An Active Game Bot Detection with Security Bots

Igor Tomičić, Tomislav Peharda
University of Zagreb
Faculty of Organization and Informatics
Artificial Intelligence Laboratory
Pavlinska 2, 42000 Varaždin, Croatia
{igor.tomicic, tpeharda}@foi.unizg.hr

Andrija Bernik
University North
Jurja Križanića 31b, 42000 Varaždin, Croatia
abernik@unin.hr

Abstract. *Cheating in computer games can cause legit players to feel deprived, lose their interest and in time leave the game - which further leads to the decrease of profit in the gaming industry. Avoiding such a scenario is the main focus of research papers dealing with game cheating and cheat detection mechanisms within computer games, where there is a seemingly simple question to answer - is this player a legit human player, or an artificial bot? Most of the research is trying to identify patterns that could give away the presence of the bot, and in the same time, research is being done on building the bots with human-like behaviours which might deceive such mechanisms. In this paper, we present an active bot detection method implemented within a "security bot" - an artificial player which is infiltrated within the game and is able to identify suspicious game players, approach them, and perform a form of the Turing test on them, in order to identify possible cheating bots which have successfully deceived all the existing passive detection methods.*

Keywords. computer games, artificial intelligence, AI, bots, security

1 Introduction

For quite some time, computer games have been part of everyday life for a significant number of people, and according to a news report published by DFC Intelligence, there were more than 3 billion gamers by the 2020 (DFC Intelligence authors, 2021). Nowadays, almost every computer game provide some sort of online component; either as comprehensive as playing with other players completely online, or as simple as providing a scoreboard containing results from other players (Wikipedia contributors, 2021b).

Considering that games require an input - as in, a user action - and process it in order to provide a reaction to it in terms of an output, they are prone to exploits (Cone et al., 2007). Exploiting a computer game is more often referred to as cheating (Wikipedia contributors, 2021a) and indicates an act where a player obtains some form of an unfair advantage against the other legitimate players. In competitive games, where the goal may potentially be to obtain score points, cheating would refer to an illegal action that player performs to get better points. Some games may incorrectly implement functionalities that cunning players leverage in their favour. The described type of oversight falls into a group of bugs and glitches (Velasco and Delgado, 2021). The other type, where a player influences the game-play directly thus obtaining advantage requires application of different techniques that often include advanced computer science skills.

The more complex a computer game is, the higher number of vulnerabilities it might have. In a simple 2D snake game (Wikipedia contributors, 2021d), the most influencing exploit would be related to score points. Contrary, in a 3D multi-player first-person shooter (FPS) game (technopedia contributors, 2021a), a player has much more functionalities to exploit. That could relate to items a player brings, geo-location, behaviour towards other players and so on. Besides for numerous game functionalities that may be exploited, there are also several ways to develop cheats (Wikipedia contributors, 2021a). Some cheats directly affect player capabilities that could, for example, result in enabling a player to see locations of opponent players that otherwise shall not be visible. Different kind of cheats for the multi-player type of games are focused on malforming network requests that are dispatched to the opponent players. That way, a player hypothetically misinforms its opponents about actions it performs. In massively multi-player on-line role-playing game (MMORPG) (technopedia contributors, 2021b) there is increasing use of bots (Thawonmas et al., 2008) that also fall into group of cheating. Bots are primarily used to do simple repeating tasks on behalf of a human player in order to gain some form of an advantage. As (A. R. Kang et al., 2012) argue, "game bots destroy the game balance and consume game contents fast. They cause honest users to feel deprived, lose interest and eventually leave the game." Consequentially, this leads to the decrease of the profit in the gaming industry. Limelight (Limelight networks authors, 2018) argues that 57% of gamers will not continue to make purchases or play games on a website that has previously suffered a security breach.

Clearly, computer game producers do not want their games to be exploited, as that could negatively impact them from different aspects (Y.-C. Chen et al., 2007). Therefore, a lot of effort has been put into securing games as much as possible. In computer games themselves, game producers have been adding advanced mechanisms to detect if player in any way attempts to change the game-play. In order to prevent malmorfed network requests, additional focus has been put towards better communication encryption and advanced identification/authentication techniques (Zhao, 2018). The stated improvements are dealing directly with hardening the game security. However, a more human-like type of cheating includes development of bots which are harder to catch as they are developed to work within the game boundaries, without exploiting them. Therefore, different behaviour analysis mechanisms (Thawonmas et al., 2008) are put in place to differentiate a human player from a bot.

Because the modern research is focused on developing bots that seem to be more human in their behaviours, thus deceiving the passive detection methods as will be argued in the Related Work section, we propose a method of an active bot detection in the form of an automated Turing test, where our security bot approaches a player within the game and tries to communicate with it. The answers of the approached player ("the suspect") would be evaluated, and based on the given answers (or the lack of the same), our security bot would assess if the "suspect" is a human player, or an artificial one.

2 Related Work

As stated previously, cheating in computer games is a large concern for any type of audience (Y.-C. Chen et al., 2007). Different parties attempt to take various strategies in order to prevent cheating. S. Ferretti and M. Roccetti (Ferretti and Roccetti, 2006) focus on cheating detection of malformed actions in peer-to-peer games. Essentially, what happens is that players perform actions that do not complete immediately, but instead take time to resolve. An opponent is informed about the execution of an action including time it took to perform it. The problem here is that any player may malform an action data that is sent to their opponent and fake execution times, therefore, influencing the game-play. What authors propose in order to detect such cheating is to measure the execution time of each available action in the game, and store it on each players' end. That way, when players are engaged in a match, an automated mechanism built in each players' game engine evaluates the opponent action execution times to detect potential cheating.

C. Zhao in (Zhao, 2018) speaks about a similar problem, but on a more general level. In the article, Zhao emphasises on the development of cheats that exploit the game and trojans that can be injected to peers over network. Considering the both listed type of cheating is performed online, the solutions that are proposed for avoiding the associated risks are based on implementing proper encryption systems such as the RSA algorithm and/or enhanced identification/authentication mechanisms so that players could not easily spoof their identity, as well as on higher performance servers and bandwidth, in order to prevent any unplanned server downtimes, that could open up more space for additional harm in sense of leveraging network instability.

In (Thawonmas et al., 2008), authors deal with a type of cheating in MMORPG that introduced bots that play a game on behalf of a player. Essentially, the bots are designed to do simple repeating tasks, such as battling NPCs or trading possessed items to gain further advantage. Authors suggests that by the analysis of bots behaviour, it is possible to detect them. In particular, frequencies of the stated actions should distinguish human players from bots, as bots tend to perform certain actions much faster.

Python-based bots that can play a game instead of the human player, by leveraging an unsecured network protocol, is presented in detail in (Tomičić et al., 2019). These bots use a low-level game interface to connect to the game and control a game character through specifically crafted network packets, and a higher-level interface for basic reasoning.

A group of authors (R. Kang et al., 2013) examine how game logs can potentially be used to track down bots in a MMORPGs game. Their assumption is that when players and bots engage in a party play, each type has different goals. A player engages in a party play to complete quests that are otherwise harder to complete for a single player, while behaviour of bots is geared towards items acquisition. That also implies that party plays last much longer (even indefinitely) when bots are playing, which is one of the main indications of bots involvement. Authors propose that the analysis of party play logs that contain information about game events, especially repetitive player actions, could unveil bots.

Similarly, in (Lee et al., 2016) research is also geared on unveiling bots in MMORPGs based on the log analysis. Concrete technique they vouch for is the analysis of player actions from a game log as a function of the time lag. On a low level, that includes comparison of a human player actions against bot actions to construct a model with capabilities to detect a bot behaviour.

Depending on a game genre, there are some key actions and characteristics that can be looked for to distinguish a bot from a human player. In 2008 and 2009 there was a BotPrize competition (Hingston, 2010b) organised with a goal to develop a human-like bot. Bots were developed for a FPS game called Unreal Tournament 4. Judges, whose role was to give their verdict whether a player in a game was a bot or a human, would analyse behaviour and game-play. Behaviour of bots were not so convincing, therefore, judges were able

to properly classify players. However, based on comments from judges, what uncovered bots were lack of capabilities to plan their actions as well as inconsistencies in their intentions, static movements, and aggressive behaviour. As for lack of planning capabilities, bots would engage into one action and before completing it, move onto another one. Any advanced planning, that would include applying strategy to combat opponents was fully missing. In terms of static movements, bots would show signs of not having smooth movements, but rather stiff. Not only that, at times they would get stuck and were unable to proceed further. Aggressive behaviour is another indicator that makes it easier to detect bots. More specifically, bots would shoot a lot, and be very accurate at it. The speed and accuracy can be remedied trivially however; as (Hingston, 2010a) argue, "bot that shoots too quickly and accurately is easily identified as non-human, but it is simple to slow the bot down and make its shots less accurate".

To identify bots, they might be even tested in specifically crafted scenarios. For example, putting a bot in a location where there are plenty of obstacles would much faster reveal it than putting it in a location with lots of open space. On a similar note, having a bot battle multiple opponents at the same time may potentially result in bot resolving the situation in its favour with greater success comparing to how a human player would perform (Hingston, 2010b). Thus to minimise time required to identify bots, scenarios where bots tend to show weaknesses could be designed with indications on bot reactions to given scenario.

The primarily approach to detect a bot is by behaviour analysis (Thawonmas et al., 2008). Bots can be seen as regular players with a concrete goal set. Often times, they are developed in such a way that they do not hold superior capabilities to regular players. Identifying bots depends on a context. In MMORPGs some bot types are defined to do repeating tasks indefinitely (Thawonmas et al., 2008) thus to identify them, key indicators to look for would be how long is a bot engaged in a game (if information is provided) and what sort of tasks it executes. If a game is taking longer than some chosen time reference point, that could be a relevant indicator. If a bot does a singular action continuously, such as trading items, that also might be a relevant indicator. Game events and player actions are usually stored in game logs, therefore, analysis of game logs can be used to identify them (Lee et al., 2016). One of the possible ways to analyse game state changes is by applying machine learning algorithms, such as Bayesian network approach (Yeung and Lui, 2008).

A similar approach involves analysis of the network traffic (K.-T. Chen et al., 2009). A player pragmatically informs either a server, or its peers about actions it performs. When examining network traffic, the main focus should be on the execution time and the traffic magnitude. What that means is that a bot may potentially inform server or its peers about completion of an action by malmorfing a network request and modification of execution times (Ferretti and Roccetti, 2006) so that the execution time takes shorter time than it originally does.

Besides action execution times and repetitive actions, another aspect that could be analysed is player movement (Mitterhofer et al., 2009). Movement is also communicated over different channels either as a network traffic, in logs or shared exclusively with the server. By extracting waypoints, it is possible to recognise paths that are repetitive. If a bot always takes the same path between concrete outset location and a destination, it may indicate that it has calculated the path as the optimal one.

With all the reviewed research body in the domain of bot detection in mind, an automatic bot detection methods where an artificial system may differentiate between the human and the artificial player still seems to be in its relative infancy. The use of CAPTCHA tests are considered in (Golle and Ducheneaut, 2005), but this brings along the unnatural disruption of the game flow. Others, as mentioned, perform an analysis of behaviour patterns, such as patterns of movement ((Kuan-Ta Chen and Hong, 2007), (Kuan-Ta Chen, Jiang, et al., 2008)), but as previously argued, behaviours can be adapted to slip through such techniques (Hingston, 2010a), (Soni and Hingston, 2008), (Schrum et al., 2011). Moreover, this year for the first time on the BotPrize competition, two bots achieved humanness ratings of over 50%, whereas the human players achieved average humanness ratings of just 40% in the form of a Turing test which is conducted within this competition (BotPrize authors, 2021).

In light of all this bot behavioural advancements towards bot humanness, we propose a method of an active bot detection in the form of an automated Turing test, where our artificial player (security bot) approaches the player and tries to communicate with the player. The answers of the player would be evaluated, and based on the given answers (or the lack of them) our security bot would assess if the "suspect" is a human player, or an artificial one.

2.1 Types of Bots

Capabilities and complexity of bots depend on their purpose which also indicates how challenging is it to develop them. In this context, bots can be classified in groups based on their level of advancement. A bot that performs a single repetitive task, such as, writing a message in the chat can be completely agnostic to its environment, as it requires no input. To develop this type of a bot, there is no reasoning skills to be involved, therefore, development shall be easy (Thawonmas et al., 2008).

Bots that require to process the input in order to properly react to an event require reasoning capabilities

to some degree. If they are developed to perform a simple action, such as battling NPCs or trading items, in order to make an advantage in favour of a player, they need to be able to recognise patterns that lead them to it. Usually this mean comparison conditions that evaluate player state and input attributes (R. Kang et al., 2013).

A higher degree of reasoning capabilities involve machine learning algorithms (Yeung and Lui, 2008). That could either be utilised to develop complete behaviour of a bot so that it may participate in all actions provided in a game or to enhance capabilities of performing a single task. One of the possible approaches to develop behaviour of a bot is by application of Bayesian network. The approach includes programmatic analysis of attributes that describe different states of a player. The outcome of the process is a model that is capable to react to state changes in the real-time. In simpler games, where there is a limited number of actions available, a bot could be specialised to execute only some of the actions rather than all of them. For example, in Pictionary (Wikipedia contributors, 2021c), where the main tasks consist of properly drawing a suggested item and for opponents to guess them, various computer vision tools may be facilitated to enhance bots capabilities (Baroffio et al., 2015).

Previous types of bots are primarily developed to gain advantage on behalf of a player, however, they do not put much significance on how easy or hard is it to detect them (Hingston, 2010b). Human-like bots do not only intend to be precise in their actions, but the goal of this type of bots is also to replicate human behaviour as much as possible, so that opponents and detection mechanisms have hard time detecting them Soni and Hingston, 2008. In an FPS game that was discussed previously, that implies smooth movements, planning capabilities and environment awareness. To develop a bot with such degree of advancement, it is likely that sole application of machine learning algorithms may not be sufficient thus requiring manual adjustments of bot behaviour.

In this paper, we have assumed "the worst case" scenario, where a bot is advanced to the level that it can imitate human behaviour - both in playing the game and in conversational capabilities - based on the currently available research and their limitations.

3 Requirements

The model proposed within this papers relies on a game API through which we could infiltrate our security bot into the game, and on the game chatting platforms, through which our bot can communicate with other players. The game API on which this model partly relies is based on the work presented in (Schatten et al., 2018), which currently supports one MMORPG game (The Mana World). In order to be able to distinguish the legit human player from an artificial one, our security bot needs to be equipped with the ability to perform a form of a Turing test to a suspect player. Because of this, we tried to identify types of questions that a more advanced AI bot playing the game would fail to answer correctly. In most cases, bots are designed to perform specific actions, and rarely are equipped with advanced chat capabilities; such bots are programmed to do only a few certain things, and can easily be detected by starting a simple, general-topic conversations with them. For example, a security bot can ask a simple questions like "what is the colour of the clear sky?", or "give me any number that is not the result of adding two plus two", and even a simple response to the "Hello" can yield results if there is no answer, as the real players are obliged to answer to the security bot. In this work, we anticipate the "worst case scenario", i.e. a bot that can imitate human chat responses.

A group of authors have summarised passive and active bot detection methods, and more significantly for our work, strategies for actively interrogating suspected chatbots (McIntire et al., 2010).

As for the passive detection, authors analyse message size and inter-message delays in order to successfully distinguish humans from bots. According to (Gianvecchio et al., 2008), in contrast to most bots, human inter-message delays (times between sequential message transmissions) “appeared to follow a distinct power law distribution, and human message sizes seemed to follow an exponential distribution (with $\lambda = 0.034$)”. As for the active detection, authors list possible strategies detailed in several other papers. We have initially dismissed a few of them (questions with very informative answers, keyword targeting, evasiveness, using rating games, using ambiguous questions) as such tactics either do not seem feasible for the implementation of our security bot because of the automated nature of the proposed interrogation method, whereas these tactics would require a more complex answer analysis and reasoning, or they might be too intrusive for the game play, breaking the game flow and irritating human players.

Other listed tactics proved to be more useful for our security bot, such as:

- Challenging the syntactic engine, which include questions based on elementary logic (for example: “if New York is north of Atlanta, is Atlanta south of New York?”), common typing shortcuts (for example: “do u like to go 2 dinner b4 going to c a movie?”), using figures of speech or slang, and requesting enumerations to simple questions (for example: “name three things you can do with a ball”).

- Using so called “URL questions” (Understanding, Reasoning, and Learning), where the idea is to probe basic human intelligence. Examples are listed as follows.
  UNDERSTANDING: “What shape is a door?”; “What happens to an ice cube in a hot drink?”

REASONING: “Altogether, how many feet do four cats have?”; “What does the letter M look like upside down?”;
LEARNING: “What comes next after A1, B2, C3, . . . ?”; “PLEASE IMITATE MY TYPING STYLE!!!!”
The bots would answer these questions nonsensically, evade or ignore the questions according to (French, 1988).

- Using sub-cognitive questions, which rely on the “internal, physical, and/or personal-historical experiences of human beings”, which computer lacks. In (Cullen, 2009), authors suggest questions aimed at physical structure and not on the higher-level cognition, but on the low-level sensations and perceptions, for example: “What happens to your clothes if you fall into a pool?”; “If you touch a hot pan, what does it feel like?”
- Using a guessing game based on internal human experiences. For example: “Would you like to play a game? Then try to guess what I‘m thinking of. . . " We than provide a series of hints, such as: "It digests food"; "It sometimes aches"; "Most people cannot pat their head and rub this at the same time” (the answer is stomach). According to (Cullen, 2009), bots should fail a Turing Test based on such questions, because again the lack of common human experiences.
- Using a guessing game on general and common sense knowledge. For example, the "suspect" would have to guess what is being hinted at: "You play this game with a black and white ball, your feet, 2 nets, and 11 players on each team" (the answer is soccer). Or "You use this to talk to people, you hold it in your hand, and you dial numbers on it" (the answer is telephone).
- Using emotional-based questions. For example: “How would you feel if you won the lottery”; “Can you describe how you would feel if you were fired from your job for no obvious reason?”
- Using intentional misspellings. For example: "Doo yuo knowe whut thyme it iz?"; "ha+ is y0re 8irthday?"; "Whhat iss yerr favvorite memmorie?" "Can you raed these wrods taht I‘ve tyepd?"

Following these tactics we can extract a significant set of questions that our security bot can use, and most of them could have a relatively simple answers which could be verified by simple if-then logic and regular expressions with practically no need for more advanced reasoning, making the automated Turing test more feasible.

4 A Conceptual Model

The proposed model relies on the work presented within the (Schatten et al., 2018), implementing so called lower and higher-level game interfaces for playing the game with artificial players.

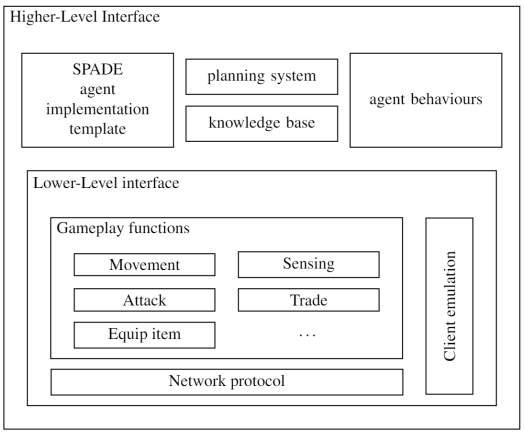

Figure 1: A possible architecture for an agent-based security bot model (Schatten et al., 2018)

The "lower-level interface" presented in the paper enables the emulation of a legitimate game client connecting to the server and playing the game. It does this by re-creating and manipulating network packets being sent between the game server and the client, where the game server could not differentiate between the packets sent by the Python script from those sent by the legitimate game client (Tomičić et al., 2019).

The idea is to imitate the actions of the real human player. By following this idea further, the artificial security bots could be infiltrated within computer games through such interfaces. The reasoning and autonomy of such an security bot could be implemented within the "higher-level interface" (Schatten et al., 2018), which builds upon the lower-level by introducing an agent template implemented in SPADE (Gregori et al., 2006). Authors have implemented a STRIPS-based planning system in Prolog, and an agent knowledge base implemented using the SPADE knowledge-base system for SWI Prolog, which could be refitted for different use-cases - for example, for new types of agents that may require a different AI method, such as machine learning. There is a constant interaction between the two interfaces; low-level interface is providing higher-level agent with actionable behaviours such as navigation, NPC conversation handling, fight handling, party management, etc. The agent autonomy is based on the belief-desire-intention (BDI) model, where the agent initially senses the environment, updates it’s knowledge base, chooses a goal to accomplish, generates a plan for this particular goal and then starts executing it.

Since the focus of our proposed security bot is on other players using potentially illegal activities, the sensing part of the BDI would include observing the actions of other players, and upon detecting the potential illegal action, the bot would set the goal to locate the player in question, approach it, and test it for being

either human player or artificial one in some form of the reverse Turing test. In order for this to work, the security bot would need more detailed insight into the in-game data - other player locations, actions, statistics, and other metadata, depending on the game itself, in order to passively identify suspicious behaviours. This part may be implemented with any of the aforementioned methods for detecting game bots. On the other hand, the security bot may also randomly pick players and "interrogate" them. The approached "suspect" must be able to identify the security bot as such, because every player would be obliged to respond to the security bot, otherwise risk being kicked out of the game.

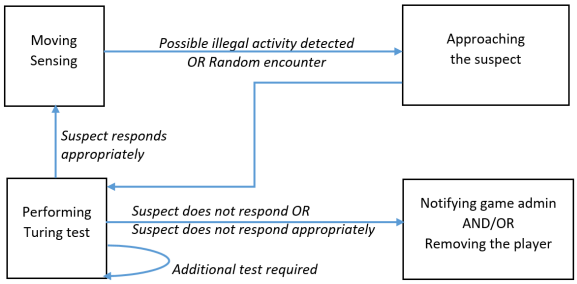

Figure 2: A high level concept of the security bot model

Should the security bot identify the suspect as the game cheating bot, it could report it to the game administrator, or autonomously ban the player from the game. Should the suspect pass the test and prove itself to be human, the security bot departs and goes back to the loop of wandering, sensing and detecting illegal activities, and randomly approaching players.

The game administrators would have a simple interface for upgrading their security bots with new questions and tests in order to have the possibility to be ahead of the game bot designers which could in time download all the challenges and hard code the answers into their bots.

5 Conclusion

The existing body of research is abundant with passive bot detection tactics, analysing game logs, network traffic, bot behaviours and patterns, action execution times, message size and inter-message delays in communication, and other relevant indicators, but rarely the literature answers the question: what if more advanced, human-like cheating bot passes through all these mechanisms and remains within the game undetected? The security bot model proposed within this paper is an active in-game mechanism that would aim to detect and prevent illegal artificial players or scripts from playing the game, should those get by other security mechanisms. It would use a pool of carefully designed questions to conduct a form of an automated Turing test on players with a sole purpose of distinguishing a legit human player from an artificial one. Depending on its designed autonomy, a security bot might report an illegal player, or even permanently remove it from the game. The research so far includes the conceptual model of such a mechanism, with feasible implementation methods through previously developed MMORPG game interfaces which would enable the security bot to login to the game and play it as a regular player, tactics to be included in the development of the question database, and a higher-level interface which would enable the security bot with basic reasoning and planning capabilities. Further research will include the full implementation of such a bot and proof of concept on a single computer game, with the aim of detecting other bots within the game and measuring its efficiency comparing with the existing passive detection methods that were previously implemented.

Acknowledgement

This work has been fully supported by the Croatian Science Foundation under the project number IP-2019-04-5824.

References

Baroffio, G., Galli, L., & Fraternali, P. (2015). Designing bots in games with a purpose. *IEEE Symposium on Computational Intelligence and Games*, *2015*.

BotPrize authors. (2021). Botprize humanness results [[Online; accessed 13-June-2021]].

Chen, K.-T. [K.-T.], Jiang, J.-W., Huang, P., Chu, H.-H., Lei, C.-L., & Chen, W.-C. (2009). Identifying mmorpg bots: A traffic analysis approach. *EURASIP Journal on Advances in Signal Processing volume*.

Chen, K.-T. [Kuan-Ta], & Hong, L.-W. (2007). User identification based on game-play activity patterns. *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, 7–12.

Chen, K.-T. [Kuan-Ta], Jiang, J.-W., Huang, P., Chu, H.-H., Lei, C.-L., & Chen, W.-C. (2008). Identifying mmorpg bots: A traffic analysis approach. *EURASIP Journal on Advances in Signal Processing*, *2009*, 1–22.

Chen, Y.-C., Hwang, J.-J., Song, R., Yee, G., & Korba, L. (2007). Online gaming cheating and security issue. *International Conference on Information Technology: Coding and Computing (ITCC"05) - Volume II*, 518–523.

Cone, B. D., Irvine, C. E., Thompson, M. F., & Nguyen, T. D. (2007). A video game for cyber security training and awareness. *Computers & Security*, 63–72.

Cullen, J. (2009). Imitation versus communication: Testing for human-like intelligence. *Minds and Machines*, *19*(2), 237–254.

DFC Intelligence authors. (2021). Global video game consumer segmentation [[Online; accessed 10-June-2021]].

Ferretti, S., & Roccetti, M. (2006). Game time modelling for cheating detection in p2pmogs: A case study with a fast rate cheat. *The 5th Workshop on Network & System SUpport for Games 2006 - NETGAMES 2006*.

French, R. (1988). Subcognitive probing: Hard questions for the turing test. *Proceedings of the Tenth Annual Cognitive Science Society Conference*, 361–367.

Gianvecchio, S., Xie, M., Wu, Z., & Wang, H. (2008). Measurement and classification of humans and bots in internet chat. *USENIX security symposium*, 155–170.

Golle, P., & Ducheneaut, N. (2005). Preventing bots from playing online games. *Computers in Entertainment (CIE)*, *3*(3), 3–3.

Gregori, M. E., Cámara, J. P., & Bada, G. A. (2006). A jabber-based multi-agent system platform. *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1282–1284.

Hingston, P. (2010a). A new design for a turing test for bots. *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, 345–350.

Hingston, P. (2010b). A new design for a turning test for bots. *IEEE Conference on Computational Inteligence and Games*, 345–350.

Kang, A. R., Kim, H. K., & Woo, J. (2012). Chatting pattern based game bot detection: Do they talk like us? *KSII Transactions on Internet and Information Systems (TIIS)*, *6*(11), 2866–2879.

Kang, R., Woo, J., Park, J., & Kim, H. K. (2013). Online game bot detection based on party-play log analysis. *Computers & Mathematics with Applications*, 1384–1395.

Lee, E., Woo, J., Kim, H., Mohaisen, A., & Kim, H. K. (2016). You are a game bot!: Uncovering game bots in mmorpgs via self-similarity in the wild. *The 5th Workshop on Network & System SUpport for Games 2006 - NETGAMES 2006*.

Limelight networks authors. (2018). The state of online gaming - 2018 [[Online; accessed 14-June-2021]].

McIntire, J. P., McIntire, L. K., & Havig, P. R. (2010). Methods for chatbot detection in distributed text-based communications. *2010 International Symposium on Collaborative Technologies and Systems*, 463–472.

Mitterhofer, S., Platzer, C., Krugel, C., & Kirda, E. (2009). Server-side bot detection in massive multiplayer online games. *IEEE Security and Privacy Magazine*.

Schatten, M., Đurić, B. O., & Tomičić, I. (2018). Towards an application programming interface for automated testing of artificial intelligence agents in massively multi-player on-line role-playing games. *Central European Conference on Information and Intelligent Systems*, 11–15.

Schrum, J., Karpov, I. V., & Miikkulainen, R. (2011). UT_{λ} 2: Human-like behavior via neuroevolution of combat behavior and replay of human traces. *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, 329–336.

Soni, B., & Hingston, P. (2008). Bots trained to play like a human are more fun. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 363–369.

technopedia contributors. (2021a). First person shooter (fps) [[Online; accessed 10-June-2021]].

technopedia contributors. (2021b). Massively multiplayer online role-playing game (mmorpg) [[Online; accessed 11-June-2021]].

Thawonmas, R., Kashifuji, Y., & Chen, K.-T. (2008). "detection of mmorpg bots based on behaviour analysis. *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, 91–94.

Tomičić, I., Grd, P., & Schatten, M. (2019). Reverse engineering of the mmorpg client protocol. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1099–1104.

Velasco, D., & Delgado, D. (2021). Gaming dictionary for newbies [[Online; accessed 10-June-2021]].

Wikipedia contributors. (2021a). Cheating in video games — Wikipedia, the free encyclopedia [[Online; accessed 14-June-2021]].

Wikipedia contributors. (2021b). Online game — Wikipedia, the free encyclopedia [[Online; accessed 14-June-2021]].

Wikipedia contributors. (2021c). Pictionary — Wikipedia, the free encyclopedia [[Online; accessed 14-June-2021]].

Wikipedia contributors. (2021d). Snake (video game genre) — Wikipedia, the free encyclopedia [[Online; accessed 14-June-2021]].

Yeung, S. F., & Lui, J. C. (2008). Dynamic bayesian approach for detecthing cheats in multi-player online games. *Multimedia Systems*, 221–236.

Zhao, C. (2018). Cyber security issues in online games. *AIP Conference Proceedings 1955*.