Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools

Ruan Spijkerman
Academy of Computer Science and Software Engineering,
University of Johannesburg, South Africa

Elizabeth M. Ehlers*
Academy of Computer Science and Software Engineering,
University of Johannesburg, South Africa

ABSTRACT

The use of cheating software in video games to gain an unfair advantage has required the use of anti-cheat software and deterrents such as account bans. Anti-cheat software is, however, always a step behind the opposition and as such new and innovative solutions are required. This paper considers AI driven tools as one such approach and compared decision trees, SVMs and Naïve Bayes classifiers in an attempt to classify cheating and honest player behaviour. The results of the research highlighted the potential for mouse dynamics as a measure of player behaviour, and decision trees as the most accurate detector of honest player behaviour.

CCS CONCEPTS

• **Computing methodologies**; • **Bayesian network models**; • **Support vector machines**; • **Classification and regression trees**; • **Boosting**;

KEYWORDS

Machine Learning, Cheat Detection, E-Sports, Decision Tree, SVM, Naïve Bayes

ACM Reference Format:
Ruan Spijkerman and Elizabeth M. Ehlers. 2020. Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools. In *2020 The 3rd International Conference on Computational Intelligence and Intelligent Systems (CIIS 2020), November 13–15, 2020, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3440840.3440857

1 INTRODUCTION

Cheat codes have long been an intentional part of many single player games with some cheat codes becoming well-known enough to be featured across multiple games [14]. While these built-in cheat codes may give the player an advantage in the game, they do not affect the experience of anyone other than the individual player. The same, however, cannot be said about cheats, exploits, hacks, and scripts used to gain an unfair advantage in competitive multiplayer games [7]. These third party programs or software libraries are used to gain an unfair advantage over other players and in doing so ruins the experience for all other players in the game. The use of cheating software becomes even more problematic when games are played as an e-sport and the integrity of the game as a professional sport must be maintained.

Although the reasons behind a player's decision to use cheating software will not be considered in depth through the research conducted in this paper, it is important to note that it is a problem at all levels of play [13]. The presence of cheaters in a competitive game such as Counter-Strike: Global Offensive (CS:GO), especially at the highest level of competition, affects the game's entire community. If players believe they will be unfairly compromised by playing against a cheater they will look for other games to play or even start cheating themselves, thus creating a feedback loop.

To combat cheaters, video game developers utilise anti-cheat software which attempts to detect the use of cheats before honest players are affected by it [12]. Due to the limitations of traditional anti-cheat methods this research proposes an anti-cheat system that uses a set of Artificial Intelligence (AI) tools that can be used to detect previously known and unknown cheats alike. The remainder of this paper includes a literature study and review of systems similar to the proposed solution in order to better understand the problem as well as current solutions. Thereafter the proposed system's model will be discussed to highlight certain features which may enable better performance than traditional methods. The results obtained after implementing the proposed system will identify major advantages and disadvantages of the system compared to traditional methods. Lastly the limitations of the research will be discussed and potential avenues for future research will be identified.

2 LITERATURE REVIEW

2.1 Cheating software

In general cheating software can interact with a game by either running as a standalone program alongside the game, or by injecting the game's DLL files in order to directly manipulate memory values [4]. Counter-Strike is a first-person shooter, meaning the player controls movement using their keyboard, their vision as well as aim using their mouse, and receive visual feedback from their monitor. These points of interaction with the game are also what most cheating software interacts with in order to give the player an advantage over opponents [3]. Some of the most common cheats used by players are:

- Aimbots, which enhance or completely controls a user's aim.
- Wallhacks (or Extra Sensory Perception), which provide the user with visual information they would not usually have access to. This includes enemy players' positions, health, and equipment.

*Corresponding Author: emehlers@uj.ac.za

CIIS 2020, November 13–15, 2020, Tokyo, Japan

ACM ISBN 978-1-4503-8808-5/20/11…$15.00
https://doi.org/10.1145/3440840.3440857

- Movement scripts, that allow a user to abuse game mechanics that increase player movement.

2.2 Mouse dynamics

Research regarding mouse dynamics as a method for user authentication has proven that user behaviour while using an input device such as a mouse or trackpad can successfully identify individual users [15]. Mouse dynamics consist of a variety of mouse inputs such as clicking, double clicking, and simply moving the mouse. The act of clicking and dragging the mouse, however, has been shown to be the most effective at uniquely identifying a user due to its combination of simpler mouse dynamics [1].

The study by Shujie Hu et al. [8] showed that mouse dynamic data is often influenced by external factors such as individual user hardware and software configurations, and mouse the measurement of actions that occurred in physical space through virtual environments.

2.3 Keystroke dynamics

A study on the use of keystroke dynamics as a method for detecting irregular entries of user passwords identified a number of key considerations for use as a biometric authentication method [9]. Firstly, it is worth noting that the accuracy increased when subjects were entering input that was typed before. This was due to the increased consistency which made trends easier to detect and predict.

Secondly, the study compared various classification algorithms in terms of accuracy and precision. This comparison found Multiclass Support Vector Machines to be clearly superior at classifying users based on keystroke dynamics.

2.4 Statistical data analysis

A study done in 2006 found a scalable cheat detection solution which works on the server side in a multiplayer game [18]. The solution aimed to classify users as either cheating or not, and to also label the type of aiming cheat being used. Player behaviour such as hits and kills per games were found to be similar for cheating players regardless of the type of cheat being used. Taking into consideration the increased likelihood of a player hitting a shot on an enemy already aimed at (through the use of a first order Markov Model) a Bayesian Network was used to categorise players. Most importantly, the method of classification produced no false positives and remained accurate even when cheating software tried to imitate honest behaviour.

3 SIMILAR SYSTEMS

Most competitive online multiplayer games make use of third party anti-cheat solutions for cheat detection and prevention. These traditional systems have a limited level of success since they work in a similar manner as anti-virus software. They are installed and run on a player's machine and the signature of a running game is scanned to detect changes to DLL files or the effects of a known cheat. Other running processes may be scanned compared to known cheating software as well [2][5][16]. Because of this only known cheats can be detected, however, they can be classified as such with a great level of certainty. Traditional anti-cheat software can also be installed on the server side in a multiplayer setting [6]. These solutions evaluate data sent from the client to ensure that player behaviour is within acceptable limits.

Valve, the developers of CS:GO, have implemented a peer review system in addition to their traditional anti-cheat [17]. The "Overwatch" system allows players of the game to review the recorded gameplay from a suspected cheater's point of view before judging the suspect's innocence. Reviewers also build a reputation score as they review more cases in which their judgement was found to be correct which could then increase their weighting when judging later cases.

In the last year Valve have revealed an AI driven solution to cheat detection [10]. Using the results from convicted Overwatch cases, they developed a deep learning solution to detect behaviour indicative of cheating software being used. The features being classified include change in a player's crosshair placement before and after a shot, the weapon they were using, as well as the result of the shot.

4 MODEL

Based on Section 2 of this paper some of the areas in which cheating can be detected are mouse dynamics, keystroke dynamics, and statistical data analysis. This research will compare the accuracy of classifiers with multiple types of user behaviour as can be seen below in figure 1

4.1 Data capture

Data is captured through the use of keyloggers and the in-game console. The captured mouse and keystroke data will be stored in text files and captured in one minute intervals. Captured events will not span over multiple text files, meaning an interval may be extended should an event still be ongoing.

4.1.1 Mouse Behaviour. Based on research by Shen et al. [15] a keylogger will be used to capture all mouse movements and click events. For each mouse action the action type (movement or click event), x-position of the curser, y-position of the cursor, and the time the action took place (measured in milliseconds) is captured and stored in a log file.

4.1.2 Keystroke Dynamics. Similar to the method for Mouse Behavioural data, keypress events will be captured using a keylogger. The key that was pressed and the time it was pressed will be written to a log file with the same being done for key release events. In order to highlight abnormal behaviour, a player's game config file will also be read in and their in-game key bindings extracted. Keypress events that are not of bound keys will then be marked as such.

4.1.3 Console Logs. The game makes use of an in-game console that allows users to change settings, load configuration files, and view game events. At the end of each the console shows the number of hits a player dealt to each enemy player and the damage done through those hits. It also shows the hits taken from enemy players and the damage taken through that. The output of this in-game console will thus be written to a text file for further processing. Log files will be captured at the end of every match, which consist of at most 30 rounds.

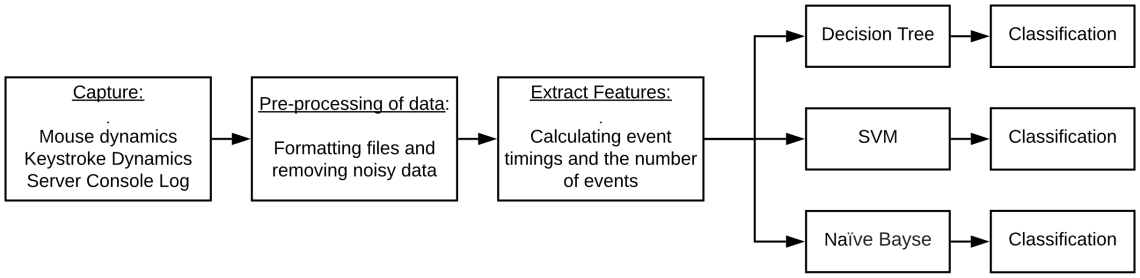

Figure 1: An overview of the process implemented for this study. The Naïve Bayes classifier will, however, only be applicable to the console log features as they are independent events.

Table 1: Requirements for features to be classified as low, medium, or high

Feature	Low	Medium	High
Hits Given	Less than 3	Less than 10	Greater than or equal to 10
Hits taken	Less than 2	Less than 5	Greater than or equal to 5
Damage Dealt	Less than 100	Less than 600	Greater than or equal to 600
Damage Taken	Less than 30	Less than 80	Greater than or equal to 80

4.2 Feature extraction

For the Support Vector Machines and a Naïve Bayes classifier, the extracted features will show statistically independent events. The average input values will be calculated using the data captured as described in Section 4.1 and used for further feature extraction. The extracted features can then be split into test and training sets. Each log file will thus be summarised and labelled as either cheating or honest input. By calculating the average input the effects of a single outlier will be lessened and with it the risk of false classification as cheating.

4.2.1 Mouse Behaviour. Since clicking and dragging is the most information rich mouse dynamic [15], and the most common in CS:GO, the implementation will identify click and drag events in the log files. The average time, average distance, general direction, and average translation of the event will be extracted and stored in a file. The direction of the movement will be calculated using vectors as an angle on a Cartesian plane. Time and distance measures will be used to calculate movement speed as the number of pixels per millisecond. As these features are related, they will not be used together in the SVM implementations.

4.2.2 Keystroke Dynamics. Using the log files, the total time keys were held down for, the average time of a keypress, the average time between keypresses, the total number of keys that were pressed, the number of unique keypresses, the number of unbound keypresses, and the number of unique unbound keypresses can be calculated. The total time of keypress events that are not used for in-game movement will also be calculated.

4.2.3 Console Logs. For each round of the game the following will be calculated using the log files: Total hits given, total damage dealt, total targets, total hits taken, total damage taken, and total attackers. For the features to be used within a Naïve Bayes Classifier they will be further grouped into categories. Please see table 1 below for these categories.

The values in table 1, above, are based on the number of hits required to kill a player using the most common weapons in the game. The game's user interface indicates that a player has low health points remaining once under 20%, while most of the in-game rifles to slightly less than 30 health points' worth of damage. The damage dealt will not necessarily be higher if a player has more hits given. One reason is due to cheating software allowing a player to consistently hit headshots. Hitting a player in the head deals considerably more damage than multiple bodyshots, but the head is a smaller and more difficult target to hit.

4.3 Decision tree

The mouse dynamics are read into a matrix with each row representing the features from a single log file and the columns representing the individual features. This implementation follows the ID3 algorithm as designed by J. R. Quinlan [11] in which entropy is calculated as follows:

$$E(s) = \sum_{i=1}^{c} -p_i . \log_2 (p_i) \tag{1}$$

In the above equation pi is the probability of a record having either a cheating or an honest label. For a field to be chosen to split the data on, however, the entropy has to be calculated using the frequency table of two fields, namely one feature and its label. In this case entropy is calculated as follows:

$$E(T, X) = \sum_{c \in X} P(c) E(c) \tag{2}$$

Probability is calculated as the total number of times a feature as a specific value divided by the total number of records, regardless of the features cheating and honest split. Lastly Information Gain of a

feature is calculated as the entropy of the matrix's labels subtracted with the entropy of that feature with the labels as in Equation 2. Since the features extracted for mouse dynamics are floating point values within a wide range entropy-based binning is used to categorise the values.
Trees are pruned after testing to avoid overfitting. Pruning is done based on a maximum depth with the depth chosen based on when accuracy drops.

4.4 Support vector machines

For this research only linear Support Vector Machines will be considered. A variety of approaches using the same general algorithm will, however, be compared to verify the results.

4.4.1 Feature Pairs. This approach attempts to classify behaviour by looking at feature pairs. A separate SVM model will be trained using each possible combination of the extracted features with the exception of statistically dependent features. This approach will be used as a basis for the following method which will make use of the individual SVMs to make a single decision.

4.4.2 Ensemble Learning. This approach attempts to increase the accuracy of the individual SVMs trained in the first method. All Feature Pair SVMs that achieved a satisfactory accuracy are tested against new data and a prediction is made from a combination of their results.

The overall prediction is calculated in two ways, with the first allowing each SVM's prediction to carry the same weight. The second approach will make use of weighted totals. Each SVMs prediction will be multiplied with the square of its accuracy. For both approaches the best odd number of SVMs are selected.

4.4.3 Multi-Feature. The third approach will attempt to find a linear separation using all features. The best classifier will be selected based on high accuracy and low complexity with complexity being calculated as the sum of the absolute value of each weight. This will verify which features are not linearly separable as their weight should be close to 0.

4.5 Naïve Bayes

Since the only feature set that can be grouped categorically comes from the console logs, this approached will only be used to classify cheating behaviour using this data. The following algorithm will be used to calculate the probability of a player's behaviour being indicative of cheating or honest play.

$$posterior = \frac{prior * likelihood}{evidence} \tag{3}$$

The likelihood of a data sample being both cheating and honest is calculated, with the greater of the two results classifying player behaviour.
The implemented model ensures that a variety of both classifiers and player behaviour trackers get compared. This allows for a wider range of results and a clear comparison between successful and unsuccessful methods can be drawn.

5 RESULTS

Results can be grouped by classifier type with subsections for the considered player behaviour. This is done to present a better direct comparison between the captured player data as that is what impacted the algorithms' performance. The following data compares decision trees based on their input, as well as their performance before and after pruning.

The next table shows the results of feature pair support vector machines. As the majority of feature pairs were not linearly separable, and thus did not perform well, only a summary of the results will be discussed.

The Naïve Bayes classifier was only used to detect cheating behaviour based on console log statistics. It achieved an 87.5% accuracy rate, which is on par with the other classifiers' performance when using console log data.

6 ANALYSIS OF RESULTS

The following section will consider the potential causes of the results shown in the previous section. Through this inspection potential failures in the research will be identified and solutions offered for future research. The implemented model also produced results that may be worth investigating further in another study.

6.1 Decision tree

The effects of pruning the initial decision trees were drastic and a clear indication of initial overfitting as can be seen in table 2 above. This was, however, not the case with a decision tree trained on keystroke dynamics. This was due to the initial tree's shallow depth as a single attribute (the number of unbound keys that were pressed) caused an almost complete split.

It is worth noting that this feature is highly dependent on individual player configurations that may make use of different keybindings. Players accidentally pressing unbound keys could also lead to numerous false positives. Based on these facts, decision trees can only be recommended as a classifier of honest behaviour.

6.2 Support vector machines

Although the Feature Pair approach is statistically valid due to the independent nature of the features, the results shown in table 3 above indicated that not all pairs were linearly separable. Especially the keystroke dynamics based SVMs had a wide range of accuracy ratings. This could be attributed to the larger number of features being considered, however, the console log features provided the most consistent results. All but one feature pair managed to score above 60%.

The ensemble learning approach yielded unexpected results with regards to both the difference between equal and weighted predictions and the most improved input type. Please see these results in table 4 above. Firstly, it should be noted that in all but one of the cases there was no difference apparent based on how predictions of individual SVMs were counted. This is likely due to the majority of Feature Pairs for a given behaviour measure achieving roughly the same result. As the weighting is based on accuracy SVMs with similar accuracies carry roughly the same weight. Secondly, the only behaviour measure that showed any sign of increased performance was that of mouse dynamics. This is potentially due to the

Table 2: Decision Tree accuracy per player behaviour type

	Keystroke Dynamics	Mouse Dynamics	Console Logs
Not Pruned	95.45%	58.33%	66%
Pruned	No Pruning	83.33%	100%

Table 3: Feature pair SVM accuracies

	Keystroke Dynamics	Mouse Dynamics	Console Logs
Highest Accuracy	100%	75%	83.33%
Lowest Accuracy	54.54%	50%	50%

Table 4: Ensemble Learning using Feature Pair SVMs

	Keystroke Dynamics	Mouse Dynamics	Console Logs
Highest Accuracy	90%	77.77%	83.33%
Lowest Accuracy	100%	77.77%	83.33%

Table 5: Multidimensional SVMs

	Keystroke Dynamics	Mouse Dynamics	Console Logs
Accuracy	73%	77%	83%

relatively poor performance of the individual Feature Pair SVMs which allowed for more improvement

The multidimensional SVMs highlighted the poor linear separability of most feature pairs. This can be seen by the comparably poor results shown in table 5 above. Inspection of the weight vector showed that the majority of features were not impacting the final classification in any major way. A major problem during training was the algorithm implementation. A lack of gradient descent or similar algorithm meant that computational resources were wasted searching for a more optimal weights vector where none existed.

6.3 Naïve Bayes

The naïve Bayes classifier was only applicable to a single behaviour measure due to its dependency on categorical data. No clear categories existed within the other feature spaces and as such any comparison would have little value to future research. The classifier did, however, still perform well on the given data. The positive performance is an indication that future research could attempt to find classes within the other behaviour spaces and compare their performance using this classification method.

7 FUTURE CONSIDERATIONS

The research conducted in this paper relied on a single player for data capture. Initial testing included data from multiple players which was soon found to drastically skew results. As such, a complete data set would be required from multiple players before a comparison between players can be made. A large enough set may also be able to overcome the impact of a few players when creating an average with which to compare the input of an unknown player.

Another limitation was the ethical and technical implications of testing the use of cheat software against real players. Data could not be gathered from online lobbies as participants would not be willing participants. For this reason, all data was gathered from observing a human player playing against the highest difficulty in-game bots with and without the use of cheating software.

One major advantage of the implemented solution is the ability to capture a user's input independent of the game being played. CS:GO was selected as there is an abundance of free cheating software available, but cheating is a concern in many online multiplayer games. It would be worth investigating the ability to detect the use of cheating software in not only other First Person Shooter games but also in genres such as Real-Time Strategy or Multiplayer Online Battle Arenas. Different genres would likely require novel features to be extracted, but the machine learning algorithms would not require any adaptations.

8 CONCLUSION

The research conducted through this paper aimed to determine not only the potential of AI based cheat detection methods, but also the usefulness of various player behaviour measures. The successful implementation of three distinct classifiers proves the exciting possibility of accurate, AI-driven cheat detection. Future research should focus on expanding the most successful features and classifiers, and in doing so help build on the academic research into cheat detection.

REFERENCES

[1] Antal, M. and Egyed-Zsigmond, E., 2018. *Intrusion Detection Using Mouse Dynamics*, Stevenage: Institution for Engineering and Technology

[2] BattlEye, 2019. About: How we came here. [Online] BattlEye About. Available at: <https://www.battleye.com/about/>[Accessed 28 March 2019].

[3] Counter Strike Global Offensive Cheats, Hacks & Aimbot. [Online] IWantCheats.net. Available at: <https://www.iwantcheats.net/counter-strike-global-offensive-hack/>[Accessed 28 March 2019].

[4] Double, V., 2016. Unknown Cheats: CSGO Cheat Making 101. [Online] Unknown Cheats. Available at: <https://www.unknowncheats.me/forum/counterstrike-global-offensive/183526-cscheat-101-a.html>[Accessed 26 March 2019]

[5] Easy Anti-Cheat, 2019. Easy Esports. [Online] Available at: <https://www.easy.ac/en-us/esports/>[Accessed 28 March 2019]

[6] Evenbalance, 2019. PunkBuster FAQ. [Online] Available at: <https://www.evenbalance.com/faq.php#>[Accessed 28 March 2019]

[7] Gaspareto, O. B., Barone, D. A. C., Schneider, A. M.*: Neural Networks Applied to Speed Cheating Detection in Online Computer Games*. Fourth International Conference on Natural Computation 4, 526-529 (2008)

[8] Hu, S. *et al.*, 2017. Deceive Mouse-Dynamics-Based Authentication Model via Movement Simulation. Hangzhou, IEEE.

[9] Jawed, H., Ziad, Z., Khan, M. M. & Asrar, M., 2018. *Anomaly detection through keystroke and tap dynamics implemented via machine learning algorithms*. Turkish Journal of Electrical Engineering & Computer Science, 26(1), pp. 1698-1709.

[10] Lahti, E., 2018. Valve has 1,700 CPUs working non-stop to bust CS:GO cheaters. [Online] Available at: <https://www.pcgamer.com/vacnet-csgo/>[Accessed 28 March 2019]

[11] Li, N., Wang, X., Wang, L.: *An Application of Decision Tree Based on ID3*. Physics Procedia 25, 1017-1021 (2012)

[12] Liu, H. L., Tang, B.: *DACA: Dynamic Anti-Cheating Archintecture for MMOGs*, International Conference on Advanced Information Networking and Applications. 2009

[13] Mira, L., 2020. KQLY: "Ban Was Justified". [online] HLTV.org. Available at: <https://www.hltv.org/news/13651/kqly-ban-was-justified>[Accessed 14 July 2020].

[14] Most Popular Cheat Code, 2010. [Online] Guinness World Records. Available at: https://www.guinnessworldrecords.com/world-records/most-popular-cheat-code/ [Accessed 9 July 2019]

[15] Shen, C. *et al.*, 2013. *User Authentication Through Mouse Dynamics*. IEEE Transactions on Information Forensics and Security, 8(1), pp. 16-30.

[16] Valve, 2017. Valve Anti-Cheat System (VAC). [Online] Steam Support. Available at: <https://support.steampowered.com/kb_article.php?p_faqid=370>[Accessed 28 March 2019]

[17] Valve, 2019. Overwarch FAQ. [Online] Available at: <https://blog.counterstrike.net/index.php/overwatch/>[Accessed 28 March 2019]

[18] Yeung, S. F., Lui, J. C. S., Liu, J. & Yan, J., 2006. Detecting Cheaters for Multiplayer Games: Theory, Design and Implementation. Las Vegas, IEEE