1.3	O.1.9
T.1.4	O.1.3
T.2.1	O.2.1 O.3.1
T.2.2	O.1.3
T.2.3	O.1.6
T.2.4	O.2.1 O.2.2 O.3.1
T.2.5	O.3.2
T.3.1	O.1.4 O.1.7 O.1.8
T.3.2	O.1.4
T.3.3	O.1.7
P.1.1	O.1.2
P.1.2	O.1.1

Below follows a more specific discussion about the relevance of each objective to address the issues specified in the threats and policies.

T.1.1
A compromise of the personal or game related information stored in the database part of the TOE may result from a player performing actions he/she is authorised to perform, intentionally or not.
Due to each player being assigned and referenced through a pseudonym (O.1.5), it is very difficult for another player to partake of user information he/she should not partake in, if the inter-TOE communication channels are followed, as stated in the threat.

T.1.2
A player may access information (game related or otherwise) without having permission from the player possessing said information.
Due to each player being assigned and referenced through a pseudonym (O.1.5), it is not trivial to gain access to personal information, even when accessing information through some extra-TOE channel.

T.1.3
A player may, intentionally or accidentally, observe game related and other information that he/she is not cleared to see.
The implementation of access and information flow policies (O.1.9) ensures that players logged into the system at least through legitimate means cannot gain access to more information than necessary.

T.1.4
An unauthorised entity may eavesdrop on, or otherwise overhear, information which may be sensitive to one or more players.
Encryption of the messages sent to the client (O.1.3) will serve to address the problem of entities eavesdropping on the information, and will most certainly make sure that such eavesdropping, if successful, is not accidental. Since the client is not part of the TOE, the security of the information being sent from the client to the server cannot be assured, but this is not particularly important for confidentiality reasons.

T.2.1
The integrity of the information stored and being changed in the database may be compromised if a player connects to the TOE at an inappropriate time.
The logs created by the TOE of any events sent (O.2.1) will detect any cause of error in the above case. If this should occur, this information can be used to roll back the state of the database to a previous consistent state (O.3.1), losing only some of the information stored.

T.2.2
The information being sent between the client and the player proxy – in either direction – may be altered by an outside entity.
As T.1.4 above, encryption of the messages sent to the client (O.1.3) will serve to address the problem of entities changing the information being sent. The problem remains for messages originating from the client, however, and this can cause a problem since the integrity of the messages travelling in this direction is the most important aspect to consider. However, since the client is not part of the TOE, this cannot be addressed through security functions in the TOE.

T.2.3
The system will use at least once semantics, ie. any information sent between client and server will always arrive at least once, but possibly more than once. This creates the problem of detecting duplicate actions, so that the integrity of the intent of the player is not jeapardised.
The objective to detect and handle any duplicate messages received directly takes care of this threat.

T.2.4
An entity may gain access to a player's account, either by hacking into the TOE, or by somehow gaining access to the player's login information.
Logging any events sent from a client to the server (O.2.1) helps detect the occurence of something like this. Furthermore, displaying information about the recent failed login attempts, and information about the last successful login (O.2.2) makes it possible for the player to see whether or not someone has been trying to guess his/her password, and/or actually succeeded. Finally, the TOE will make available – as a last resort – to return the state of the system to a previous consistent one as an administrator function (O.3.1).

T.2.5
The TOE might suffer from some kind of general fault or crash, either as the result of an attack, or through a bug or accident, which would be a threat to both integrity and availability, the former being considered more important.
The TOE will aim to restart itself automatically after crashing, or failing that, enter an administrative state where the administrator can try to find the error and return the server to normal functionality (O.3.2).

T.3.1
A player, through his/her client, consumes network bandwidth, processor power and memory from the server when being connected, which can compromise the ability of other player client connecting to and performing actions on the server.

The objective specifying that a client may only try to connect a certain number of times per time period (O.1.4) takes care of part of this problem. Moreover, the limitations specified about the resources available to players (O.1.7) and the limitation to one session per client (O.1.8) serve to further address this threat.

T.3.2
An entity may use some kind of Denial of Service attack (DoA) to compromise the availability of the server to other players.
Due to the nature of this attack, this threat is much more difficult to counter than T.3.1 above. However, objective O.1.4 (see above) helps counter this to some extent.

T.3.3
The database will be using some kind of locking scheme to assure that concurrent access to the data is handled in a proper way. This can compromise the availability of the data for write and read operations.
Considering that the database will not be locked for too long, the objective to assign priorities and perform actions based on those (O.1.7) should suffice to counter this threat.

P.1.1
Only those users who have been authorised to access the TOE may do so.
The objective specifying that all clients connecting must be identified and authorised (O.1.2) makes sure that the policy is satisfied.

P.1.2
There will be at least one person who will assume the role of administrator, and have the privileges to manage the TOE and all player information.
The functions made available through objective O.1.1 ensures that the policy is satisfied.

7.2. Security requirements rationale

This section maps objectives to requirements, and states why the outlined functional and assurance requirements help reach the objectives outlined, and by extension address the perceived threats and organisational policies. For the actual mapping, see table 7.2.

Table 7.2. Objectives to requirements mapping.

Security Objective	Functional/assurance requirement
O.1.1	FMT_MOF.1. Management of security functions behaviour FMT_SMR.1. Security roles
O.1.2	FIA_ATD.1. User attribute definition FIA_UAU.1. Timing of authentication FIA_UAU.7. Protected authentication feedback FIA_UID.1. Timing of identification FIA_USB.1. User-subject binding FMT_MTD.1. Management of TSF data
O.1.3	To be determined.
O.1.4	FIA_AFL.1. Authentication failures
O.1.5	FPR_PSE.2. Reversible pseudonymity
O.1.6	FPT_RPL.1. Replay detection FPT_STM.1. Reliable time stamps
O.1.7	FRU_PRS.1. Limited priority of service FRU_RSA.1. Maximum quotas
O.1.8	FTA_MCS.1. Basic limitation on multiple concurrent sessions FTA_SSL.3. TSF-initiated termination
O.1.9	FDP_ACF.1. Security based access control FDP_IFF.1. Simple security attributes
O.2.1	FAU_GEN.1. Audit data generation FAU_GEN.2. User identity association FAU_SAR.1. Audit review
O.2.2	FTA_TAH.1. TOE access history
O.3.1	FDP_ROL.2. Advanced rollback FPT_STM.1. Reliable time stamps
O.3.2	FPT_RCV.2. Automated recovery

Below follows a more specific discussion about the relevance of each requirement to enforce the objectives.

O.1.1
The TOE will make available functions such that an administrator can configure and manage the server, database and the security functions, and will also ensure that only the authorised administrator can access those functions.
Through the use of security roles (FMT_SMR.1) it is established that there is one role called *the administrator* and one called *user*. Furthermore, the requirement about management of security functions behavior (FMT_MOF.1) specifies that certain security-relevant functions are restricted such that only the administrator may use them.

O.1.2
The TOE will uniquely identify all clients connecting, and will authenticate the claimed identity before granting the client access to the TOE.
The user attribute definition (FIA_ATD.1) specifies which security attributes exist for each user, and the timing of authentication (FIA_UAU.1) and timing of identification (FIA_UID.1) require the user to identify and authorise himself before he attempts any action other than requesting information about the server status. The protected authentication feedback (FIA_UAU.7) makes sure that no feedback is given to the client while the authorisation is in progress. Finally, user-subject binding (FIA_USB.1) is used to bind the client to the proxy, which can then act on the user's behalf, and it is specified through the management of TSF data that only the administrator and the player in possession of the data can access the user information and authorisation data.

O.1.3
The TOE will encrypt the information being sent to the client, to make it more difficult for unauthorised entities to intercept and modify said information.
The encryption algorithms and key lengths to be used have not yet been determined, and as such, these functional requirements are lacking.

O.1.4
The TOE will only accept a certain number of connection requests from one source per minute.
The requirement about authentication failures (FIA_AFL.1) takes care of this by making sure that the server will only accept a certain number of requests before refusing further connection requests from that source for a period of time.

O.1.5
The TOE will maintain a pseudonym for each registered player, and will only make available references by that pseudonym to other players. The personal information of all players will be hidden from all but the administrator.
Reversible pseudonymity (FPR_PSE.2) makes sure that the player specifies an alias that he/she will go by, to make sure that all personal information belonging to him/her cannot be accessed by any other entity save by the administrator.

O.1.6
The TOE will implement mechanisms for detection and handling of duplicated messages received.
The requirement about replay detection (FPT_RPL.1) makes sure that any duplicated messages received by the server can be identified and addressed in a correct fashion. Reliable time stamps (FPT_STM.1) helps to determine which messages are actually replays and which are separate messages.

O.1.7
The TOE will assign different priorities to different actions on behalf of the players, such that – in case of resource shortage – players can still perform their most important actions.
Limited priority of service (FRU_PRS.1) makes sure that no player can "hog" all resources for himself. The maximum quotas (FRU_RSA.1) are intented to support this by assigning a configurable amount of resources available to each player.

O.1.8
The TOE will ensure that a given user can only be connected through one session at a time. The player will also automatically be logged out after a certain amount of time has passed.
The requirement about basic limitation on multiple concurrent sessions (FTA_MCS.1) states that only one session is allowed for each user at a time. The TSF-initiated termination (FTA_SSL.3) makes sure that if the player remains inactive for a certain amount of time, he/she is automatically logged out of the system.

O.1.9
The TOE will enforce certain access and information policies that will make it more difficult or impossible for players to gain more information than the designer of the game rules intend.
The security attribute based access control (FDP_ACF.1) and simple security attributes (FDP_IFF.1) ensure that player accesses to the database are mediated by a certain policy, and that the information flow between players and the database are regulated through a customizable policy.

O.2.1
The TOE will create a log of the events sent to and from the client program connected to each server proxy, which will be stored for a certain amount of time on the server. This log will be accessible only to the administrator of the TOE.
The audit data generation (FAU_GEN.1) and user identity association (FAU_GEN.2) make sure that the relevant events are logged. Through the audit review requirement (FAU_SAR.1), it is ensured that only the administrator has access to the audit information.

O.2.2
The TOE will make available information to the player about any failed logins since the last successful login, and also information about where the last successful login was made from.

The TOE access history requirement (FTA_TAH.1) makes sure that the server makes available information about the login history of the player, most importantly any failed logins and the last successful login.

O.3.1
The TOE will make available functions to return the information stored in the database to a previous state, without ending up in an inconsistent state.
Through the use of advanced rollback (FDP_ROL.2), the database can be restored to a previous state in case of error or sabotage. Reliable time stamps (FPT_STM.1) is necessary to ensure that the rollback can be done in a safe and correct fashion.

O.3.2
Upon restart of the TOE after a malfunction, the TOE will aim to return to a functioning state, but failing that, the TOE will enter an administrative state where the administrator can find the error and correct it.
Automated recovery (FPT_RCV.2) makes it possible for the system to enter an administrative state in case it cannot restart properly after an error.

7.3. Assurance requirements rationale

EAL2 was chosen to provide a low to moderate level of independently assured security since the value of data stored in the system combined with the likelihood of data loss and/or revelation only motivates such security.

8. PP claims

No PP claims have been made.