Very brief look at a handful of cheats in Dark Souls III ranging from mild annoyances to dangerous exploits. Not an exhaustive list, Blue Sentinel has hundreds of patches for almost every single networked interaction, but they generally fall into one of these categories.

Crashes

Out-of-bounds coordinates and collision related issues

Overview:
FLT_MAX/FLT_MIN, +/-inf, or (float)NaN coordinates from player entities cause serious performance issues due to issues with movement interpolation and collision. Turning off collision fixes the crash and performance issues around this but it's unreliable.

Fix:
Stop players going to these locations, checking player frame update packets for the new coordinates and checking for NaN, ensuring both the local and other players in the session can either not be teleported away, or at the very least don't teleport away on the local player screen.

Bullet-related frame drops

Overview:
Simply put, bullets which either have an enormous hitbox, expand over time, or cause a lot of visual effects put a great deal of strain on the game. This isn't just restricted to bullets either, it is possible to do this by producing other visual effects (e.g. receiving souls) to accomplish the same effect.

Fix:
Manually blacklisting bullets, alternatively checking local param files for large bullet hitboxes and not allowing them to be spawned in an excessive quantity. The game already has a cap on bullets at ~128.

Heap exhaustion

Overview:
Dark Souls III accepts size arguments for HeapAlloc through P2P traffic, that's pretty much all that needs saying. You can either deplete that particular heap or run it all out at once, to either cause problems immediately or a bit later on.

The game also doesn't check to see if memory has been allocated successfully in some areas.

Fix:
Sanity checks on heap sizes that are networked, or alternatively to just re-calculate what the size should be based on the amount of data segments received from the other player.

Out-of-bounds reads

Overview:
Too many to count, generally either related to packets declaring the amount of data segments they have themselves, with no bounds checking or verification, or mostly null pointers. The game does check for null pointers when trying to retrieve a WorldChrIns pointer from an entity handle / WhoID but in certain places it just doesn't.

Fix:
Bounds checking, verification of data segments, retrieving WorldChrIns pointers and checking them before allowing the game to retrieve them for itself - Avoiding code modifications.

Out-of-bounds writes

Overview:
As above - lots of these. Hard to use but have the most malicious potential

Fix:
Same as the out-of-bounds reads. The issues are usually lack of bounds checking so it's a simple exercise to make sure nothing is accessing memory that it shouldn't be.

Stack overflow

Overview:
Specific here to enemy action and enemy world initiation packets, there is no packing integrity performed on packed data to check if it has been formed correctly or not. Functions allocate 300 - 400 bytes of memory for the buffer in order to unpack the data in to, but from my testing you can go well above 900 bytes which overwrites the stack cookie and ultimately crashes the application.

Fix:
Enemy action packets are flawed in that they have no bounds checking on the argument which tells the game how many segments of enemy action data is in the packet, therefore you can send a 4-byte packet and declare 4 billion segments of data. This would be fairly benign but eventually leads to a lengthy freeze and eventual crash due to out-of-bounds read.

Therefore both the segment count and segment unpacked size need to be managed.

ACE / RCE

Overview:
Allows execution of attacker-written code on an otherwise uncompromised machine. Wholly unreliable, but has a very low complexity so is very easy to pull off and apply to a greater proportion of players.

Fix:
Fixing the underlying cause (not going into detail here because still not seen in the wild)

Exploits

Malicious / Instant-kill effects

Overview:
Instant kills: Annoying but tricky to patch.

Fix:
Player number spoofing patches, on-hit effect validation, damage re-calculation based on the attacking player's character.

Player number spoofing

Overview:
Dark Souls III sometimes uses a unique number in P2P packets to identify which player to perform an action on. It uses this number to get WorldChrIns structures to the player it wants. Two players who share the same player number will experience strange game behaviours, not limited to the game receiving data from a remote player, and applying it to the local player in error.

In addition to this, when the remote player is constructed it allocates a PlayerGameData slot for them by using their player number. In this way two players may share the same PlayerGameData structure.

Fix:
Discarding player numbers, as they are from older code in favour of SteamGameData pointers to find which player has sent what packet.

Bad warps

Overview:
Warping players to an invalid map with a normal (but not usually networked) MsgMapEvent ('LuaWarpEvent_01'). This is used normally in conjunction with other game events in order to warp you from one bonfire to another, however when used alone will take you to an invalid map where you'll continue to fall and die over and over.

Fix:
Ignore these events.

PvE damage

Overview:
PvE damage packets (packet 21) can be modified to become PvP damage packets. The difference is that you can directly modify the stamina and/or health of the other player, regardless of iframes or distance.

Fix:
Ignoring PvE damage packets that are structured to be PvP damage packets.

Humanity / Soul changes

Overview:
Allows direct modification of humanity (aka: ‘HeroPoint’). This is fairly benign because this attribute isn’t used in Dark Souls III.

Fix:
The packet is not sent under normal game circumstances. Ignoring it works.

Attribute and player game data changes

Overview:
Allows modification of player game data (aka: ‘PlayerParam’) such as attributes, name, covenant, play time, etc.

Fix:
Fixing the underlying cause (not going into detail here because still not seen in the wild)

Invalid items

Overview:
Item networking is strange and confusing in Dark Souls games. It’s host based unlike many other parts, and involves a relay of different types of item object packets:

- Type 0: ‘RequestWorldItemInfo’
- Type 1: ‘DeleteItemCategory ‘
- Type 2: ‘OnDelete’
- Type 3: ‘RemoteToHost’
- Type 4: ‘HostToRemote’
- Type 5: ‘PickupRequest’
- Type 6: ‘Give’

So in an example, if Player A was a host, and player B was a phantom. Player B joins the world and sends a type 0 packet to the host. The host then sends them a type 1 packet back which removes a certain category of items from displaying (for example ItemLot items, enemy drops, etc.) and also one type 4 packet for every existing item object there is in the world.

If player B drops an item, they will send a type 3 packet to the host, who will then send a type 4 packet to everyone who is connected, which registers the item on the ground.
If player B then picks that item up, they send a type 5 packet to the host, who sends a type 6 packet back with the item information inside it. If two players pick up the same item at the same time, only one of them will get it (whoever sends the packet type 5 first)

Now the issue is that you can just skip right to type 6 packets. It has no prerequisites and doesn't need the item to exist on the ground. This is how the 'Item Give' cheat works, where cheats can put items right into your inventory.

Fix:
Confirmation of packet type 5 sending before accepting a packet type 6. Working on an item whitelist basis is much easier than a blacklist.

Game state changes

Overview:
The game needs to network game event flags in order for synchronous multiplayer, however there is no differentiation between flags that should and shouldn't be set by other players. Some examples of these are:

- Flags that proceed your character to the next new game cycle
- Flags that either cause friendly NPCs to become hostile, or disappear
- Flags that prevent interactions with certain objects (e.g. ladders)
- Flags that kill bosses
- ..

The host is at greatest risk here - Phantoms have a unique flag set ('IsMyWorld') to false which means that any game events set aren't saved to their progression flags in their own world. Killing an invader's boss isn't going to do anything because once they return to their original world the flag will be restored.

That being said, flags that warp the player or affect other flags can be harmful to save files. Even though setting the NG+ flag on an invader's game is fairly harmless, it sets off a series of events that returns the player back to their own world before wiping basically every other game flag. This is why invaders just have their progress reset, and don't advance to a higher game cycle.

Fix:
Monitoring of incoming game progression flags, making automatic backups of saves via Blue Sentinel.