Tagger – An HvZ Source Replacement

Hello, if you are reading this, then it has either become your responsibility to look after Tagger, or you are interested in how it works and what it can do. I am Dan Woods, long serving member of committee (2015-2019), and I wrote all of this. Tagger uses the discord.py library to hook into discord to run games of HvZ. This document will include three different parts. The first pages will explain how the HvZ discord server works and what structure needs to be present for the bot to work. The next section will describe each command included in tagger, how to use them, and how not to use them. And the sections after those will discuss some under the hood stuff about how the bot works to inform future changes. If you ever need help or more information on any of this, please contact me on GitHub at dotdandotunderscore

# Section 1 – Discord Server Structure

The bot looks for certain things to know where to post messages and how to use commands. The following roles must exist for bot functionality (Case-sensitive).

* Admin
* Committee
* Moderator
* Human
* Zombie
* Corpse
* Non-Member
* All 8 class roles (collateral, reroll, heavy, ranged, explode, splash, fast, deflect) for e-hvz

The following channels must exist and must have the following-following permissions (note that server categories are not important to the bot):

Accessible = read and message, blocked = no read or message.

* #join
  + Accessible by @everyone (default).

Used for joining games.

* #game-announcements
  + Human, Zombie, Corpse readable. Moderator, Committee message-able.

Location for player starve messages.

* #bounty-wall
  + Human, Zombie, Corpse readable. Moderator, Committee message-able.

Location for bounty messages.

* #the-starve-wall
  + @everyone readable, no-one message-able.

Location for list of starve times.

* #human-chat
  + Human, Moderator accessible. All else, blocked.

Location for player tagged messages.

* #zombie-chat
  + Zombie, Corpse, Moderator, any user who is an OZ accessible. All else, blocked.

Location for player tagged messages and tagging commands.

Information on how to administrate discord servers can be found all over the internet, but unless there has been a big overhaul to the discord server, this structure should still exist. Please try not to manually set any roles as this can break bot functionality.

If for some reason a new server is ever set-up, you will need to get the ID of the new server (right click the server) and replace the text in guildID.txt with the new server ID.

If the #the-starve-wall channel is ever remade, the message ID of a message made by Tagger in the new channel must replace the text in starvewallmessageID.txt

# Section 2 – Tagger Command List

The commands included in tagger are currently split into three parts: Player, Moderator, and Admin commands. Tagger precedes all commands with a ‘.’ and so any other bots added to the server in the future must not use this prefix.

In all of the following [argument] bracket would be fully replaced with the relevant argument, i.e. .set\_colour [colour] would be used like .set\_colour blue

Arguments in grey are optional

Player Commands

**.join [student-number] or .join [firstname] [lastname]**

Will check the members.csv database for a matching student number and, if found, will make a new entry for that discord user in player\_database.csv, setting them as a Human. The bot will also DM that user with their braincode.

This command requires the user to be a member and for the members.csv to have been updated since they joined (more on that later).

Non-student members will not have a student number. Instead you can open up members.csv, find that members row and set their student number from (Unknown) to a unique identifier they can use to join games using .join . This will need to be redone whenever the members.csv is updated.

For the [firstname] [lastname] form of the command, it will add the user to player\_database.csv and set them as a Human **IF** they have the Non-Member role on discord. This is for people that cannot get membership for some reason, or for give-it-a-go games. The bot will also DM that user with their braincode.

**.check\_braincode**

**Must be Human.**

Will message the user their braincode.

**.bounty “[Firstname Lastname]” “[bounty]”**

**Must be Zombie.**

i.e. .bounty “Dan Woods” “1 Pint”

Will post a bounty on the bounty wall, mentioning the player who has been bountied.

**.how\_many\_zombies**

Will respond with how many Zombie players there are.

**.how\_many\_humans**

Will respond with how many Human players there are.

**.tag [braincode] @[user1] @[user2] “day/month-hour:minute”**

**Must be in #zombie-chat.**

Will turn a Human user into a Zombie user. Will feed that Zombie user to the current starve time. If up to two Zombie users are tagged, will feed them to the current starve-time. If time stamp added, will set the starve times to timestamp+starve time instead.

This is significantly the most complicated command from a backend perspective. I hope it doesn’t break.

Moderator Commands – All commands require the Moderator role

**.get\_braincode [username]**

i.e. .get\_braincode Infinity#2156

Will DM the mod with the braincode of the specified user.

**.revive [username]**

Will turn the specified Zombie user back into a Human. A new braincode will be generated and DM to the user.

**.revive\_all**

Will turn all Zombie users back into Human users. New braincodes will be generated and sent out.

**.feed @[user1] @[user2] ….. [hours to feed]**

Will feed the specified Zombie users to the current starvetime or for a specified number of hours.

**.feed\_all [hours to feed]**

Will feed all Zombie users to the current starve time or for a specified number of hours.

**.delete\_player [username]**

Will remove the specified user from the game, removing roles and deleting them from player\_database.csv

Admin Commands – All commands require the Admin role

**.add\_non\_member @[user]**

Sets the user as a Non-Member.

**.check [row] [column]**

Prints all of, or parts of, player\_database.csv

**.set\_starvewall**

Sets starve-wall message content to the contents of starvelist.csv. Used if the starvewall breaks or maybe if there is a mass feed.

**.deroll\_all**

Removes all game roles from all server members. Used for ending the game. Will take time.

**.reset\_database**

Resets player\_database.csv and starvelist.csv. Used for ending the game.

**.end\_game**

Runs .reset\_database, .set\_starvewall, and .deroll\_all. Sends ‘All reset.’ when done. This fully ends and resets a game. **THIS WILL TAKE A LONG TIME.**

**.k**

Shutsdown Tagger. ONLY use this if you know what you are doing and you know how to restart the bot.

eHvZ Commands – All commands govern eHvZ functionality

**.e\_tag @user**

User must be a Zombie, @user must be a Human. Using the classes and statistics based on the eHvZ game Left 2 Die, performs a fight between two users resulting in telling the zombie the braincode so that they can register the tag, or the zombie being stunned.

Users must have one of the class roles specified at the start of the doc.

.potshot @user

User must be a Human, @user must be a Zombie. Using the classes and statistics based on the eHvZ game Left 2 Die, does one roll between the players and deals one damage to the zombie if successful.

.refill

Will reset your health to the max health of your class.

.heal [amount]

Will heal you by a set amount, up to your max health.

.take\_damage [amount]

Will deal damage to you by the amount given.

.tag\_reset [how many times to post]

Will post now and every hour from now (for the amount of hours given, so 3 posts now, 1 hour from now, 1 hour from then, and 1 hour from then) in game-announcements, telling the players that they may make tag attempts again. Use this in eHvZ games with limits on number of tag attempts per hour.

.explode

User must have the explode roll. Will give all zombies +1 fight and -1 max health for 5 minutes.

.refill\_zombies

User must be moderator. Will refill all zombies to max health.

.refill\_humans

User must be moderator. Will refill all humans to max health.

# Section 3 – Things you can change if you know what you are doing

Due to the way the bot has been made, some useful changes cannot be made by commands alone. These include:

**Setting the starve timer**. In /cogs/player\_commands.py and /cogs/moderator\_commands.py, one of variables set on initialisation is self.feed\_ammount = timeD(days = 2). The argument in timeD() can be set to any number of days, or any other time increment by looking into the python library datetime (real function name is timedelta). This must be changed in both files to function properly.

**Changing class functionality**. Changing the actual abilities of the classes requires actual code knowledge, but the statistics attached to each class can be changed in /cogs/ehvz\_commands.py and are set during the initialisation as self.class\_fights, self.calss\_damages, and self.class\_healths. Note that the stats as they currently sit have been decided upon based on statistical analysis of e\_tag success probabilities that lead to appropriate odds of human and zombie success. As such the current stats are about as “balanced” as they can be without changing the e\_tag system. Any changes must also be reflected in the .explode function.

# Section 3.5 – Things you can change if you **really** know what you are doing

If you are familiar with python then feel free to add or change functionality within this bot. MAKE A BACKUP OF THE CODE AS I HAVE LEFT IT FOR YOU, IN CASE YOU PERMANENTLY BREAK SOMETHING.

The bot is programmed in python using the discord py library. The version of the discord py module on my server doesn’t update and so any changes to that won’t break the bot. If you start running the bot on an HvZ server, updating of the discord py library is something you will have to pay attention to as new functionality may break the bot.

Occasionally Discord itself changes API functionality, which can break the bot. If the bot stops working, check on discord py and Discord itself for any recent changes. Discord API and discord.py are the names of some excellent and very helpful discord servers.

# Section 4 - Under The Hood

**UPDATE WHEN NO LONGER HOSTED ON MY SERVER**

Tagger is an, admittedly overly, complicated beast. Its folder structure can be seen in the following pictures. Commands need to be kept in the “cogs” folder, clean versions of databases need to be kept in “backupdatabase”, and all else should stay in the root folder. To run tagger, all you need to do is run the file: nrun.sh. This will run the server in nohup mode, simply meaning that you will see no logs and it will run as a background process (so you can shut your ssh instance and it will still run). If you would like to see logs, run the file: run.sh.

If you would like to make code base edits, please read up on discord.py for api functionality and best practices.

There are a few “fun” editable parts of Tagger, notably “death\_messages.txt” and “words.txt”. These control death message text snippets and braincode generation. Please follow the formats these already use if making additions.

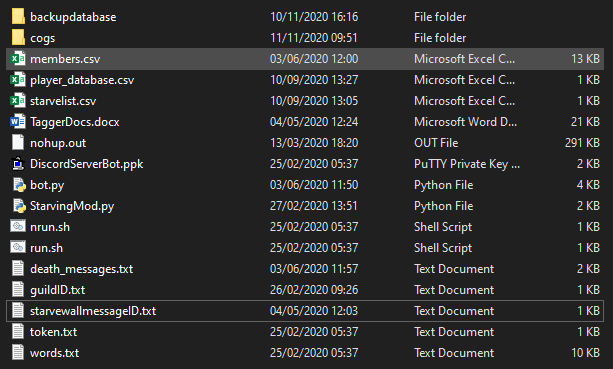
To update members.csv go to <https://www.su.rhul.ac.uk/> and use your committee member log in. Go to Admin -> Humans vs Zombies then Sales Reports. Change the “Date range” to this full academic year and then click “Purchasers Report”. Download with the floppy disc icon and click CSV (comma delimited). Rename this file to members.csv and replace the files in the root folder and in backupdatabase with it.

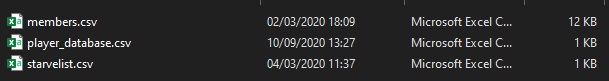
StarvingMod.py contains some of the infrastructure for the starving system, and bot.py deals with running the bot itself and tracking starving.

Most of the game is run through player\_database.csv so don’t fuck with it.

Good luck.

Root folder:



/backupdatabase/

/cogs/