

# An Investigation of Cheating in Online Games

Cheating is rampant in current gameplay on the Internet but not well understood. In this article, the authors summarize various known cheating methods and define a taxonomy for online game cheating to help security experts and game developers alike.



Online games are one of the most popular Internet applications, but cheating has emerged as a notable phenomenon in current gameplay on the Internet. However, such cheating isn't as well understood as we might expect.

To address this lack of knowledge, we looked at various known cheating methods and defined a taxonomy of online game cheating with respect to underlying vulnerability (What is exploited?), consequence (What type of failure can be achieved?), and the cheating principal (Who is cheating?). This taxonomy systematically introduces cheat characteristics in online games and describes how they can arise. We intend it to be comprehensible and useful not only to security specialists but also to game developers, operators, and players who are less knowledgeable and experienced in security. We presented a preliminary version of this article elsewhere<sup>1</sup> that significantly extended our previous work<sup>2</sup>; here, we present a further refined version.

## **Security in Computer Games: An Overview**

Over the years, security has played varying roles in different games, ranging from early mainframe-based games through arcade, console, and PC games to the latest online games. The evolution of security requirements in these games confirms the well-known observation that security can mean different things in different application contexts.

### **Security in Mainframe, Arcade, and PC Games**

Although security started becoming an issue for com-

puter scientists in the 1960s, it was a concern only for operating system designers. No real security considerations were specific to computer games, which were mainly placed on the mainframe at the time.

The only plausible exception known to us was a unique access control feature implemented in the Cambridge Multiple Access System at Cambridge University in the mid 1960s. This feature controlled access to data via the identity of the program being used for access as well as—or instead of—through user identity. Examples using this unusual feature included the league table file, which “recorded people’s relative standing in a competitive guessing game, which was accessible to the game program only” (Roger Needham, personal communication, Oct. 2002).

Coin-operated arcade games emerged in the 1970s, and the physical security of coin boxes locked in the arcade game machines was a major security concern. People tackled this issue by using safer boxes and putting game machines in trustworthy or guarded places.

As with other content-critical industries, such as music or video, when content piracy issues expanded to threatening levels, copy protection became important, especially for PC games. From the beginning (the 1980s), PC games were especially vulnerable to piracy because they were typically shipped on standard removable storage media, such as floppy disks or CD-ROMs. Game companies adopted many diskette- or CD-ROM-based copy protection techniques for their games but achieved only mixed success, given that copy protection is a coevolutionary war between guardians and crackers.

JEFF YAN  
AND BRIAN  
RANDELL  
*Newcastle  
University, UK*

### **Security in Console Games**

Manufacturers also developed copy protection mechanisms for console games. However, console-game vendors' security efforts have been directed toward locking customers in to their hardware and making strategic plays in the market.

Nintendo appears to be the first console vendor to adopt security techniques for this purpose<sup>3</sup> and introduced practices that define the game-console industry to this day.

The security system that Nintendo introduced to its consoles implemented a simple lock-and-key concept. Each authentic Nintendo console and cartridge contained a security chip, and these chips communicated via a shared secret key. As long as the chip in the console and another in the cartridge were communicating, the system operated; otherwise, the system froze up. In this way, a console would reject any non-Nintendo game cartridges; if a player inserted a Nintendo cartridge into a console that didn't know the key, the game wouldn't run either.

Nintendo protected its security chip via both copyright and patent; the industry referred to it as a "lock out" chip. No one could manufacture their own games for this platform without Nintendo's approval. Nintendo used this advantage to enforce strict licensee agreements, censor games, and demand that it manufacture all cartridges for its console.

All this combined to make Nintendo the number one videogame manufacturer at the time—at its peak, it achieved a near-monopoly position in the industry, which it retained for nearly 10 years.<sup>3</sup>

Today, all major console vendors, including Microsoft and Sony, use security techniques to make strategic market plays, continuing the tradition Nintendo initiated.

### **Security in Online Games**

Online games' emergence has not only changed the way people play games but has also led to fundamentally changed security requirements for computer games. As (distributed) Internet applications, online games involve more complicated security issues than traditional computer games did, whereas some previous concerns present little problem.

For example, many online game vendors distribute their game software free, or with a symbolic fee, and charge users when they log in to play on the vendors' servers. This lets vendors (more or less) bypass the traditional headache of copy protection.

Instead, security for online games includes security for game-hosting systems in a networked environment, and it also covers issues such as privacy and access control. For commercial online games, security also involves the payment mechanism. These issues are relatively well understood, however, because other

Internet applications share them.

Recent research has suggested that online cheating, on the other hand, represents an important new security concern for computer game design.<sup>2,4-6</sup> This might seem like a surprising observation, but it's a natural consequence of such games' intrinsic properties. Interestingly, to get to this point, we had to wait almost 50 years after the appearance of the first computer game—in fact, until people played computer games in the same way they played their counterparts in the nonelectronic world.

We can sum up the latest developments by saying that while people for the past 50 years tried to cheat the system, now they're trying to cheat each other.

### **Cheating in Online Games**

We adopt the following definition for cheating (which refines our earlier version<sup>2</sup>):

Any behavior 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 (that is, the game service provider, who is not necessarily the developer of the game), the advantage or the target is one that the player is not supposed to have achieved.

Several common forms of cheating exist, and they have some general properties.

### **Common Cheating Forms**

We can classify common forms of cheating into 15 categories. We've marked those specific to or of special relevance to online games with an asterisk. The others are generic to all network applications, although they might have different names in different contexts, such as "attacks" or "intrusions." (See "Related Work in Creating Security Taxonomies" for more methods for classifying security concerns.)

**A. Exploiting misplaced trust.\*** Many cheats involve tampering with game code, configuration data, or both, on the client side.<sup>4</sup> Cheaters can modify their game client program or data, then replace the old copy with the revised one for future use. Alternatively, they can modify or replace code and data on the fly.

Cheaters can also tamper with their game client programs on the fly to access sensitive game states that are otherwise unavailable to players. Typical examples include the hacker program that displayed all army formation information in the popular Asian game *Si Guo* (or "Four States" in English),<sup>5</sup> and the "map hack" that dishonest players often use to reveal unexplored map areas on the display in real-time strategy games such as *Age of Empires*.

Such cheating occurs primarily due to misplaced

## Related Work in Creating Security Taxonomies

Several authors have attempted to define a framework for classifying and understanding online game cheating. For example, Steven Davis categorized traditional forms of casino cheating and discussed their potential counterparts in online games.<sup>1</sup> However, a casino isn't representative enough to reflect all forms of online game settings, in which cheating might occur with differing characteristics.

Matt Pritchard reported many real online cheating cases that have occurred in various games and classified them into a six-category framework.<sup>2</sup> However, his classification is ad hoc and not comprehensive. Indeed, many online game cheats don't readily fit into any of his categories.

Several papers investigate the definition of taxonomies for security vulnerabilities, attacks, or intrusions in a general setting. For example, Carl Landwehr and his colleagues constructed a classification of security flaws in software with respect to genesis (How did the flaw enter the system?), time of introduction (When did it enter the system?), and location (Where in the system is it manifested?),<sup>3</sup> but this classification largely focused on flaws in operating systems. Gary McGraw presented a taxonomy of soft-

ware coding errors.<sup>4</sup> Ulf Lindqvist and Erland Jonsson discussed the desired properties for a taxonomy and defined a taxonomy of intrusions with respect to intrusion techniques and results.<sup>5</sup> However, as we discuss in the main text, although a gameplayer can cheat by launching an attack or intrusion, cheating in online games can also have some unique manifestations.

### References

1. S.B. Davis, "Why Cheating Matters: Cheating, Game Security, and the Future of Global On-Line Gaming Business," *Proc. Game Developer Conf.*, 2001.
2. M. Pritchard, "How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It," *Information Security Bulletin*, Feb. 2001, pp 33–41.
3. C.E. Landwehr et al., "A Taxonomy of Computer Program Security Flaws," *ACM Computing Surveys*, vol. 26, no. 3, 1994, pp. 211–254.
4. G. McGraw, *Software Security: Building Security In*, Addison-Wesley, 2006.
5. U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions," *IEEE Symp. Security & Privacy*, IEEE CS Press, 1997, pp. 154–163.

trust—developers place too much trust in the client side, which in reality can't be trusted at all because cheating players have total control over their game clients. Countermeasures based on security-by-obscurity approaches, such as program obfuscation, eventually fail because they try to protect the wrong thing—that is, the game clients' binary codes, which are in fact under the cheaters' control.

**B. Collusion.\*** In online games, players can collude with each other to gain unfair advantages over their honest opponents. For example, so-called "win trading" was a collusion cheat widely seen in the popular StarCraft game, in which two cheaters colluded with each other as follows. Each lost to the other alternately in a ladder competition. The loss that one took would give the other a victory point, raising his ladder rank, and vice versa. Thus, both of them could climb to top positions in the ladder without playing a legitimate game.

Collusion cheating has also been widely seen in online contract bridge, which is a four-person card game played between two pairs of partners. Unlike chess, in which all pieces are on the board and known to each side, bridge is a game with hidden information. Each player normally sees only a subset of all 52 cards during gameplay. However, by illicitly exchanging card information over the telephone, instant messenger, or the like, collusive cheaters can gain huge advantages over honest bridge players. We discuss this collusion cheat and various other collu-

sion scenarios, as well as their countermeasures, in earlier work.<sup>5</sup>

**C. Abusing game procedure.\*** Players can carry out this form of cheating without any technical sophistication because the cheater simply abuses the game's operating procedure. One common case we've observed in many online games is *escaping*: cheaters disconnect themselves from the game system when they're about to lose.

Another example is *scoring cheating*, which we observed in the popular online Go community. When a game is finished in Go, the players must identify and remove "dead" stones by hand before the system can determine which side wins. During this scoring process, however, cheating players can stealthily remove their opponents' "alive" stones, thus overturning the game result. (When each side's territory is similar in size, cheated players might easily overlook the cheat, especially if they aren't strong players.)

**D. Cheating related to virtual assets.\*** In some online games, players can trade virtual characters and items they've acquired for real money. A cheater might offer a virtual item and receive real money for it, but never deliver it as agreed.

**E. Exploiting machine intelligence.\*** Cheating players can sometimes exploit artificial intelligence (AI) techniques. For example, advances in computer chess research have produced many programs that can compete

with human players at the master level. When playing chess online with other human players, cheaters can look for the best candidates for their next move by stealthily running a strong computer chess program.

**We can sum up the latest developments by saying that, whereas people for the past 50 years tried to cheat the system, now they're trying to cheat each other.**

This is cheating in this particular situation due to machine intelligence's superiority over an ordinary human. It can occur in many other online games (including online versions of traditional board and card games), depending on two factors: the game's properties—that is, whether the game can be modeled as a computable problem—and the maturity of AI research into such games.

**F. Modifying client infrastructure.\*** Without modifying game programs, configurations, or data on the client side, players can cheat by modifying the client infrastructure, such as device drivers in their operating systems. For example, they can modify a graphics driver to make a wall transparent so that they can see through it, locating other players who are supposed to be hidden behind it. This is the so-called *wall hack*, a popular cheat in some online games.

**G. Denying service to peer players.** Cheaters can gain advantages by denying service to their peer players. For example, we've observed that cheaters can delay responses from their opponents by flooding the opponents' network connection. Other peer players would then be cheated into believing that something was wrong with the victim's network connection and agree to kick him or her out of the game to avoid stalling the session.

**H. Timing cheating.\*** In some real-time online games, cheating players can delay their own moves until they know all their opponents' moves, and thus gain a huge advantage.<sup>7</sup> This *look-ahead cheat* is one type of timing cheating; other types include the *suppress-correct cheat*, which lets a cheater gain an advantage by purposefully dropping update messages at the "right" time.<sup>7</sup>

**I. Compromising passwords.** Passwords are often key to much of or all the data and authorization players have in an online game system. By compromising a password, a cheater can access the victim's data and authorization in the game. Due to the limitations of human memory, some players tend to choose easy-to-

remember passwords, such as phone numbers, birthdays, or names of friends or family, or common words in their native languages. Cheaters can often easily guess such weak passwords.

**J. Exploiting lack of secrecy.** When game programs exchange communication packets in plaintext format, cheaters can eavesdrop on these packets and insert, delete, or modify game events or commands transmitted over the network. This form of cheating can also result in compromised passwords if players send their passwords to the server in plaintext.

**K. Exploiting lack of authentication.** If no proper mechanism is in place for authenticating a game server to clients, cheaters can collect many ID-password pairs from legitimate players by setting up a bogus game server. Similarly, if a proper mechanism doesn't exist for authenticating a client, cheaters can exploit this to gain advantages. For example, it's critical that the system reauthenticate players before executing password changes for them. Otherwise, when players leave their computers temporarily unattended with a game session open—in an Internet cafe, for example, which is a primary place people in countries such as China and Korea play online games—cheaters who can physically access players' machines might stealthily change the players' password and exploit that changed password afterward.

**L. Exploiting a bug or loophole.** Here, cheaters exploit a defect in game programs or the game design itself without having to modify game code or data. Once discovered, such a defect will give knowledgeable players a major advantage. Lucasfilm's *Habitat*, one of the first multi-user virtual environments, had such a loophole. Due to an inadvertent pricing error, people in the game could sell virtual items to a pawn shop at a higher price than they paid to get them from a vending machine. By shuttling back and forth between the vending machine and the pawn shop, some players became (virtual) millionaires overnight.<sup>8</sup>

**M. Compromising game servers.** Cheaters can tamper with game server programs or change their configurations once they've obtained access to the game host systems. Examples of such cheating cases are available elsewhere.<sup>4</sup>

**N. Internal misuse.** A game operator usually has system administrator privileges that insiders—game operator employees—can easily abuse. For example, they can generate super characters or other valuable virtual items by modifying the game database on the server side.



**O. Social engineering.** Cheaters often attempt to trick players into believing something attractive or annoying has happened to them, requiring the honest players to enter an ID and password. Many major game operators, such as Blizzard, have published guidelines on avoiding such scams on their Web sites, implying that this kind of cheating has been a real problem.

### **Atomic vs. Complex Cheats**

The list we just presented attempts to be comprehensive, but it isn't necessarily disjoint, so a given cheat might fall into more than one category. Ideally, we could define a list of common cheating forms that is disjoint, but this has not proved possible.

Although each listed form can be an independent cheat, an actual cheating case might be complex and involve multiple forms of cheating. For example, the Pogo cheat<sup>5</sup> involved two dishonest players who collusively abused a voting protocol to gain advantages. This was a cheat due to collusion that abused game procedure and also exploited a loophole in the game system design.

Another example is the *hit-and-run* cheat that we observed in online Go games. Go is a time-critical game played between two people. The Go server counts the time each player spends in the game, and a player who runs out of time automatically loses. Many online players choose to play 25 moves in 10 minutes or less, and it's usual for one to play five stones in the last 10 seconds. So, cheating players can easily defeat opponents by timing them out with a well-timed flooding attack. This denies service to peer players, and the hit-and-run cheaters can use this *timeout* cheat while also abusing the game procedure as follows.

Some Internet Go services implemented a penalty rule to fight against the escaping cheat: players who disconnect themselves will lose their unfinished game unless they return to finish it within a limited period. However, a hit-and-run cheater can exploit this rule by flooding one opponent so that the system records that this opponent disconnected the game. Then, the cheater doesn't log on again until the penalty period has passed. The players can't finish that game in time, and the opponent automatically loses points for it.

### **Online Cheating Taxonomy**

Our taxonomy for online game cheating is three dimensional; as mentioned, we classify cheating via the underlying vulnerability, the cheating consequence, and the cheating principal. Our classification is intentionally reminiscent of the well-known dependability taxonomy.<sup>9,10</sup>

Table 1 shows the taxonomy details by vulnerability, possible failure, and exploiter (cheating principal), respectively. Every cheating form appears at least once in each of these categories.

### **Vulnerability**

Some cheats exploit design inadequacies in game systems, and others don't. For example, cheating by exploiting a bug or loophole exploits inadequacies in the game design, implementation, or both. However, social engineering doesn't exploit any technical design inadequacies. We classify the vulnerabilities exploited via online cheating into two divisions: *system design inadequacy*, which involves a technical design flaw that arises during system development, and *human vulnerability*, involving those who operate or play online games.

System design inadequacy has two subdivisions: *inadequacy in the game system* and *inadequacy in the underlying systems*. Online games are applications running on top of an underlying networking and operating system. A cheater can exploit a flaw in a game system, a flaw in its underlying networking or operating system, or both.

Exploiting misplaced trust, lack of secrecy or authentication, timing cheating, and exploiting a bug or design loophole take advantage of technical inadequacies in the game system and belong to the first subdivision. Collusion, abusing the game procedure, and exploiting machine intelligence can also ultimately arise from technical design failures in the game system, and due to "the inability to foresee all the situations the system will be faced with during its operational life, or the refusal to consider some of them"<sup>9</sup> for reasons such as time to market. They also belong to the first division.

Two common cheating forms—modifying client infrastructure and compromising game servers—belong to the second subdivision. Specifically, the former occurs on the game client side. However, rather than exploit the game application itself, it modifies the system infrastructure—for example, a device driver that's part of the operating system. Similarly, a cheater compromising a game server usually breaks into it by exploiting a flaw in the operating system or network protocols on the server side. (A game server program might have flaws that cheaters can remotely exploit, but we haven't seen such cases in real life.)

Denying service to peer players usually exploits some inherent network layer weakness. However, cheaters can also exploit a design inadequacy in the game system alone. For example, a cheat that occurred in the Firestorm game<sup>4</sup> exploited a buffer-overflow condition in the game program to disconnect all players. Another example is the so-called "spawn point camping" cheat popular in some real-time shooting games. A spawn point is a place in the game where players create their avatars at the beginning of gameplay (spawning) and recreate the avatar immediately after its death (respawning). By waiting near the spawn point, cheaters can easily kill players as they

**Table 1. Online game cheating classification.**

| CLASSIFICATION OF VARIOUS TYPES OF CHEATING | VULNERABILITIES          |        | POSSIBLE FAILURES  |            |                     |                |                                     | EXPLOITERS  |                       |                |        |
|---|--------------------------|--------|--------------------|------------|---------------------|----------------|-------------------------------------|-------------|-----------------------|----------------|--------|
|   | SYSTEM DESIGN INADEQUACY | PEOPLE | FAIRNESS VIOLATION | MASQUERADE | INTEGRITY VIOLATION | SERVICE DENIAL | THEFT OF INFORMATION OR POSSESSIONS | INDEPENDENT | COOPERATIVE           |                |        |
|   |                          |        |                    |            |                     |                |                                     |             | In underlying systems | In game system | Player |
| A. Exploiting misplaced trust               |                          | •      |                    |            | •                   |                | •                                   | •           |                       |                |        |
| B. Collusion                                |                          | •      | •                  |            |                     |                | •                                   |             |                       | •              |        |
| C. Abusing the game procedure               |                          | •      | •                  |            |                     |                |                                     | •           |                       |                |        |
| D. Cheating related to virtual assets       |                          |        | •                  |            |                     |                |                                     | •           |                       |                |        |
| E. Exploiting machine intelligence          |                          | •      | •                  |            |                     |                |                                     | •           |                       |                |        |
| F. Modifying client infrastructure          | •                        |        |                    |            | •                   |                |                                     | •           |                       |                |        |
| G. Denying service to peer players          | •                        | •      |                    |            |                     | •              |                                     | •           |                       |                |        |
| H. Timing cheating                          |                          | •      | •                  |            | •                   |                |                                     | •           |                       |                |        |
| I. Compromising passwords                   |                          |        | •                  |            |                     |                | •                                   | •           |                       |                |        |
| J. Exploiting lack of secrecy               |                          | •      |                    |            | •                   |                | •                                   | •           |                       |                |        |
| K. Exploiting lack of authentication        |                          | •      |                    | •          |                     |                |                                     | •           |                       |                |        |
| L. Exploiting a bug or design loophole      |                          | •      | •                  |            |                     |                |                                     | •           |                       |                |        |
| M. Compromising game servers                | •                        |        |                    |            | •                   |                |                                     | •           |                       |                |        |
| N. Internal misuse                          |                          |        |                    | •          |                     | •              |                                     |             | •                     |                | •      |
| O. Social engineering                       |                          |        |                    | •          |                     |                | •                                   | •           |                       |                |        |

spawn or respawn and prevent them from joining the action. This is indeed a service denial that exploits a design inadequacy in the game system. A more cautious design could, for instance, protect newly (re)spawned players by granting them immunity to ammunition for a short period so that they could move to a safe place. (During this period, they wouldn't be able to harm other players.) So, this form of cheating is included in both subdivisions.

Other cheating forms, such as social engineering, password compromising, cheating related to virtual assets, and internal misuse, are only marginally related to any technical design inadequacy. Instead, the first three forms largely exploit innocent players' vulnerabilities, and the fourth involves game operator vulnerability (involving insiders).

### Consequence

We base our cheating consequence classification largely on the four traditional computer security aspects:

confidentiality (preventing unauthorized information disclosure), integrity (preventing unauthorized information modification), availability (preventing unauthorized information withholding), and authenticity (assuring a remote user's identity regardless of that user's host). A confidentiality breach results in *theft of information or possessions*, an integrity breach results in the *improper modification of game characteristics*—that is, an *integrity violation*—an availability breach results in service denial, and an authenticity breach allows cheaters to disguise their identity (also known as a *masquerade*).

Collusion, compromising passwords, or social engineering can result in information or possession theft in a game; exploiting a lack of authentication results in a masquerade. Denying service to peer players involves selective service denials, but compromising game servers—by modifying client infrastructure or via internal misuse—usually involves improperly modifying game characteristics, or integrity failure.

When done only to eavesdrop on communications,

**Table 2. Distribution of observed cheating forms in the vulnerability–consequence matrix.**

| VULNERABILITY \ CONSEQUENCE                 | INFORMATION THEFT | SERVICE DENIAL | INTEGRITY FAILURE | MASQUERADE | FAIRNESS VIOLATION |
|---|-------------------|----------------|-------------------|------------|--------------------|
| Design inadequacy in the game system        | A, B, J           | G              | A                 | K          | C, E, H, L         |
| Design inadequacy in the underlying systems |                   | G              | F, M              |            |                    |
| Vulnerability in player                     | I, O              |                |                   |            | D                  |
| Vulnerability in insider                    |                   |                | N                 |            |                    |

exploiting lack of secrecy results in information theft. But when a cheater also tries to exploit such stolen information by inserting, deleting, or modifying game events or commands, this form of cheating will also cause an integrity violation. Moreover, as shown in the previous section, exploiting misplaced trust can lead to information theft, an integrity violation, or both.

However, these traditional computer security aspects are insufficient for covering all online game cheating’s consequences. For example, neither the cheat that exploited the erroneous pricing bug in Habitat or the win-trading collusion in StarCraft violated confidentiality, availability, integrity, or authenticity.

We’ve therefore introduced *fairness* between peer players as an additional aspect for understanding online game cheating; a fairness breach results in a *fairness violation*. Abusing the game procedure, exploiting a bug or design loophole, exploiting machine intelligence, or cheating related to virtual assets can all result in a fairness violation. As the win-trading case demonstrates, collusion can also result in a fairness violation. Moreover, timing cheating usually involves improperly modifying game characteristics and leads directly to a fairness violation.

**Cheating Principal**

A player can cheat single-handedly in either single- or multiplayer online games, whereas in multiplayer games, two or more players can collaborate to cheat. Furthermore, a player can also collude with an insider the game operator employs. The cheating principal’s identity is the third dimension in our classifications, and it provides a way to distinguish *cooperative* cheats from their *independent* counterparts.

The cooperative division includes two subdivisions: *multiple players* covers cheats, such as collusion, that require two or more players cooperatively, whereas *operator and player* accommodates cheating committed by at least one player and an insider—this typically involves collusion and game-specific internal misuse.

The independent division also includes two subdivisions: *game operator* covers cheating related to internal misuse, in which no collusion between a player and insider is involved. One example is an insider who’s also a player. (As we discuss in our previous

work,<sup>5</sup> house cheating that a game operator alone orchestrates can also occur; however, it’s beyond the online cheating definition we use in this article.) *Single player* accommodates all cheating forms that a player can commit single-handedly.

**Discussion**

Our taxonomy brings out a systematic view of online cheating from which we can make several observations.

First, online cheaters have exploited vulnerabilities in both computer systems and humans involved with the games. Cheaters can exploit both design inadequacies in game systems and weaknesses in their underlying system infrastructures. They can also exploit innocent players and corrupt insiders.

Second, the cheating principal classification shows that one player independently can commit most current game cheating but that some cheating involves collusion with peer players or an insider. We can find security breaches from a single user or due to user and insider cooperation in many contexts. However, it appears that with the possible exception of online auctions, collusion between peer users in other contexts isn’t as serious a problem as it is in online games.

Third, the consequence classification demonstrates that cheating is, in fact, largely due to various security failures. We can see this more clearly by examining the distribution of each common cheating form in the orthogonal dimensions of vulnerabilities and consequences. Table 2 constructs this distribution matrix, with vulnerability and consequence displayed in rows and columns, respectively, and cheating forms represented with the labels we assigned in a previous section. The matrix clearly shows that most types of online game cheats involve information theft, improperly modifying game characteristics, or fairness violations, and that they largely exploit flaws in the game systems.

As a result of our taxonomic analysis using Table 2, we were able to correct a mistake in a previous version of this article—namely, we found that we missed a type of service denial cheating that exploits design inadequacies only in the game system.

We can also use Table 2 to suggest novel forms

of cheating that will likely occur in the future. For example, cheats that lead to masquerade, information theft, or fairness violations and that occur due to design inadequacies in their underlying systems will probably occur in the future, although we can't yet know how they will manifest themselves.

**W**e've found that cheating in online games is due largely to various security failures. However, the four traditional aspects of security are insufficient to explain these cheats and their consequences. Instead, fairness becomes a vital additional aspect.

Fairness and its enforcement also appear to be a proper perspective for understanding security's overall role in applications such as online games. On the one hand, fair play is essential to any game, and online gaming is no exception. Fairness should be an inherent concern in its design. On the other hand, online play-

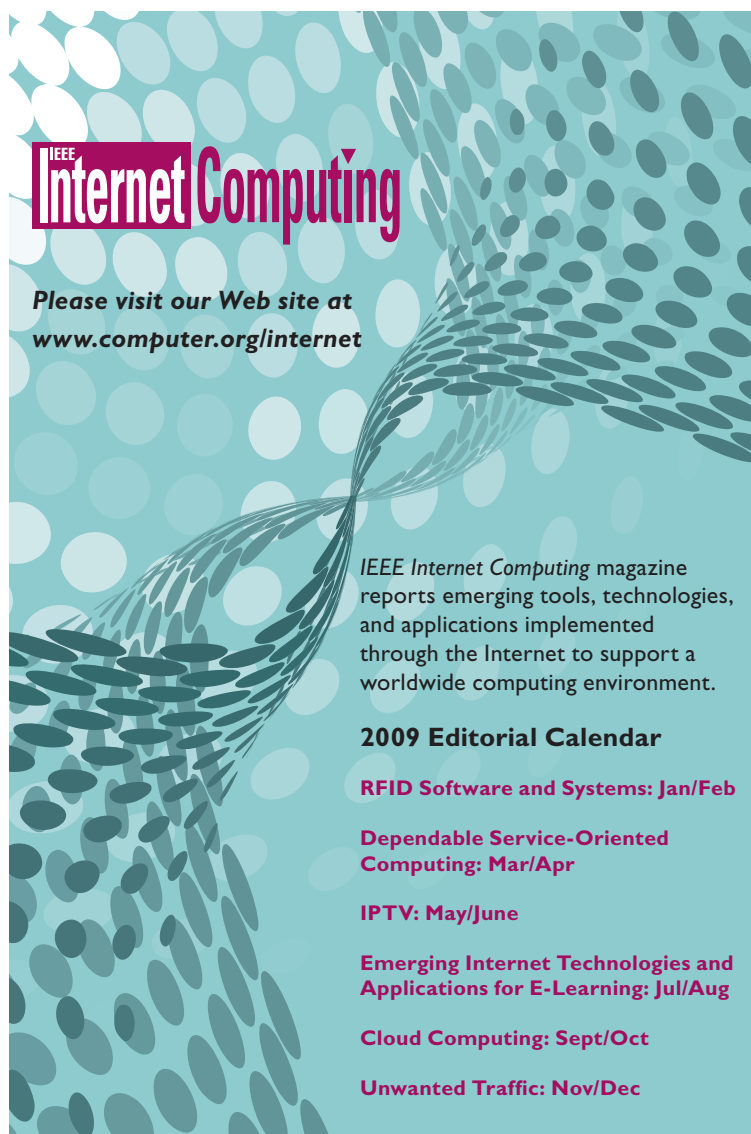
ers usually don't know each other and are often scattered across different physical locations. So, the social structures preventing or discouraging cheating in the nonelectronic world are no longer in place for online games. Instead, security (together with other technical means) becomes essential to enforcing fairness. □

### References

1. J. Yan and B. Randell, "A Systematic Classification of Cheating in Online Games," *Proc. 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames 05)*, ACM Press, 2005, pp. 1–9.
2. J. Yan and H.J. Choi, "Security Issues in Online Games," *The Electronic Library*, vol. 20, no. 2, 2002, pp. 125–133.
3. D. Sheff, *Game Over*, Hodder & Stoughton, 1993.
4. M. Pritchard, "How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It," *Information Security Bulletin*, Feb. 2001, pp. 33–41.
5. J. Yan, "Security Design in Online Games," *Proc. 19th Ann. Computer Security Applications Conf.*, IEEE CS Press, 2003, pp. 286–297.
6. G. Hoglund and G. McGraw, *Exploiting Online Games*, Addison-Wesley, 2007.
7. N. Baughman and B. Levine, "Cheat-Proof Payout for Centralized and Distributed Online Games," *Proc. IEEE INFOCOM Conf. Computer Communications*, IEEE Communications Soc., 2001, pp. 104–113.
8. C. Morningstar and F.R. Farmer, "The Lessons of Lucasfilm's Habitat," *Cyberspace: First Steps*, M. Benedikt, ed., MIT Press, 1990, pp. 273–302.
9. J.C. Laprie, ed., *Dependability: Basic Concepts and Terminology*, Springer, 1992.
10. A. Avizienis et al., "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable and Secure Computing*, vol. 1, no. 1, 2004, pp. 11–33.

**Jeff Yan** is a member of the computer science faculty at Newcastle University, where he leads research on systems and usable security. His recent work includes graphical passwords, the robustness and usability of CAPTCHAs, and security in online games. Yan has a PhD in computer security from Cambridge University. He's a contributor to *Security and Usability: Designing Secure Systems that People Can Use* (O'Reilly, 2005), and was on the program committee for the first *Symposium on Usable Privacy and Security (SOUPS)*. Contact him at [jeff.yan@ncl.ac.uk](mailto:jeff.yan@ncl.ac.uk).

**Brian Randell** is an emeritus professor of computing science and a senior research investigator at Newcastle University. His research interests include system reliability and security, and the history of computers. Randell has a BSc in mathematics from Imperial College, London, and a DSc in computing science from the University of London. He is coauthor or editor of seven books and is a member of the ACM A.M. Turing Award Committee. Contact him at [brian.randell@ncl.ac.uk](mailto:brian.randell@ncl.ac.uk).



**IEEE Internet Computing**

Please visit our Web site at [www.computer.org/internet](http://www.computer.org/internet)

IEEE Internet Computing magazine reports emerging tools, technologies, and applications implemented through the Internet to support a worldwide computing environment.

**2009 Editorial Calendar**

**RFID Software and Systems: Jan/Feb**

**Dependable Service-Oriented Computing: Mar/Apr**

**IPTV: May/June**

**Emerging Internet Technologies and Applications for E-Learning: Jul/Aug**

**Cloud Computing: Sept/Oct**

**Unwanted Traffic: Nov/Dec**