

Security issues in massive online games

Jiankun Hu*[†] and Fabio Zambetta

School of Computer Science and IT, RMIT University, GPO Box 2476V, Melbourne VIC 3001, Australia

Summary

In this paper, an investigation is conducted on the security issues in massive multiplayer games. A taxonomy framework for online cheating is provided. Under this proposed framework, online cheating is classified and state of the art counter-cheating techniques are analysed with emphasis on attacks that pose a considerable challenge to the security of massive multiplayer online games. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: online games; security; cheating

1. Introduction

Massive multiplayer online games (MMOGs) have been growing and have been consistently influencing our life from a social, economical and cultural point of view. The videogames industry currently generates staggering revenues of around US\$30 billion a year worldwide: figures from the Entertainment Software Association (ESA) indicate that the current sales of games related items in the US alone is about US\$230 million [1]. Also, a report by Strategy Analytics goes further with its predictions: they forecast revenues from online games will reach \$11.5 billion by 2011, a 25.2 per cent compound annual growth rate [2], meaning one-third of games software revenues will be generated in the online space.

MMOGs have following characteristics: (i) in the games, human players play against each other via game client software distributed among individual players connected via different networks, (ii) it is extremely difficult to know whether an action response received at the game server is generated within

the game's rules or not. These characteristics have rendered MMOGs vulnerable to attacks such as collusion attacks and game procedure abuse attacks etc.

Moreover, most MMOGs create virtual currencies whose role varies from game to game, which has often given rise to interesting interactions with real currencies. For instance, virtual currency can buy new items for the players to enhance their game characters and increase their chances of success in the game. However, third-party companies (i.e. companies other than the game developer and publisher) have engaged in the practise of acquiring large volumes of virtual currency for the purpose of selling to other individuals for real currency. This phenomenon goes under the name of 'gold farming', and has caused outrage in player's communities. An exemplar case is represented by Chinese gold-farming companies in the extremely popular Blizzard Entertainment's World of Warcraft [3]. Gold farmers 'play' computer games by killing monsters and harvesting artificial gold coins and other virtual goods as rewards that can be transformed into real cash. Online gamers who lack the

*Correspondence to: Jiankun Hu, School of Computer Science and IT, RMIT University, 376-392 Swanston Street, Melbourne, Victoria 3001, Australia.

[†]E-mail: jiankun.hu@rmit.edu.au

time and patience to work their way up to the higher levels of the renowned MMORPG are willing to pay the young Chinese to play the early rounds on their behalf [4]. In such an environment, security breaches have the potential to interfere with real economies as much as they do with virtual ones.

A security breach in playing MMOGs refers to breaking game rules. Cheating in MMOGs is defined as [5]:

‘Any behaviour that a player uses to gain an advantage over his peer players or achieve a target in an online game is cheating if, according to the game rules or at the discretion of the game operator (i.e. the game service provider, who is not necessarily the developer of the game), the advantage or the target is one that he is not supposed to have achieved’.

In general, MMOGs cheating is caused by conventional security breaches where confidentiality, integrity and availability are the only attributes concerned. However, MMOGs cheatings have their own features, i.e. social behaviours that are not commonly observed in the conventional security scenarios. Therefore it is important to understand these special security issues and state of the art anti-cheating measures both from a theoretical and a practical perspective. A unified framework covering the MMOGs could be of great help. However, most existing literature focus on individual cheats and associate counter-cheating measures [3,5–10]. Several attempts have been made to construct a framework for classifying and understanding online game cheating. Davis [6] categorised traditional forms of casino cheating. The drawback of this framework is that a casino scenario is not good enough to represent all forms of online game settings in which different cheats may be used. Pritchard [11] discussed many real cases of online cheating that have occurred in various games, and classified them into a framework of six categories. The disadvantage of this framework is that many online game cheats do not readily fit into any of these six categories [5]. Yan and Choi [7] have classified cheats into 11 common cheating forms in online games.

Work by Yan and Randell [5] presents an improved classification by means of a taxonomy of online game cheating defined with respect to the underlying vulnerability (what is exploited?), consequence (what type of failure can be achieved?) and the cheating principle (who is cheating?). This classification is reminiscent of the dependability taxonomy provided

in Avizienis *et al.* [12]. However, the work of Yan and Randell [5] has not covered the counter-cheating measures which are also very important if not more important than the discussion of various cheats. This is because the purpose of understanding cheats is to help design suitable counter-cheating measures. Also generic and seamless integration of individual proposed framework components is lacking. Our recent work extends the taxonomy framework of Avizienis (2004) by incorporating security and dependability in a more generic way [13]. In this paper, we apply this framework to the MMOGs for the cheat classification and also provide in-depth discussion of the latest development on the counter-cheating measures with emphasis on timing cheatings. The paper is organised as follows. Section 2 provides a framework for the classification of MMOGs cheats. Section 3 reviews major counter-cheating measures and finally Section 4 is devoted to final discussion and conclusions.

2. A Framework for the Classification of Cheating in MMOGs

Yan and Randell [5] have classified online cheating forms into two classes as shown in Table I. The ‘generic’ class comprises eight types of cheating in online games, which are shared by all network applications [5]. The ‘of special relevance’ class includes cheating that is either occurring in online games or is exhibiting some interesting features in the context of online games. Based on this table, Yan and Randell [5] has proposed a three-dimensional taxonomy for the classification of online cheats. Online cheating is classified by *vulnerability*, the *cheating consequence* and the *cheating principle* which is reminiscent of the dependability taxonomy framework provide by Avizienis [12].

This framework is very helpful in understanding several types of cheats in online games. However, it does not address the issue of counter-cheating means. From a practical application point of view, this is only a very first step. Counter-cheating measures are also important. Hu *et al.* [13] provided a taxonomy framework for integrating dependability and security. This framework includes means for fault prevention which is useful in establishing a similar framework in the context of online cheating. As our main concern in this paper is about cheating, the remaining discussion will be confined to the malicious ‘faults’ with respect to the framework in [13]. Therefore MMOGs security

Table I. Common cheating forms in online games [5].

Type	Label	Cheating form
Of special relevance to online games	A	Cheating by exploring misplaced trust
	B	Cheating by collusion
	C	Cheating by abusing the game procedure
	D	Cheating related to virtual assets
	E	Cheating by exploiting machines intelligence
	F	Cheating by modifying client infrastructure
	H	Timing cheating
	Generic	G
I		Cheating by compromising passwords
J		Cheating by exploiting lack of secrecy
K		Cheating by exploiting lack of authentication
L		Cheating by exploiting a bug or design loophole
M		Cheating by compromising game servers
N		Cheating related to internal misuse
O		Cheating by social engineering

will be characterised by attributes, threats and means as shown in Figure 1.

The notations below are introduced for the descriptions of such relationships.

2.1. Key Notations

Notations are adopted from our prior work Hu *et al.* [13] and they are tailored for the description of the MMOGs framework proposed in this paper. Detailed explanations for many of these notations are not provided in this paper. Interested readers are referred to Reference [13] for more details.

System: a composite constructed from functional components. The interaction of these components may exhibit new features/functions that none of the composite components possess individually. In the conventional framework of dependability and security, the system normally refers to the network infra-

structure or platform. In this paper, we need to include the game clients as a part of the system as the game interacts directly with the client.

Control system: a system which is under control, normally under regulators' control, to achieve the desired objectives.

Feedback: use of the information observed from system's behaviour to readjust/regulate the corrective action/control so that the system can achieve the desired objectives.

Feedback control system: a control system that deploys a feedback mechanism. This is also called a closed loop control system as shown in Figure 2.

Open loop control system: a control system without a feedback mechanism.

Regulator: a mechanism that can combine the input/users' instructions and feedback information to take corrective control actions to make the system behaviour to achieve its desired objectives.

Desired Trajectory: desired objectives that are normally specified by the user. In the context of MMOGs, they refer to gaming rules.

Disturbance: anything tending to push the system behaviour off the track is considered as a disturbance. A disturbance can occur within the system or from external environment.

Filter: a mechanism retrieving system state to deliver output perceived by the user.

System output: system behaviour perceived by the user.

Error: deviation of the system behaviour/output from the desired trajectory.

Error tolerance boundary: a range within which the error is considered acceptable by the system or user. This boundary is normally specified by the user.

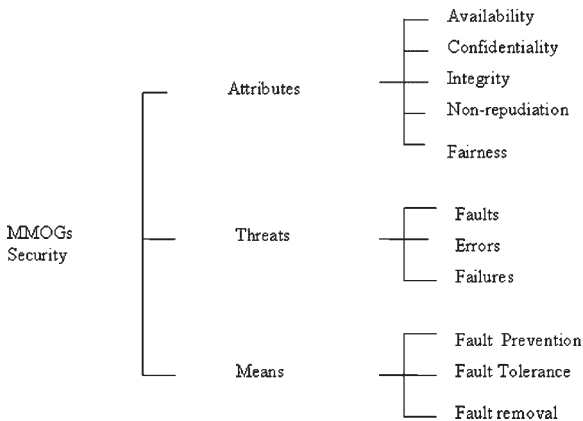


Fig. 1. MMOGs security tree.

Desired service: delivered system behaviour is at or close to the desired trajectory.

Correct service: delivered system behaviour is within the error tolerance boundary.

Service failure or failure: an event that occurs when system output deviates from the desired service and is beyond the error tolerance boundary.

Fault: Normally the hypothesised cause of an error is called fault [12]. It can be internal or external to the system. An error is defined as the part of the total state of the system that may lead to subsequent service failures. Observing that many errors do not reach the system's external state and cause a failure, Avizienis *et al.* [12] have defined active faults that lead to error and dormant faults that are not manifested externally.

Availability: readiness for correct service. The correct service is defined as delivered system behaviour that is within the error tolerance boundary.

Integrity: absence of improper system alterations or absence of malicious external disturbance which makes the system output off its desired service.

Confidentiality: property that data or information is not made available to unauthorised persons or processes. In the proposed framework, it refers to the property that unauthorised persons or processes will not get system output or be blocked by the filter.

Authenticity: ability to provide services with provable origin. In other words, the output can be verifiably linked to the system.

Non-repudiation: services provided cannot be disclaimed later. In our model, once the system provided an output, there is no way to deny it.

2.2. A Framework for Online Games Cheating

Note that authenticity and non-repudiation have not been addressed before in the unified framework of security and dependability [12] except in Hu *et al.* [13].

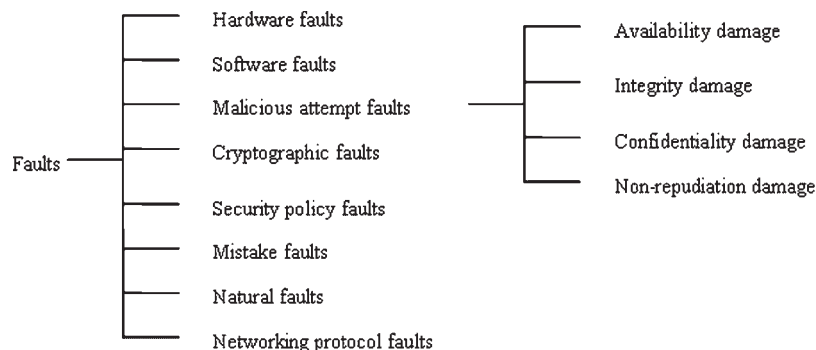


Fig. 3. Conventional elementary fault classes [13].

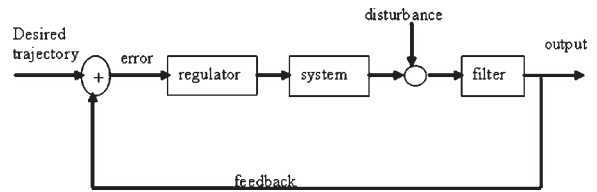


Fig. 2. Feedback control system.

These two attributes do not fit into conventional attributes of availability, confidentiality and integrity (ACI) of security. It seems difficult to include authenticity as a part of ACI. However, we observe that authenticity issue can lead to any of the availability problem, confidentiality problem and integrity problem. Hence it is more appropriate to express authenticity as an intermediate event towards security faults and also a means to achieving security and dependability. Similarly it is difficult to include the non-repudiation as part of ACI. However, unlike the authenticity issue non-repudiation problem does not necessarily lead to any of problems of availability, confidentiality and integrity. Therefore it seems appropriate to classify non-repudiation as an independent attribute Hu *et al.* [13]. A major concern of many systems is about the damage caused by faults, it is important to understand various faults and the countermeasures addressing the faults. In Hu *et al.* [13], elementary classes of faults have been suggested as shown in Figure 3.

This classification is suitable for addressing conventional dependability and security. However, it is not comprehensive enough to cover security issues in MMOGs. For instance, timing cheating in MMOGs such as *suppress-correct* cheat allows a cheater to gain an advantage by purposefully dropping update messages at the 'right' time. This attack causes peer players' enjoyment damage, which is commonly

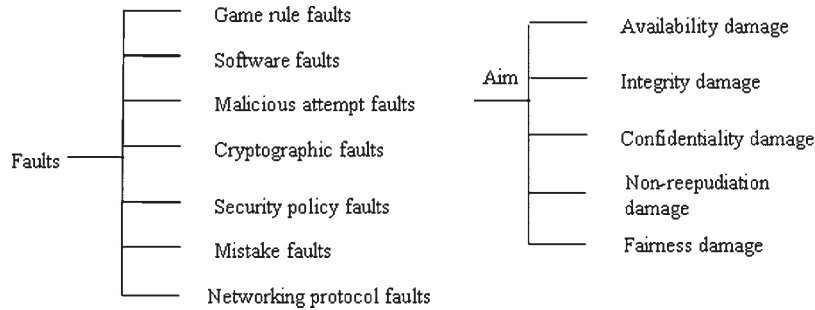


Fig. 4. Elementary fault classes of MMOGs.

termed ‘griefing behaviour’. Mulligan and Patrovsky [14] define a griever as ‘A player who derives his/her enjoyment not from playing the games, but from performing actions that detract from the enjoyment of the game by other players’. Taxonomy for grief play is given which covers four categories of griefing namely harassment, power imposition, scamming and greed play [15]. In general griefing behaviour occurs due to the existence of unfair play which is produced by the breaches of any combination of *availability*, *integrity*, *confidentiality* or *non-reepudiation* damages listed in Figure 3. However, as the final aim of this griefing behaviour leads to fairness damage, it is more appropriate to single out the fairness damage as a new item in the aim list. In this way, damages of availability, integrity, confidentiality and non-reepudiation serve as intermediate processes instead of an aim for griefing. By combination of common cheat forms shown in Table I and the conventional elementary fault classes shown in Figure 3, an MMOGs elementary fault class is illustrated in Figure 4.

In this paper, we are interested only in damages that have been caused by malicious actions. The right part shown in Figure 4 represents the aim of a malicious attempt. A malicious attempt has the objective of damaging the system. A fault is produced when this attempt is combined with other system faults. From the perspective of elementary security attributes, i.e. availability, confidentiality, integrity and non-reepudiation, we classify malicious attempt faults according to their aims that can be (1) intention to disrupt the service, e.g. denial of service attack; (2) attempt to access confidential information; (3) intention to improperly modify the system and (4) denying having gained services. Note that a malicious attempt fault is not a real fault unless it is combined with other faults. Therefore we can have the following classification of MMOGs faults as shown in Figure 5 where HMF represents human made faults, FUA represents faults

with unauthorised access and NFUA represents non-FUA. This tree and associate notations are similar to the tree presentation of faults proposed in Hu *et al.* [13] except the natural faults component is removed and a component of fairness faults are added in this paper. This framework can be used to describe the common cheating forms shown in Table I. We provide illustrative samples as follows.

- (A) Cheating by Exploiting Misplaced Trust: this cheat often involves tampering with game code, configuration data, or both, on the client side [5]. If we consider the game client code and configuration data as an integral part of the MMOGs system, this cheating can be attributed to integrity faults caused by the combination of malicious attempt faults caused by the combination of malicious attempt faults F1.2 and software faults F2.
- (B) Cheating by Collusion: for the case of ‘win-trading’ cheating, it can be represented as a fairness fault caused by the combination of malicious attempt fault caused by the combination of malicious attempt fault F1.5 and a game rule fault F3. For the case of collusion cheating in contract bridge, it can be considered as a fairness fault caused by the combination of the malicious attempt fault with aim of confidentiality damage F1.3 and a game rule fault F3.
- (C) Cheating by Abusing the Game Procedure: for the escape cheating by disconnection, it can be considered as a fairness fault caused by the combination of the malicious attempt fault with aim of fairness damage F1.5 and game rule fault F3. The intermediate fault is the availability fault.
- (D) Cheating Related to Virtual Assets: for the case of cheating by trading real money without delivering the virtual items, it can be considered as non-reepudiation fault caused by the malicious attempt fault with the aim of non-reepudiation damage F1.4 and security policy fault F5.

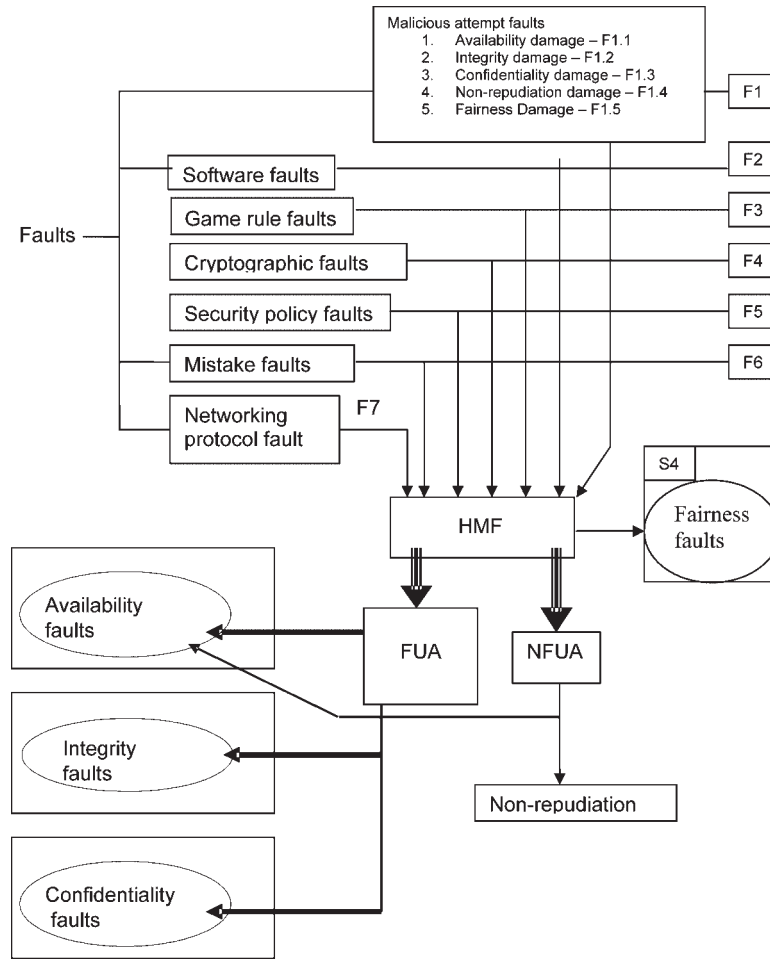


Fig. 5. Tree representation of MMOGs faults.

- (E) Cheating by Exploiting Machine Intelligence: for the cheating of using external computer program to assist gaming, it can be considered as a fairness fault caused by the combination of the malicious attempt fault with the aim of fairness damage F1.5 and the game rule fault F3.
- (F) Cheating by Modifying Client Infrastructure: for the cheating of ‘wall hack’, it can be considered as a fairness fault caused by the combination of the malicious attempt fault with the aim of fairness damage F1.5 and the software fault F2. Confidentiality damage F1.2 can also fit.
- (G) Cheating by Denying Service to Peer Players: this is similar to generic availability fault which can be caused by the combination of the malicious attempt fault with the aim of availability damage F1.1 and game rule fault F3.
- (H) Timing Cheating: suppress-correct cheat by dropping update messages at the ‘right’ time

can be considered as a fairness fault caused by the combination of the malicious attempt fault with the aim of fairness damage F1.5 and game rule fault F3.

- (I) Cheating by Compromising Passwords: it can be considered as a confidentiality fault caused by the malicious attempt fault with the aim of confidentiality damage F1.3 and the security policy fault F5.
- (J) Cheating by Exploiting Lack of Secrecy: it can be considered as a confidentiality fault caused by the malicious attempt fault with the aim of confidentiality damage F1.3 and the security policy fault F5.
- (K) Cheating by Exploiting Lack of Authentication: it can be considered as a fairness fault caused by the malicious attempt fault with the aim of fairness damage F1.5 and the security policy fault F5.

- (L) Cheating by Exploiting a Bug or Loophole: it can be considered as a fairness fault caused by the malicious attempt fault with the aim of fairness damage F1.5 and the security policy fault F5 or game rule F3.
- (M) Cheating by Compromising Game Server: it can be considered as an integrity fault caused by the malicious attempt fault with the aim of integrity damage F1.2 and the software fault F2.
- (N) Cheating Related to Internet Misuse: it can be considered as a fairness fault caused by the malicious attempt fault with either the aim of non-repudiation damage F1.4 or the aim of fairness damage F1.5 and the security policy fault F5.
- (O) Cheating by Social Engineering: it can be considered as a confidentiality fault caused by the malicious attempt fault with the aim of confidentiality damage F1.3 and the security policy fault F5.

For more generic interpretation of the fault tree representation, interested readers are referred to our prior work Hu *et al.* [13]. As illustrated in Figure 1, threats of MMOGs security are closely related to the faults discussed above. Therefore means of addressing fault prevention and removal are very important and should be included as an integral part of the MMOGs security issue. The next section is devoted to discussion of means and faults that are specific to MMOGs security. For discussion of more generic faults and means, interested readers are referred to our prior work Hu *et al.* [13].

3. Means and Counter-Cheating Measures for MMOGs

An integral part of MMOGs security issues concerns means, i.e. counter-cheating measures in preventing, identifying and removing cheats. As generic cheats can be mostly dealt with standard and conventional security measures, we focus our discussion on issues that are more specific to MMOGs security.

3.1. Means for Addressing Cheating by Exploiting Misplaced Trust

In MMOGs, a vendor has to provide a client application for the player to participate in the game. This game's client program is within the player's host and is therefore within player's reach for analysis,

modification and even replacement. This cheating can grant the cheater a huge advantage over honest players. An intuitive solution would be to perform all game related calculations on a game server and leave the game client being a very simple interaction front end for the game server. However scalability is already a serious issue in the MMOGs and such solution will further worsen the situation by placing more loads on the game server which renders it infeasible.

The public server architecture is proposed to address this scalability issue. In this architecture, the publisher sells the game client to players and gives away the server binary to anyone who wants to run a game server. Anyone has a choice to put up servers or take them down as they see fit. Hence the publisher's only involvement is in tracking the servers that are up and helping to connect individual players to running servers. For the games of first-person shooter such as Half-Life, this is the most popular architecture [16]. To address the difficult issue of security in this architecture, Chambers *et al.* [16] have proposed incentives schemes for potential public server abuse and public key cryptography for the content exchange and player and item authentication. One disadvantage is the significantly increased overhead placed on the clients and the weak reliability of the overall system.

Another opposite solution is to replicate game processing at all players' hosts. This is actually a peer-to-peer network case and all game clients process the same events and hence eliminate the possibility to cheat through game client modifications. However, this solution suffers from two additional problems. The first problem is the scarcity of resources in peer hosts. This would limit the complexity of the games as every client has to process the whole game state. Another problem is that all state information would be stored in all peers which might allow a cheater to gain perfect state information which he should not be allowed to. To reduce the infeasible burden of client hosting the whole game state in the MMOGs environment, distributed schemes have been proposed to divide the game space into regional groups [17–19]. To address the inherent issue of information exposure, Kabus and Buchman [18] propose a system design that can prevent storing data on unauthorised clients. In this system, it is required that every message is signed by the sender and every recipient keeps a log of all received messages for a certain period of time and signs it. Moench *et al.* [9] introduced a concept of mobile guides to prevent

modifications of game clients and also prevent cheating through information extraction or manipulation. The basic idea is that mobile guides are created and embedded with protection mechanisms within a trusted environment and then sent to the players' hosts. As each mobile guide has a limited lifetime and is created randomly each time, attackers will have difficulty breaking the security system within such a short time period. To solve the problem of potential bypassing the execution of the mobile guide, enforcement algorithms are embedded into the mobile guides so that an access to the game data requires an access key which can only be generated by execution of the associate mobile guide. In addition to the enforcement algorithms, protection algorithms are also embedded in the mobile guide to hide the game data. To ensure that both enforcement algorithms and data protection algorithms are executed, they are functionally and spatially entangled in the mobile guide. The main drawbacks of this work are that it incurs a high overhead for generating many mobile guides, and it has problems for games in which trusted servers are not available (e.g. Quake III Arena, Counterstrike, etc.).

3.2. Means for Addressing Cheating by Collusion

Collusion cheating refers to several players cooperating to gain unfair advantages over their opponents in online games. One common type of collusion cheating is to exchange information which their avatars could not have found directly. One typical example of such attacks has been widely seen in online bridge. In this online contract bridge game, four-person card players play between two pairs of partners. By rules, each player knows only a subset of 52 cards during the game. However, collusive cheaters can exchange card information over telephone, instant messaging, etc. that leads to substantial advantages over honest bridge players. Another typical example is 'win-trading' which is a collusion cheat widely encountered in the popular StarCraft game. In this 'win-trading' collusion treating, each of the cooperative players lost to the other alternatively in the ladder competition. The loss from the one would give the other a victory point, raising his ladder rank, and *vice versa*. Thus, both players could climb to top position in the ladder without playing a legitimate game [5].

Yan [20] discussed a solution by using randomised partners. The problem is that it will not be much useful if too many dishonest players are involved. As a matter of fact, 35% players admitted to online cheat-

ing in a survey [20,21]. Another issue is that people would like to choose someone who they are familiar with as the partner. A randomised partner could reduce the interest of participation of the game. Collusion cheating is perhaps the most challenging security issue in MMOGs. Much research effort will be needed in order to tackle this problem and solve it properly.

3.3. Means for Addressing Cheating by Abusing the Game Procedure

Normally this cheating can be carried out without any technical difficulties. One common example is the escaping cheating where the cheater can simply disconnect from the game system when she is going to lose [5]. There is no clear-cut solution to such a problem, because enforcing rules that prevent disconnection may result in users' dissatisfaction. On the other hand this problem is usually minimised by mitigating social factors: players tend to gather in *guilds* (or clans) in MMOGs that tend to judge unfavourably such forms of cheating. The social factor at stake here is reputation, and few guilds are interested in bad reputation [22]. Overall, such cheats are very specific to individual games. The solution will depend on the details of each game procedure, which is not the focus of this paper.

3.4. Means for Addressing Cheating Related to Virtual Assets

This is the case where a cheater might receive real money for the virtual item he has offered but she never delivers it as agreed. This problem can be resolved by the conventional non-repudiation technology by which the seller cannot deny what has been offered. The security policy on the trading platform needs to enforce this technology.

3.5. Means for Addressing Cheating by Exploiting Machine Intelligence

Online games are characterised by human players playing against or in cooperation with peer players. Due to the physical limitation of human beings, using game rule-restricted automated computer controlled programs for cheating can bring huge advantages for the cheaters. Such software applications are usually dubbed 'bots' [3]. Van *et al.* [23] have proposed completely automated public turing test to tell computers and humans apart (CAPTCHA) tests to address

this problem. A CAPTCHA test can generate and grade tests that most humans can pass yet current computer software cannot pass. The most widely used CAPTCHA tests rely on humans' ability to recognise randomly distorted text or images [3,8]. CAPTCHA tests can prevent bot participation effectively in long stateful games where players have to lose more if they fail the test [3]. However, CAPTCHA tests are disruptive by drawing players' attention away from the game and CAPTCHA tests can be outsourced to low-cost workers that specialise in solving them [3]. Golle and Ducheneaut [3] have proposed hardware-based CAPTCHA tests that can distinguish human players from bots based on humans ability to interact with the physical world. Typical human reaction events are pressing a button, moving a joystick and tapping on a touch-sensitivity screen.

Chen *et al.* [24] proposed a traffic analysis approach to identify MMORPG bots. By investigating the traffic generated by mainstream game bots and human players in Ragnarok Online (one of the most popular MMOGs), it is observed that the traffic is distinguishable by: (1) the regularity in the release time of client commands, (2) the trend and magnitude of traffic of traffic burstness in multiple time scales and (3) the sensitivity to network conditions. Based on this observation, four corresponding strategies are suggested to identify bots. The suggested schemes have been validated over the dataset collected from Ragnarok. It is also shown that the proposed methods are generalisable to other games and robust against counter-measures from bot developers.

3.6. Means for Addressing Cheating by Modifying Client Infrastructure

A common attack is that the cheater overrides the graphics driver behaviour for instance to make a wall transparent so that she can see through the wall, locating other players who are supposed to be hidden behind the wall. This is a popular cheat in some online games [5]. As there is no way to control the infrastructure of the player's host, the most effective way of protection is to introduce encryption and obfuscation techniques to hide sensitive data.

3.7. Means for Addressing Cheating by Timing

Many timing attacks are related to the dead reckoning which is a common synchronisation technique used to preserve the real-time quality of online games. Dead reckoning (DR) is a part of the distributed interactive

simulation (DIS) and high-level architecture (HLA) standards [25]. DR is used to compensate for variable network transmission latency and loss by allowing a client to guess the state of another player when updates are missing based on the last known vectors. In its basic form, the future position of a player is predicted via previous velocity and position. One common attack on DR is called *suppress-correct cheat*. In this cheat, the cheater can gain benefit by purposely dropping update messages at the 'right' time. A simple illustrated example is given below [10]: A sluggish player S is chasing a more agile player A. S begins pursuit, then drops $n - 1$ updates; A dead-reckons S's missing state but cannot confirm where S really is. For the n th update, S sends a false update that places S on the heels of A. As long as S sends plausible updates every n th update, A cannot confirm that S is cheating or playing fairly; S simply claims to be on a congested, lossy link. Therefore, fair play is indistinguishable from cheating when dead reckoning is used [10]. A simple protocol called Lockstep was proposed for peer-to-peer game play, and it has been proved that cheating is not possible with this protocol [10]. The Lockstep protocol suffers from serious performance penalties. An asynchronous synchronisation concept was later introduced to make the system serverless which offers anti-cheating proof in the presence of packet loss and less communication overheads.

4. Conclusions

In this paper, a taxonomy framework has been proposed to classify MMOGs. We have also used attributes, threats and means to describe MMOGs security which can provide a comprehensive coverage of issues arising from MMOGs security. The proposed classification framework can successfully derive all common cheat forms found in Reference [5]. We also discussed latest developments on the means of addressing MMOGs cheats. The work presented in this paper provides a comprehensive reference for the understanding of MMOGs security. As the proposed framework can provide a map showing the elementary sources of faults and possible paths leading to these faults from the sources, it can be useful for designing counter-measures to mitigate these faults. Various aspects and associated solutions discussed here will be helpful both for further academic research and to improve security in future games development.

Acknowledgement

The first author of this paper acknowledges the partial financial support from the Australia Research Council (ARC) Linkage Grant under Grant LP0455324 (Developing a Scalable Infrastructure for Embedded E-Security Incorporating Cryptography and Biometric Authentication).

References

- Software Association. Essential facts about the computer and video game industry, 2006. A digital copy can be downloaded at <http://www.theesa.com/archives/files/Essential%20Facts%202006.pdf>
- Industry transformed as one-third of games software revenues will be games generated online by 2011. Strategy Analytics, September 2007. Available online at <http://www.strategyanalytics.net/default.aspx?mod=PressReleaseViewer&a0=3569>
- Golle P, Ducheneaut N. Keeping bots out of online games. *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*; 2005 June 15–17, Valencia, Spain.
- Ogre to Slay? Outsource it to Chinese. D. Barobza, New York Times, 5 December 2005. Available online at <http://www.nytimes.com/2005/12/09/technology/09gaming.html?ex=1291784400&en=a723d0f8592dff2e&ci=5090&partner=rssuserland&emc=rss>
- Yan J, Rendell B. A systematic classification of cheating in online games. *NetGames 05*, October 10–11, 2005, Hawthorne, New York, USA, ACM, pp.1–9.
- David SB. Why cheating matters: cheating, game security, and the future of global on-line gaming business. *Proceedings of Game Developer Conference*, 2001.
- Yan J, Choi HJ. Security issues in online games. *The Electronic Library*, Vol. 20, No. 2, 2002.
- Golle P, Ducheneaut N. Preventing bots from playing games. *ACM Computers in Entertainment* 2005; 3(3): 1–10.
- Moench C, Grimen G, Midstraum R. Protecting online games against cheating. *Netgames'06*, October 30–31, 2006, Singapore, pp. 1–11.
- Baughman ME, Liberatore M, Levin BN. Cheat-proof payout for centralized and peer-to-peer gaming. *IEEE/ACM Transactions on Networking* 2007; 15(1): 1–13.
- Pritchard M. How to hurt the hackers: The scoop on Internet cheating and how you can combat it. *Information Security Bulletin* 2002. Available at http://www.gamasutra.com/features/20000724/pritchard_pfv.htm [14 January 2008].
- Avizienis A, Laprie JC, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 2004; 1(1): 11–33.
- Hu J, Bertok P, Tari Z. Taxonomy and framework for integrating dependability and security. In *Information Assurance: Dependability and Security in Networked Systems*, Qian Y, Tipper D, Krishnamurthy P, Joshi J (eds.) Elsevier: Burlington, 2007; 149–170.
- Mulligan J, Patrovsky B. Developing online games: an insider's guide, New Riders, Indiana, 2003.
- Foo CY, Koivisto EMI. Defining grief play MMORPGs: player and developer perceptions. *International Conference on Advances in Computer Entertainment Technology ACE 2004*, June 3–5, Singapore, 2004, pp. 245–250.
- Chambers C, Feng WC, Feng WC. Towards public servers MMOs. *The 5th Workshop on Network & System Support for Games, NetGames'06*, October 30–31, 2006, Singapore, pp. 1–7.
- Knutsson B, Lu H, Xu W, Hopkins B. Peer-to-peer support for massively multiplayer games. *Proceedings of INFOCOM'04*, Hong Kong, China, 2004.
- Kabus P, Buchmann AP. Design of a cheat-resistant P2P online gaming system. *DIMEA'07*, Perth, Western Australia, 2007, pp. 113–120.
- Chambers C, Feng WC, Feng WC, Saha D. Mitigating information exposure to cheaters in real time strategy games. *NOSSDAV'05*, June 13–14, 2005, Washington, USA, pp. 7–12.
- Yan J. Security design in online games. *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*, 2003.
- Greehill R, Diablo, and online multiplayer game's future. *GamesDomain Review*. Retrieved at <http://gamesdomain.com/gdreview/depart/jun97/diablo.html>
- The world of warcraft stratics web site. <http://wow.stratics.com/content/features/guides/ultimate/#16>
- von Ahn L, Blum M, Hopper NJ, Langford J. CAPTCHA: Using hard AI problems for security. *Proceedings of Euro-crypt'03*, 2003, pp. 294–311.
- Chen KT, Jiang JW, Huang P, Chu HH, Lei CL, Chen WC. Identifying MMORPG bots: a traffic analysis approach. *ACE06*, June 14–16, 2006, Hollywood, California, USA.
- Singhal S, Zyda M. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley Professional: Boston, 1999.