Cyber Space Odyssey - Player Guide

Kendra Graham, Bryce Heitmeyer, Conrad Rife, Scott Nykl

1 Story

A spy has recently discovered a terrorist plot to cripple the U.S. military by destroying their top academic institution: the Air Force Institute of Technology. Central to this diabolical scheme is a plan for an undercover agent working on one of five space stations to hijack the station and use it to breach AFIT security systems. Although unable to identify the rascal, the spy was able to confirm that further incriminating information might exist on the space stations themselves. The government has therefore offered a $1,000,000 reward to the first person to bring them the rogue operative, accompanied by sufficient evidence to convict him or her in a court of law. You happen to be a technically talented bounty hunter with a spaceship, so this is the perfect job for you. Your plan is to infiltrate the private network of each space station in order to collect the necessary evidence to identify and convict the miscreant. Be quick, though; you are not the only bounty hunter pursuing such a handsome reward...

2 Flying the Spaceship

If your spaceship doesn't move when you move the throttle, click on the window with your mouse to make sure it's in focus, and make sure the emergency brake is off (shown at the bottom right of the gameplay window).

If you're lost or confused, point your spaceship nose at the earth, then go backwards as fast as you can so you can zoom out and see where everything is. You can also self-destruct (**DELETE** key) if more convenient, but this comes with a penalty.

If the spaceship seems to not move properly, you can press the 'C' key on the keyboard and follow the on-screen text to calibrate both of the controller's sticks to the correct sensitivity.

There may be additional problems even after calibration, which most likely relate to thrusting. If you are facing the problem of slow forward or backward thrust, there are three possible solutions: (1) attempt to recalibrate again, (2) disconnect and reconnect the controller and try calibrating again, or (3) disconnect the controller, close the game down, rerun the game, reconnect the controller, then recalibrate.

Due to the problems mentioned above, some axes are more sensitive than others. This means that there is a possibility that stopping the ship's movement does not mean that the joysticks are in their center position, but could be closer to one side or the other. Keep this in mind as you play the game, so you will be able to stop the ship when necessary.

The right-hand vertical switch [2] is the emergency brake; toggle the switch downward to activate the brakes. The right-hand horizontal switch [3] is a windshield wiper, which clears the screen of GUI messages. The left-hand dial [4] and right-hand dial [1] are for the laser that can be turned on if turned to the right and off if to the left; use it to lower the health of other spaceships, ultimately disabling their controls.

It is also possible to control the spaceship with your keyboard:
- **W**, **A**, **S**, **D** control your thrust and yaw (W/S for thrust, A/D for yaw)
- **Q**, **E**, **R**, **F** control your pitch and roll (R/F for pitch, Q/E for roll)
- Holding **LEFT SHIFT** halves the speed of all movement (for increased precision)

Finally, both joystick and keyboard have additional useful keys available as well:
- **SPACE** shoots the laser
- **TAB** toggles the client's bubble visibility
- **DELETE** causes the ship to self-destruct
- **RIGHT SHIFT** toggles between first and third person mode
- **X** clears the GUI

Note: When the joystick is plugged in, the movement keypresses will be disabled. So, if you prefer to move the ship with the keyboard, unplug the joystick.

3 Obtaining a Clue

1. Fly into any space station's bubble and press **H**, which sends a hail message. Stay in the bubble as you perform the rest of the steps. The stations can be visited in any order you like.
2. The station/server will respond by sending you the string of a mathematical cypher. You must find the packet in Wireshark and solve it. Send a `NetMsgCypherResponse` containing the answer once you have solved the cypher.
3. The server will check your answer and send the next step to the station's team - Red station will send to Red team, Yellow station to Yellow team, and so on. You must intercept that packet to get access to a puzzle that ultimately unveils an encrypted password, and then use `EncryptionTools.exe` or other means to decrypt the password. Once you have done so, send a `NetMsgInformationRequest` to the server containing the decrypted password.
4. The server will send a possibly encrypted clue to the station's ship. Find the packet, use `EncryptionTools.exe` or other means to decrypt it, then save it for later so you don't lose it.

4 Making an Accusation

Once you have obtained all the clues, pieced them together, and then solved the mystery, you can apprehend the agent. Fly to the agent's space station and send a `NetMsgInitiateGuess`. The server will respond by sending you a `NetMsg AuthenticateGuess`, which contains a randomly generated authentication key that you must have in order to send a `Net MsgGuess`. Use Wireshark to find the `NetMsgAuthenticateGuess` and the authentication key, then send a `NetMsg Guess`.

Each authentication key only works once, so if you make a mistake in your `NetMsgGuess`, then you must start over with a `NetMsgInitiateGuess`and get another authentication key.

5 Wireshark Filters

Filters in Wireshark are boolean expressions that can be combined using parenthesis, the `AND` operator , and the `OR` operator. The `AND` operator is `&&` (2 ampersands), and the `OR` operator is `||` (2 pipes). The following are expressions that will be useful during the game:

- `udp` shows only UDP datagrams
- `ip.src == 192.168.225.105` shows only messages from IP address `192.168.225.105`
- `ip.dst == 192.168.225.105` shows only messages sent to that IP address
- `data.len != 128` filters out packets whose data is of length 128. You can exclude all packets that are continuously broadcast by excluding packets of length 128, 36, and 58.
- `data contains 19:c0:78:0e` will only show packets that contain the byte sequence `19:c0:78:0e`. This is helpful when trying to find a particular kind of NetMsg. Find the NetMsg's ID number and use programmer mode on Windows' calculator to convert the integer into hexadecimal, then type the bytes one at a time as shown. So if the NetMsg's ID is 432044046, then the hexadecimal representation is 19c0780e, the byte sequence is 19:c0:78:0e, and the filter to find that NetMsg is `data contains 19:c0:78:0e`.
- `data contains` also works with ASCII strings; to find a packet containing the string "CYPHER", you would use `data contains "CYPHER"`

To copy the ASCII text from WireShark packets, double-click on the desired packet to pop-up a new window and right click the ASCII text you would like to copy. Under "Show text based on packet", make sure that the "...as ASCII" is selected by checking if there is a blue box with a dot in the middle to the left of it. If it is not selected, select it (you will only have to do this step the first time). Now, right click the text you would like to copy again and under "Copy Bytes as Hex + ASCII dump", click on "...as Printable Text". Now, you have copied the desired information. Be wary when doing this, as it tends to copy a small amount of junk characters in front of the characters that are actually important, so make sure you delete unnecessary characters.

6 Sending Net Messages

Every type of `NetMsg` has a unique header ID, which allows the game's server to distinguish between the different types of messages. This is an `int`, and it is the first thing to be put into the `NetMessengerStreamBuffer` when sending a `NetMsg`. These header IDs can be found by scrolling all the way to the top of the client program's console window. Next, the length of the payload in bytes is inserted as an integer value, followed by the data particular to that kind of `NetMsg`.

Each packet sent also must include the correct IP address and port. The IP address ensures that the packet will be sent to the correct computer, and the port ensures that the packet will be picked up the correct application running on that machine, which in this case is Cyber Space Odyssey.

Note: A sample `NetMsg` on how to send the response to a cypher is found in the main function of UDPPacketSender. Use it to deduce how it and the other necessary `NetMsg` are to be sent.

7 IP address & Port

```
Server: 192.168.225.13, port 12683
   Red: 192.168.225.15, port 12685
Yellow: 192.168.225.16, port 12686
 Green: 192.168.225.17, port 12687
  Blue: 192.168.225.18, port 12688
  Cyan: 192.168.225.19, port 12689
```

8 Indices

```
[0] Red
[1] Yellow
[2] Green
[3] Blue
[4] Cyan
```

9 Hacks

Send a `NetMsgHack` packet, which contains
- your index
- your target's index
- the index of the team state value you want to change
- the boolean value you want to change it to

The indices of those booleans are as follows:
- Index [0] is `ailerons_jammed`, which causes the ship to roll at a constant rate
- Index [1] is `elevator_jammed`, which causes the ship to pitch at a constant rate
- Index [2] is `rudder_jammed`, which causes the ship to yaw at a constant rate
- Index [3] is `throttle_jammed`, which causes the ship to go backwards at a constant rate

10 NetMsgIDs

Several hours ago, AFIT intelligence was able to intercept a lone transmission containing an `std::map<std::string, uint64_t>`. Our analysts were able to determine the information's semantics: it is an association that maps a human-readable string to an integral enumeration. The integer value denotes the headerID expected in a specific type of NetMsg UDP packet.

```
NetMsgAuthenticateGuess .......1017311669
NetMsgClientJoin ..............3661831096
NetMsgClientPose ..............2265900392
NetMsgCypher ..................4182925792
NetMsgCypherResponse ............785688491
NetMsgGuess .....................2437867281
NetMsgGuiString .................2422918916
```

NetMsgHack 3645588255
NetMsgHailStation 2409192127
NetMsgInformationRequest 2744556811
NetMsgInitiateGuess 1464354183
NetMsgPassword 254716383
NetMsgTeamState 2060981026
NetMsgUpdatePose 3845570838

11 Summary of Net Message Payloads

The Net Messages in bold are the ones you will need to send manually.

`NetMsgCypherResponse`: client sends answer to a particular station's cypher to server
- unsigned int clientIndex;
- unsigned int stationIndex;
- int answer;

`NetMsgPassword`: server sends an encrypted station password to the team that corresponds to that station
- string information;

`NetMsgInformationRequest`: client requests a clue from a station. client to server
- unsigned int clientIndex;
- unsigned int stationIndex;
- string password;

`NetMsgClue`: server sends an encrypted station clue to the team that corresponds to that station
- string information;

`NetMsgInitiateGuess`: client begins two-step guess authentication with server.
- unsigned int clientIndex;

`NetMsgAuthenticateGuess`: middle of two-step authentication for guesses; server sends randomly generated key to client. Contains:
- int securityKey;

`NetMsgGuess`: final part of two-step guess authentication; client sends guess to server along with previously received key
- unsigned int clientIdx;
- int securityKey;
- string guessName;

`NetMsgHack`: client hacks another client. client to server
- unsigned int hackerIndex;
- unsigned int targetIndex;
- unsigned int whichHack;
- unsigned char desiredValue;

`NetMsgGuiString`: server prints to client GUI
- string message;

`NetMsgHailStation`: client initiates communications with a station. client to server
- unsigned int clientIndex;
- unsigned int stationIndex;

`NetMsgCypher`: server sends station cypher to client
- std::string cypher;

`NetMsgClientJoin`: client connects to server
- unsigned int clientIndex;
- unsigned short clientPort;

`NetMsgClientPose`: client sends ship position (x,y,z,roll,pitch,yaw) and shooting state to server
- float[6] xyzrpy;
- unsigned int clientIndex;
- int shipShooting;

`NetMsgTeamState`: server sends TeamState to all clients
- bool[20] booleans;
- bool[5] integers;
- bool[5] passwords;
- bool[5] clues;

`NetMsgUpdatePose`: server sends all ship positions (x,y,z,roll,pitch,yaw), all ship healths, and all visible lasers to all clients
- float[30] xyzrpy;
- int[5] allHealth;
- int[5] laserArr;

12 NetMsg Packet Structure

The image below shows the composition of a UDP packet used by Cyber Space Odyssey.

1. The first 4 bytes (shown circled in pink) are the NetMsgID – this is the type of message you are sending and is fully explained in Sec 11 (above).
2. The second 4 bytes (shown circled in yellow) is the total number of bytes in the payload. This is automatically updated on line 127 `is.updatePayloadLengthInHeader();` after all the data has been inserted.
3. The remaining bytes (shown circled in cyan) is the payload of the packet. The payload contains the integers 12, 13, and 14 (inserted on lines 120,121,and 122, respectively). This is immediately followed by the capital letters A-Z, defined on line 113 and inserted on line 123. Finally, the last two integers are inserted on lines 124 and 125. Finally, after all the desired data is inserted, the `is.updatePayloadLengthInHeader();` is called on line 127.
4. Line 131 sends this NetMsgStreamBuffer `is` out to the network as a UDP packet.