Testing  
Here I will demonstrate that both the Web API and Discord+Twitch Bot function in their desired manner.  
Also that when presented with malicious or malformed data, no fault or breach of security occurs.

# Web API

To test functionality I will use the **cURL** command line utility. Which allows for you to make web requests easily from the command line.  
A typical request looks like this curl -X GET 'https://owlcoin.co.uk/webapi/Viewer' -H 'ID: 1'  
the –X indicates the request method (GET or POST)  
the ‘https://….’ Is the url to poll  
-H ‘ID: 1’ adds a header with the Key as ID and value of 1

For each key type of request, I will chose a suitable example and will demonstrate the following:

1. Will respond correctly when provided with fully correct data.
2. Will respond with suitable errors when provided with
   1. Non-existent data, like an ID that doesn’t match anything
   2. Malicious and invalid data
3. How data flows through the Web API by using Visual Studios built in debbuger

## Fetching Objects

curl -X GET 'https://owlcoin.co.uk/webapi/viewer' -H 'ID: 1'

{

"Data": {

"Balance": 300,

"WatchTime": 0,

"TwitchID": "179513695",

"DiscordID": "",

"Currency": {CURRENCY OBJECT}

},

"Code": 200,

"Message": "The requested task was performed successfully"

}

The command (far right), will fetch the **Viewer** object with **ID** of 1  
As you can see, when provided with a valid request, we receive back the data that corresponds to the entry in the database. In this case we receive the Balance, watchtime, etc. Along with the **Currency** object; which contains details about the **Currency** this **Viewer** is part of.

{  
 "Data": null,  
 "Code": 400,  
 "Message": "Bad Request, ID does not match an existing object"  
}

When an **ID** is provided that doesn’t correspond to an object we receive a response (close right) which contains an error indicating that the error is based on the **Requestors** data. Ie: Code 400 along with a message describing the issue encountered.

When the **ID** contains a non-numeric character, we receive the following (close left). Indicating that the ID value is abnormal.

{

"Data": null,

"Code": 400,

"Message": "Bad Request, No operable Headers provided"

}

{  
 "Data": null,  
 "Code": 400,  
 "Message": "Bad Request, Malformed ID"  
}

When the **ID** header is missing completely absent. We are informed (close right) that we don’t have enough **Headers** to perform any action. This will occur anywhere when too few **Headers** are provided.

{

"Data": [

{

"Balance": 300,

"WatchTime": 0,

"TwitchID": "179513695",

"DiscordID": "",

"Currency": {

CURRENCY OBJECT

},

"LiveNotifcations": false,

"DontReward": false,

"ID": 1

},

{

"Balance": 900,

"WatchTime": 0,

"TwitchID": "179513695",

"DiscordID": "",

"Currency": {

Different CURRENCY OBJECT

},

"LiveNotifcations": false,

"DontReward": false,

"ID": 1

}

],

"Code": 200,

"Message": "The requested task was performed successfully"

}

The **ID** header can be swapped out for **DiscordID** and/or **TwitchID**, hence will return all **Viewer**’s with the corresponding **ID**’s. Ie: curl -X GET 'https://owlcoin.co.uk/webapi/viewer' -H 'TwitchID: 179513695'

A further **CurrencyID** header can be included, to restrict the **Viewer**’s to be from the specific **Currency**.

If only the **CurrencyID** is provided, we will be returned a list of all **Viewer**’s that are part of said currency.  
With only the **CurrencyID** a **Order** header can be provided, which can have the value of “WatchTime” or “Balance”.  
This value indicates if we should sort the **Viewer**s in descending order based on their **Watchtime** or **Balance**.  
If this header is not provided, we will apply no sorting to the data.