Chapter 10: Cheating Prevention
Cheating

- traditional cheating in computer games
  - cracking the copy protection
  - fiddling with the binaries: boosters, trainers, etc.
- here, the focus is on multiplayer online games
  - exploiting technical advantages
  - exploiting social advantages
- cheaters’ motivations
  - vandalism and dominance
  - peer prestige
  - greed
The goals of cheating prevention

- protect the sensitive information
  - cracking passwords
  - pretending to be an administrator
- provide a fair playing field
  - tampering the network traffic
  - colluding with other players
- uphold a sense of justice inside the game world
  - abusing beginners
  - gangs
Network security

- Military
  - private networks $\rightarrow$ no problem

- Business, industry, e-commerce,…
  - ‘traditional’ security problems

- Entertainment industry
  - multiplayer computer games, online games
  - specialized problems
Taxonomy of online cheating 1(4)

- Cheating by compromising passwords
  - dictionary attacks
- Cheating by social engineering
  - password scammers
- Cheating by denying service from peer players
  - denial-of-service (DoS) attack
  - clog the opponent’s network connection
Taxonomy of online cheating 2(4)

- Cheating by tampering with the network traffic
  - reflex augmentation
  - packet interception
  - look-ahead cheating
  - packet replay attack

- Cheating with authoritative clients
  - receivers accept commands blindly
    - requests instead of commands
    - checksums from the game state
Taxonomy of online cheating 3(4)

- Cheating due to illicit information
  - access to replicated, hidden game data
  - compromised software or data
- Cheating related with internal misuse
  - privileges of system administrators
  - logging critical operations into CD-ROMs
- Cheating by exploiting a bug or design flaw
  - repair the observed defects with patches
  - limit the original functionality to avoid the defects
  - good software design in the first place!
Taxonomy of online cheating 4(4)

- **Cheating by collusion**
  - two or more players play together without informing the other participants
  - one cheater participates as two or more players

- **Cheating related to virtual assets**
  - demand $\Rightarrow$ supply $\Rightarrow$ market $\Rightarrow$ money flow $\Rightarrow$ cheating

- **Cheating by offending other players**
  - acting against the ‘spirit’ of the game
Breaking the control protocol: Maladies & remedies

- **malady**: change data in the messages and observe effects
- **remedy**: checksums (MD5 algorithm)
- **malady**: reverse engineer the checksum algorithm
- **remedy**: encrypt the messages
- **malady**: attack with packet replay
- **remedy**: add state information (pseudo-random numbers)
- **malady**: analyse messages based on their sizes
- **remedy**: modify messages and add a variable amount of junk data to messages
MD5 algorithm

- message digest = a constant length ‘fingerprint’ of the message
- no one should be able to produce
  - two messages having the same message digest
  - the original message from a given message digest
- R. L. Rivest: MD5 algorithm
  - produces a 128-bit message digest from an arbitrary length message
- collision attack: different messages with the same fingerprint
- finding collisions is (now even technically!) possible
  - what is the future of message digest algorithms?
Illicit information

- access to replicated, hidden game data
  - removing the fog of war
  - compromised graphics rendering drivers
- cheaters have more knowledge than they should have → passive cheating
- compromised software or data
- counter-measures in a networked environment
  - centralized: server maintains integrity among the clients
  - distributed: nodes check the validity of each other’s commands to detect cheaters
Exploiting design defects

- what can we do to poor designs!
  - repair the observed defects with patches
  - limit the original functionality to avoid the defects
- client authority abuse
  - information from the clients is taken face-value regardless its reliability
- unrecognized (or unheeded) features of the network
  - operation when the latencies are high
  - coping with DoS and other attacks
Denial-of-service (DoS) attack

- **Attack types:**
  - logic attack: exploit flaws in the software
  - flooding attack: overwhelm the victim’s resources by sending a large number of spurious requests

- **Distributed DoS attack:** attack simultaneously from multiple (possibly cracked) hosts

- **IP spoofing:** forge the source address of the outgoing packets

- **Consequences:**
  - wasted bandwidth, connection blockages
  - computational strain on the hosts
Analysing DoS activity

- Backscatter analysis
- Spoofing using random source address
- A host on the Internet receives unsolicited responses
- An attack of $m$ packets, monitor $n$ addresses
- Expectation of observing an attack:

$$E(X) = \frac{nm}{2^{32}}$$
DoS: Selected results

- Three week-long logging periods, February 2001
- >12,000 attacks, >5,000 distinct targets
- Significant number of attacks were directed against
  - home machines
  - users running Internet Relay Chat (IRC)
  - users with names that are sexually suggestive or incorporate themes of drug use
  - users supporting multiplayer games
- In addition to well-known Internet sites, a large range of small and medium sized businesses were targeted

DoS: Most attacked top-level domains

Look-ahead cheating

\[ a_1 = \text{Rock} \]
\[ p_1 \]
\[ s = 0 \]
\[ a_2 = \text{Paper} \]
\[ p_2 \]
\[ s = 2 \]
Two problems

- delaying one’s decision
  - announce own action only after learning the opponent’s decision
  - one-to-one and one-to-many
- inconsistent decisions
  - announce different actions for the same turn to different opponents
  - one-to-many
Lockstep protocol

1. Announce a commitment to an action.
   - commitment can be easily calculated from the action but the action cannot be inferred from the commitment
   - formed with a one-way function (e.g., hash)

2. When everybody has announced their commitments for the turn, announce the action.
   - everybody knows what everybody else has promised to do

3. Verify that the actions correspond to the commitments.
   - if not, then somebody is cheating…
**Lockstep protocol**

\[ a_1 = \text{Rock} \]
\[ c_1 = H(a_1) = 4736 \]
\[ c_2 = 1832 \]
\[ a_1 = \text{Rock} \]
\[ a_2 = \text{Paper} \]
\[ H(a_2) = 5383 \neq c_2 \]
Loosening the synchronization 1(2)

- the slowest player dictates the speed
  - short turns
  - time limits for the announcements

- asynchronous lockstep protocol
  - sphere of influence: synchronization is needed only when the players can affect each other in the next turn(s)
  - otherwise, the players can proceed asynchronously
Loosening the synchronization 2(2)

- pipelined lockstep protocol
  - player can send several commitments which are pipelined
  - drawback: look-ahead cheating if a player announces action earlier than required

- adaptive pipeline protocol
  - measure the actual latencies between the players
  - grow or shrink the pipeline size accordingly
Drawbacks of the lockstep protocol

- requires two separate message transmissions
  - commitment and action are sent separately
  - slows down the communication
- requires a synchronization step
  - the slowest player dictates the pace
    - improvements: asynchronous lockstep, pipelined lockstep, adaptive pipeline lockstep
- does not solve the inconsistency problem!
Idea #1: Let’s get rid of the repeat!

- send only a single message
  - but how can we be sure that the opponent cannot learn the action before announcing its own action?
- the message is an active object, a delegate
  - program code to be run by the receiver (host)
  - delegate’s behaviour cannot be worked out by analytical methods alone
  - guarantees the message exchange on a possibly hostile environment
- delegate provides the action once the host has sent its own action using the delegate
Threats

- what if the host delays or prevents the delegate’s message from getting to its originator?
  - the host will not receive the next delegate until the message is sent
- what if the originator is malicious and the delegate spies or wastes the host’s resources?
  - sandbox: the host restricts the resources available to the delegate
- how can the delegate be sure that it is sending messages to its originator?
  - communication check-up
Communication check-up

- the delegate sends a unique identification to its originator
  - static and dynamic information
- the delegate waits until the originator has responded correctly
- check-ups are done randomly
  - probability can be quite low
  - host cannot know whether the transmission is the actual message or just a check-up
Idea #2: Peer pressure

- players gossip the other players’ actions from the previous turn(s)
- compare gossip and recorded actions; if there are inconsistencies, ban the player
  - cheating is detected only afterwards
  - gossiping imposes a threat of getting caught
- gossip is piggybacked in the ordinary messages
  - no extra transmissions are required
- how to be sure that the gossip is not forged?
  - rechecking with randomly selected players
How much is enough?

- example: 10 players, 60 turns, 1 cheater who forges 10% of messages, gossip from one previous turn
  - 1% gossip: $P(\text{cheater gets caught}) = 0.44$
  - 5% gossip: $P(\text{cheater gets caught}) = 0.91$
  - 10% gossip: $P(\text{cheater gets caught}) = 0.98$

- example: 100 players, 60 turns, 1 cheater who forges 10% of messages
  - 1% gossip: $P(\text{cheater gets caught}) = 0.98$

- example: 10 players, 360 turns, 1 cheater who forges 10% of messages
  - 1% gossip: $P(\text{cheater gets caught}) = 0.97$
Message

- action for the current turn $t$
- delegate for the next turn $t + 1$
- set of actions (i.e., gossip) from the previous turn $t - 1$
Collusion

- imperfect information games
  - infer the hidden information
  - outwit the opponents
- collusion = two or more players play together without informing the other participants
- how to detect collusion in online game?
  - players can communicate through other media
  - one player can have several avatars
Analysing collusion

- tracking
  - determine who the players are
  - but physical identity does not reflect who is actually playing the game

- styling
  - analyse how the players play the game
  - requires a sufficient amount of game data
  - collusion can be detected only afterwards

→ no pre-emptive nor real-time counter-measures
Collusion types

- active collusion
  - cheaters play more aggressively than they normally would
  - can be detected with styling

- passive collusion
  - cheaters play more cautiously than they normally would
  - practically undetectable
Offending other players

- acting against the ‘spirit’ of the game
  - problematic: is camping in a first-person shooter cheating or just a good tactic?
  - some rules are ‘gentlemen’s agreements’

- examples
  - killing and stealing from inexperienced and ill-equipped players
  - gangs and ghettoization of the game world
  - blocking exits, interfering fights, verbal abuse
Upholding justice

- players handle the policing themselves
  - theory: players take the law into their own hands (e.g., militia)
  - reality: gangs shall inherit the game world

- systems records misconducts and brands offenders as criminals
  - theory: bounties and penalties prevent crimes
  - reality: throw-away avatars commit the crimes

- players decide whether they can offend/be offended
  - theory: players know what kind of game world they want
  - reality: how to offend you? let me count the ways…