Report Assignment 5 - Threat Modeling and Abuse Cases

Contributors

Name	Student no.	Track
Jonas J. Solsvik	473193	16HBPROGA
Kent Wincent Holt	473209	16HBPROGA
Eldar	473180	16HBPROGA

Table of Contents

Task a)

Text: Study the application very well, and try to understand its architecture (components and relationships) and its behaviour. Use suitable diagrams to model the application (not Data Flow Diagram).

We are studying the following github repository:

```
https://github.com/NetEase/pomelo
```

Size of code base

Getting overview of the complexity of the codebase by using the tool CLOC to count lines of code:

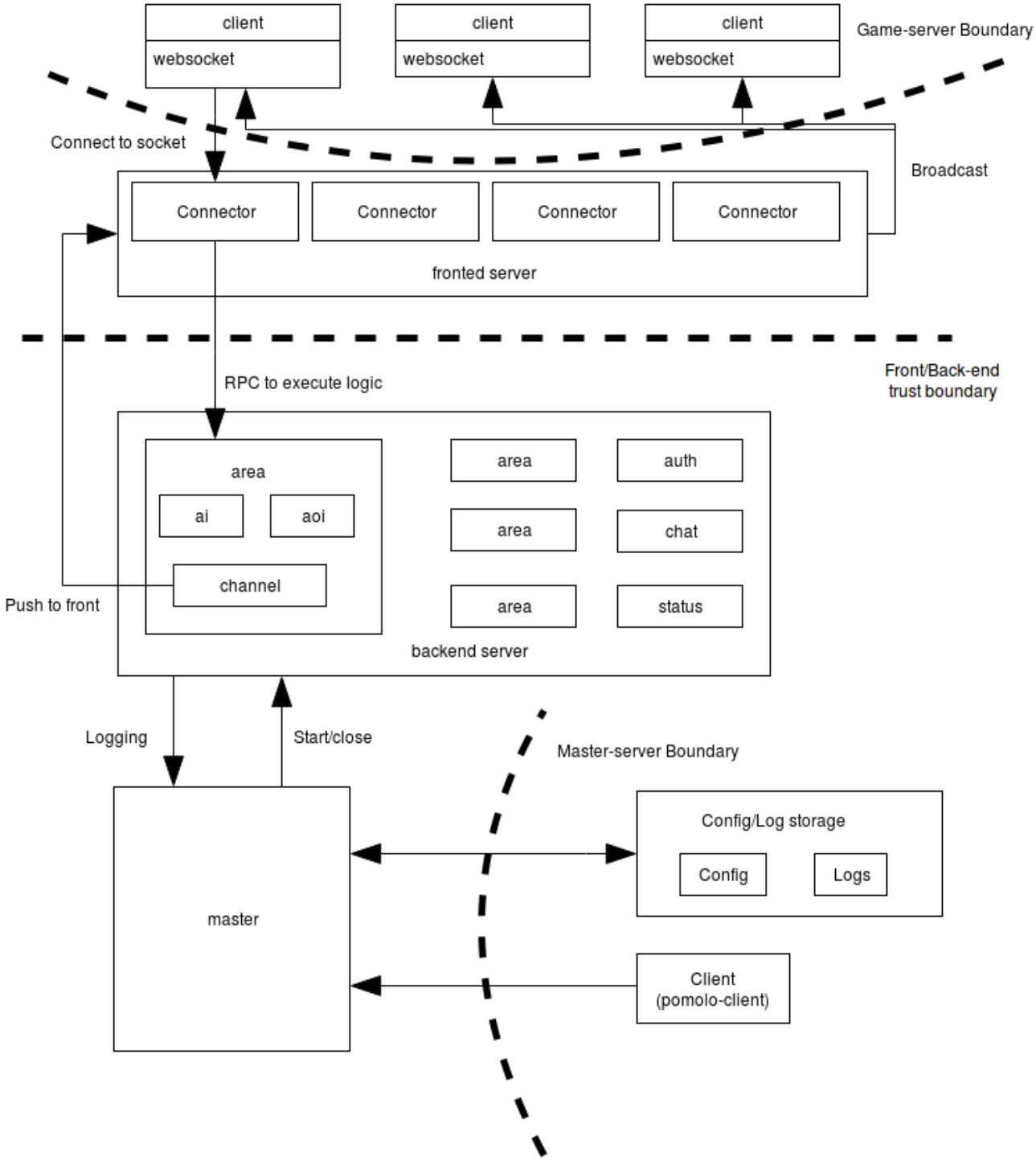
```
PS C:\Users\Bruker\github\NetEase\pomelo> cloc .
     146 text files.
     142 unique files.
Complex regular subexpression recursion limit (32766) exceeded at script/cloc-1.72.pl
line 9262.
      96 files ignored.

github.com/AlDanial/cloc v 1.72  T=2.00 s (63.5 files/s, 8153.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                     104           1998           2346          10786
Markdown                         2            140              0            539
JSON                            15              9              0            340
CSS                              1             11              0             65
HTML                             1              0              0             57
Bourne Shell                     2              0              0              6
YAML                             1              0              0              5
DOS Batch                        1              0              0              4
-------------------------------------------------------------------------------
SUM:                           127           2158           2346          11802
-------------------------------------------------------------------------------
```

```
PS C:\Users\Bruker\github\NetEase\pomelo> cloc lib/
      71 text files.
      71 unique files.
      22 files ignored.

github.com/AlDanial/cloc v 1.72  T=0.50 s (142.0 files/s, 19522.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                      71           1155           1975           6631
-------------------------------------------------------------------------------
SUM:                            71           1155           1975           6631
-------------------------------------------------------------------------------
```

```
PS C:\Users\Bruker\github\NetEase\pomelo> cloc test/
      26 text files.
      26 unique files.
      33 files ignored.

github.com/AlDanial/cloc v 1.72  T=0.50 s (52.0 files/s, 6228.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                      23            457             17           2541
JSON                             3              3              0             96
-------------------------------------------------------------------------------
SUM:                            26            460             17           2637
-------------------------------------------------------------------------------
```

Architecture overview

The goals of Pomolo is to provide a fast, scalable, easy to use and powerfull framework for setting up serverclusters for games or similar applications. The application is used to define servers, behaviour and communication. Servers must be part of frontend, backend or master. The frontend works as a router, for the client trafic, to the backend where the main logic is. Communication between the servers is done via **R**ermote **P**rocessing **C**alls. Pomolo atempts to be as generall as possible, allowing it to be used for as much as possible. Pomolo is intended to be used with additional tools and plugins.



Potential entrypoints:

ID	Name	Description	Trustlevel
1	HTTPS	The main access method for all users. Accessed by for the sake of playing the game, or managing application	(1) Anonymous Web User (2) User with Valid Login Credentials (3) User with Invalid Login Credentials (4) Administrator
1.1	Application	Access point for playing the game	(2) User with Valid Login Credentials (4) Administrator
1.2	tcp/udp Port:3005	Default port for connecting to Pomolo-cli	(1) Anonymous Web User (2) User with Valid Login Credentials (3) User with Invalid Login Credentials (4) Administrator
1.2.1	Pomolo-cli	Administrator commandline interface	(4) Administrator

Assets:

ID	Name	Description	Trustlevel
1	User credentiales	User login details and personal information	(3) User with Invalid Login Credentials (4) Administrator
2	Admin credentiales	Admin login details and personal information	(4) Administrator
3	Accessebility	The uptime of the game servers	(4) Administrator
4	Response time	How quickly the game responds touser input	(4) Administrator
5	Configuration files	Contains environment variables that control data flow	(4) Administrator

Trust boundaries

Client <---> Game-server

The dataflow that crosses this boundary, can be any external entity trying to connect too the application. The incomming data should be checked for **S**poofing, **D**enial of service and **E**levation of privileges. The outgoing traffic should be checked for **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privileges.

Admin-tool (cli/web) <---> Game server

The Admin-tool can be accessed by an external client and should be checked for the same threats as the Client <---> Game-server boundary.

Log/config storage <---> Game server

The information of logs and config files can tell an attacker a lot about the behaviour of the system ontop of what the Pomolo sourcecode already does. Config files should not be changed by the server processes, only read. The Log files should only be changed by the server, not read or sendt by the server processes. The dataflow should be checked for **I**nformation disclosure and **S**poofing when writing logs. When using configuration files, **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privileges.

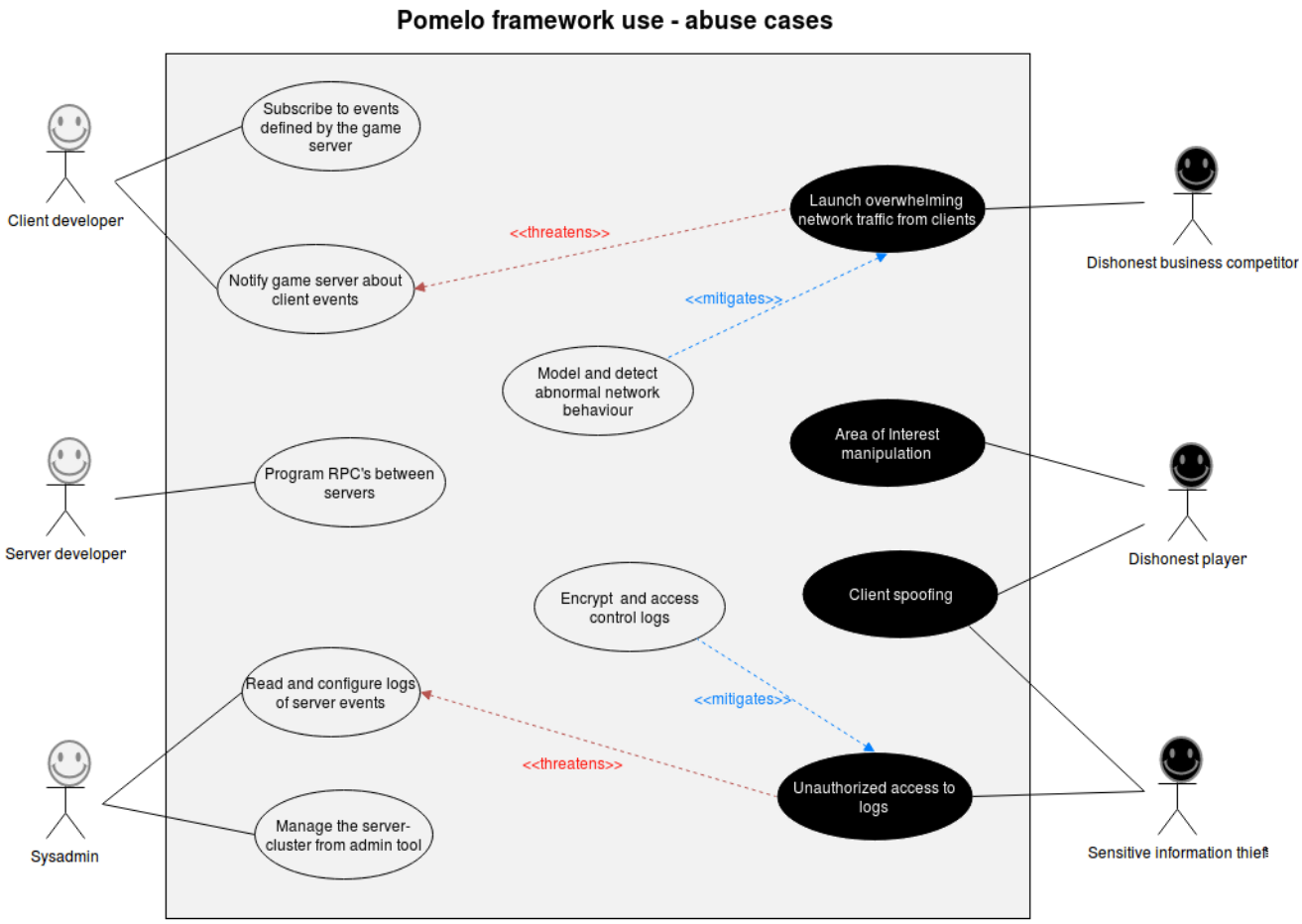
Front-end <---> Back-end

The request that comes in to the connector is validated in-part by a uuid connected to the session, so if a client can give this uuid within the request (among other possible validation data), he's validated. Thus the request goes to it's respective backend-server. It could have been an attacker spoofing a user to say, achieve an advantage if in-game or a possible change in target user's accout. The dataflow from the front end to the backen should be checked for **S**poofing, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privileges. Flow from back end to front end should be checked for **S**poofing, **R**epuditaion, **I**nformation disclosure, **D**enial of service and **E**levation of privileges.

Task b)

Text: Based on the understanding of the system, develop a use case and abuse diagram that has at least two actors and five use cases and four abuse cases.

Use case diagram



Use Actors

Client developer Will develop client side code using the various SDK's provided by the Pomelo framework. Experienced client developer's may also develop their own SDK's.

Server developer Will develop the services on server side, and set up connections between different services.

System administrator Monitors servers. Manage scaling, maintainance of servers.

Use cases

1. Subscribe to events defined by the game server application *Actor:Client developer*

2. Notify server about client events *Actor:Client developer*

3. Manage the server cluster from a master server *Actor: Sysadmin*

- Spawn/shut down servers.
- Monitor resource usage and status.

Reference: wiki/Builtin-Components#master

4. Read/Configure logs of server events *Actor: Sysdmin*

- Log events from servers.
  wiki/Log-Managment

5. Configure Remote Process Calls(RPC's) between servers *Actor: Server developer*

Abuse Actors

Dishonest player

Motivation: Most online games have some element of competition. A considereable amount of players are willing to do "whatever it takes" to gain an competitive advantage over others. *Expected skill level - Low/moderate*

Sensitive information thief

Motivation: Sensitive information has monetary value in the right hands. A thief would be interested to exploit weak systems, to uncover secrets about it's users. For instance passwords, play time, and more. As the service scales, more potential value could be extracted from it's users. *Expected skill level - Moderate/high*

Dishonest business competitior

Motivation: Would benefit from destabilizing your system, hurting the reputation of your service. This would make it easier to convince players to change to other services. *Expected skill level - High*

Disgruntled insider *Motivation:* If an employee feels badly treated, in extreme cases he or she might think it is a good idea to cause the company harm. Expected skill level - Expert

Abuse cases

1. Launch overwhelming traffic from client(s) *Actor: Business competitor*

- Creation of fake game clients to overload the servers. The clients can be created anywhere, on any platform, since the server uses open internet protocols to connect.
- A single client could also cause problems if it were to find vurnerabilities which would enable exessive traffic.

2. Client information spoofing *Actor: Player, Information thief*

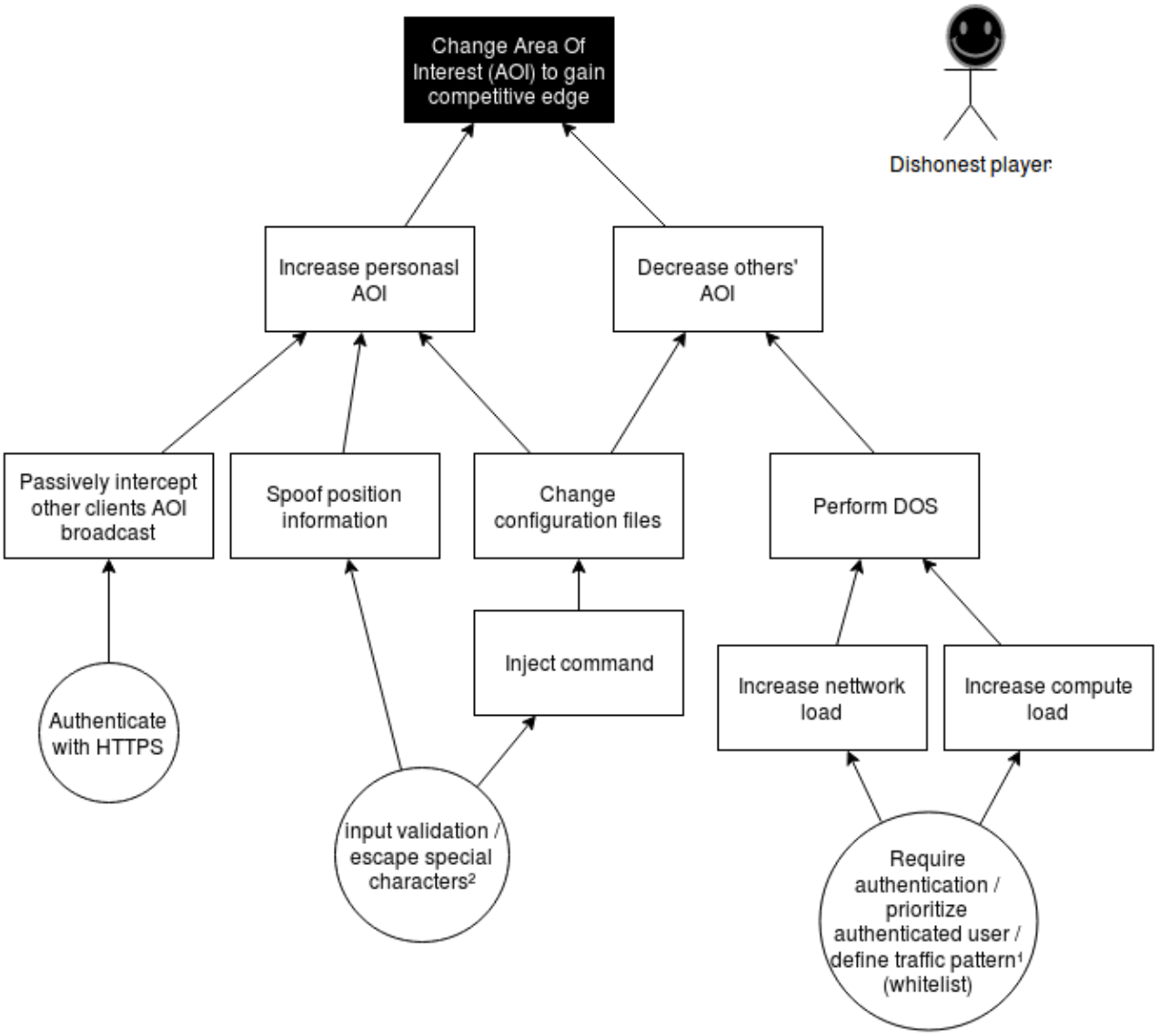
- A player may spoof the identity of other players, to gain access to sensitive information. It could also be used to gain in-game advantages, if the game server is hosting a competitive game.

3. Unauthorized access to logs *Actor: Information thief, insider*

- Access to logs reveals important information about the operation of the system. Logs could also be tampered with, to hide important security events.

4. Server broadcast Area of Interest manipulation *Actor: player*

- In cases where the server will broadcast information to clients using an Area of Interest policy. If this area of interest system can be tampered with, a player may gain a competetive advantege by increasing it's Area of Interest, and decreasing other players Area of Interest.



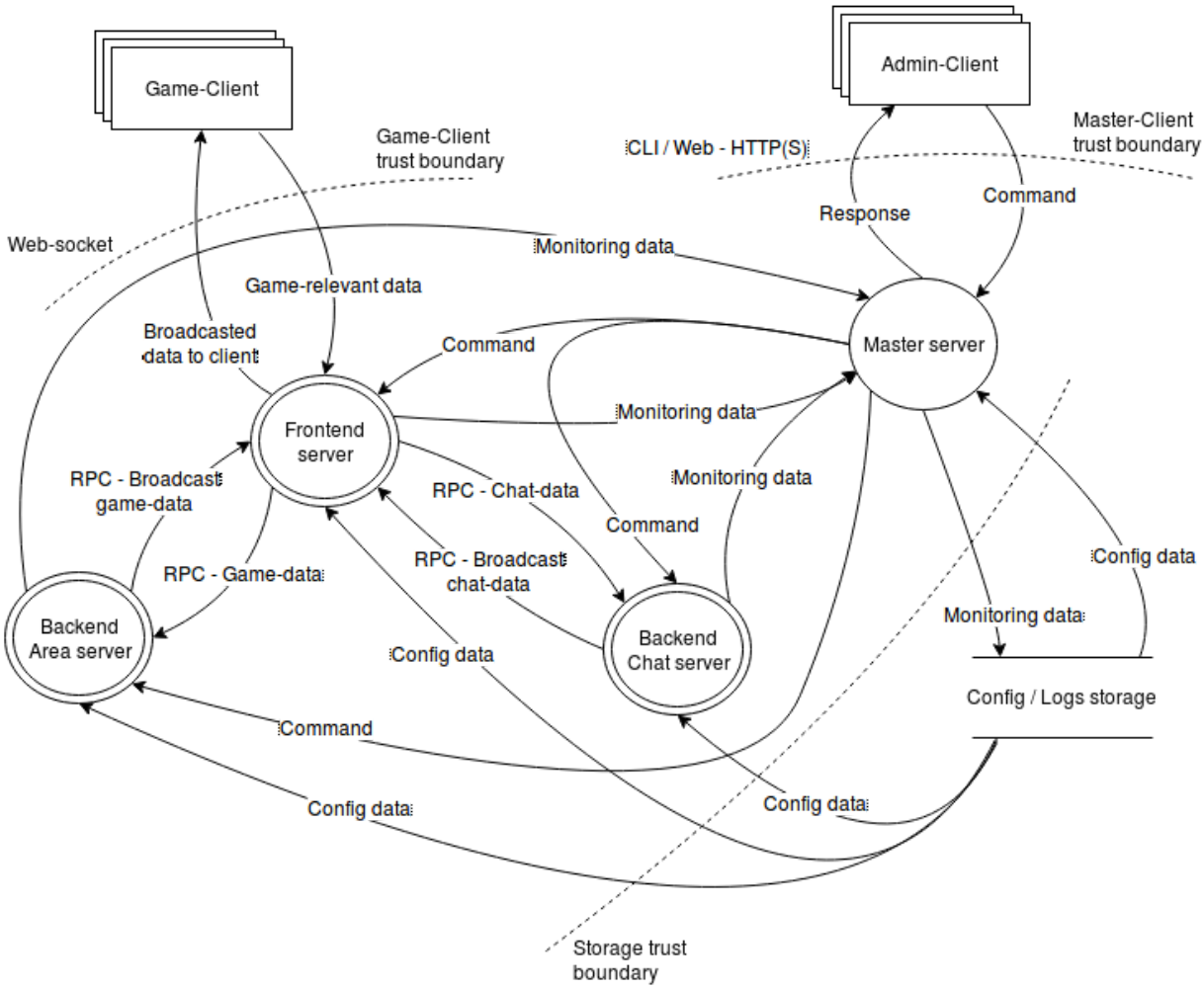
1: DDoS mitigation wikipedia - https://en.wikipedia.org/w/index.php?title=DDoS_mitigation&oldid=796916266 - 04.11.17
2: https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet#Goals_of_Input_Validation

Task c)

Text: Follow the STRIDE methodology we discussed in the lecture to perform threat modeling:

- Identify trust domains, the trust boundaries and the attack surface.
- Create a DFD diagram that represent of the application.

  DFD diagram



References:
- https://github.com/NetEase/pomelo/wiki/Builtin-components#channel
- https://github.com/NetEase/pomelo/wiki/Builtin-components#remote
- https://github.com/NetEase/pomelo/wiki/Builtin-components#server
- https://github.com/NetEase/pomelo/wiki/Builtin-components#session
- https://github.com/NetEase/pomelo/wiki/Builtin-components#connector
- https://github.com/NetEase/pomelo/wiki/Builtin-components#monitor
- https://github.com/NetEase/pomelo/wiki/Builtin-components#master
- https://github.com/NetEase/pomelo/wiki/RPC-Invocation
- https://github.com/NetEase/pomelo/wiki/Pomelo-framework-overview#rpc-invocation-abstraction
- https://github.com/NetEase/pomelo/wiki/Communication-with-Client
- https://github.com/NetEase/pomelo/wiki/Backend-Server

Threat table

- Identify the threat types to each element.
- Identify at least ten possible threats: at least two for a data flow, two for a data store, and four for processes, two for external entities. Discuss each threat and how it can be realized and what it can affect.
- Identify one possible mitigation for each threat. *

No.	Threat Description	Threat type (OWASP - ASF categorization)	Mitigation strategy
1	Session Hijacking (data flow)	User and Session Management	Input validation, escape special characters and key words
2	Monitoring information disclosure (data flow)	Auditing and Logging	No stored sensitive information and encryption
3	Log injection/removal (data store)	Data Protection in Storage and Transit	Input validation, escape special characters and key words
4	Log information disclosure (data store)	Auditing and Logging	Information Disclosure
5	Error message discolsure of server state.(processes)	Error Handling and Exception Management	Fail safely, Check if things go right, otherwise do not allow access to resources.
6	Weak cryptographic encryption used. (e.g md5/sha 1) (processes)	Data Protection in Storage and Transit	Use strong encryption e.g sha256, open source, validatied, verified
7	Unauthorized user changes server configuration (processes)	Configuration Management	Restrict configuration to administrators, log privileged actions
8	Incorrect implemented input validation. (Processes)	Data Validation / Parameter Validation	Range checks, formal definition of whitelisted input, type checks.
9	Client sends large amount of requests (external entity)	Authentication	Identify normal traffic and use it for whitlisting clients. Input validation.
10	Kicked Employee "rm -rf /"- directory (external entity)	Authorization	non-repudiation, backup, principle of least privileged access

OWASP ASF threat types

References to help you:

1. OWASP threat modeling guide
2. OWASP threat modeling cheat sheet
3. Microsoft threat modeling guide
4. How to describe a Misuse/Abuse case
5. The threat modeling book - Adam Shostack. Threat modeling: Designing for security.