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.

**Requestor –** A user that has performed a web request to the web api.  
Code – Indicates the highlighted text represents a small snippet of code  
**AccessToken** – Used to authenticate a **Login**  
**AuthToken** – Used to authenticate a **Bot**

# Web API

To test functionality I will use the **cURL** command line utility. Which allows you to make web requests easily from a command line.  
A typical request looks like this curl -X GET 'https://owlcoin.co.uk/webapi/Viewer' -H 'ID: 1' –d ‘{}’  
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  
-d ‘{}’ contains any JSON data that is to be sent with the request

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. Lack of authorisation
3. How data flows through the Web API by using Visual Studios built in debbuger

A good number of errors will occur across different requests, hence I will only talk about the error once but will demonstrate it in the video.

## Fetching Objects

{

"Data": {

"Balance": 300,

"WatchTime": 0,

"TwitchID": "179513695",

"DiscordID": "",

"Currency": {CURRENCY OBJECT}

},

"Code": 200,

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

}

The command curl -X GET 'https://owlcoin.co.uk/webapi/viewer' -H 'ID: 1', will fetch the **Viewer** object with **ID** of 1  
As you can see (far right), 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.

In some circumstances a string can be used to identify an object. For example, a Login can be identified by using the Username or Email. Hence a header of Username or Email can be provided along with the string value.  
Which will then cause the Login(s) to be returned.

## Creating Objects

{

"Data": null,

"Code": 200,

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

}

The command curl -X POST 'https://owlcoin.co.uk/webapi/viewer' -k -H 'AuthToken: AUTHTOKEN' -H 'BotID: 84' -H 'CurrencyID: 231' -H 'DiscordID: 1234' -d '' will create a **Viewer** for the specified **Currency**. If the given **DiscordID** and/or **TwitchID** isn’t present in the **Currency** then a successful response will be returned (far right).

{

"Data": null,

"Code": 400,

"Message": "Bad Request, a Viewer already exists in this currency with that Discord and/or Twitch ID"

}

However, if the **ID** is present we get the following error response (close right).

{

"Data": null,

"Code": 400,

"Message": "Bad Request, AuthToken is invalid for that Bot"

}

A further error message will be shown if the provided **AuthToken**(close right) or **BotID**(close below)is invalid.

{

"Data": null,

"Code": 400,

"Message": "Bad Request, BotID does not correspond to an object"

}

For creating other objects, less information will be required for to fill the default values.  
But almost all objects will require a reference to the parent; in this case the **Currency**; however this can sometimes be inferred from other information, such as the **Currency** the **Bot** is associated with.

## Modifying Objects

{

"Data": null,

"Code": 200,

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

}

The command curl -X POST 'https://owlcoin.co.uk/webapi/login' -H 'Email: NEWEMAIL -H 'ID: 32' -H 'AccessToken: ACCESSTOKEN' -d '' will modify the **Email** associated with the given **ID**. Headers similar to **Email** can be provided: **UserName**, **Password**. Which will change the associated **UserName**, **Password** respectively.

{

"Data": null,

"Code": 400,

"Message": "Bad Request, Email is not valid"

}

The provided **Email**, **UserName** and **Password** must all conform to their respective requirements. If they do not, we will receive the following error (far right); the error will differ based on the issue.

If an invalid

The following commands are examples of special cases for **Viewer** modification, hence appear in no other objects.

{

"Data": null,

"Code": 200,

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

}

The following command curl -X POST 'https://owlcoin.co.uk/webapi/viewer' -H 'BotID: 85' -H 'AuthToken: AUTHTOKEN' -H 'ID: 1716' -H 'Operator: +' -H 'Value: 1000' -d '' will increase or decrease the balance of the given **Viewer** by the given **Value**. If successful we receive the response (right).  
The **Operator** can be + or – and indicates if we are to increase or decrease the balance.

If the **Operator** is neither of the above. We will receive the response (immediately below).

{

"Data": null,

"Code": 400,

"Message": "Bad Request, Operator must be + or -"

}

If a negative **Value** is provided, we do not throw an error. Instead we just carry on and its negativity will take the usual effect. Ie + **Operator** and – **Value** will reduce balance.  
Hence it is in the hands of the **Requestor** to ensure correct symbols are used.

{

"Data": null,

"Code": 200,

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

}

The following command curl -X POST 'https://owlcoin.co.uk/webapi/viewer' -H 'BotID: 85'   
-H 'AuthToken: AUTHTOKEN' -H 'BalanceIncrement: 10' -H 'CurrencyID: 231' -d '{"DiscordIDs":["1234"]}'  
will increment the balance and/or watchtime of the **Viewer**s, where their **Discord** or **Twitch** **ID** appears in the list, by the given value. The **Viewer** must also be part of the **Currency** with the given **CurrencyID**.

In this case, we increase the balance of the **Viewer** with **DiscordID** 1234 by 10.

When no **Discord** or **Twitch** **ID**s are provided, we receive the same response, been as no error has occurred. Because we have incremented all corresponding **Viewers**.  
And once again, if the **Increment** is negative, it will reduce the value of the **Viewers**.

{

"Data": null,

"Code": 500,

"Message": "Internal Server Error"

}

If an invalid **JSON** (-d ‘{}’) is provided, we will receive an error (right), indicating that something had gone wrong, but the server does not know what.

## Authentication

{

"Data": {

"UserName": "Jaminima",

"HashedPassword": null,

"AccessToken": "ACCESSTOKEN",

"Email": "oscar.william.davies@outlook.com",

"LastLoginDateTime": "2019-04-02T10:16:32.2438323+01:00",

"ID": 32

},

"Code": 200,

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

}

### Performing Login

In order to obtain an **AccessToken**, we perform the command curl -X POST 'https://owlcoin.co.uk/webapi/login' -H 'UserName: Jaminima' -H 'Password: PASSWORD' -d ''  
if the **UserName** and **Password** is valid, we will receive the response (right).  
Which primarily contains our **AccessToken**, but also contains information about the account, which is usually kept private.

{

"Data": null,

"Code": 400,

"Message": "Bad Request, Password does not match"

}

**Email** can be used in place of the **Username.**

If the **Password** is invalid, we receive an error (close right).

{

"Data": null,

"Code": 400,

"Message": "Bad Request, UserName does not correspond to an existing user"

}

If the **UserName** or **Email** doesn’t match an **Login**, we receive the following error (close right).

Once we have acquire the **AccessToken**, we can use it in place of the **Password**.

### Performing OAuth

{

"Data": {

"Currency": CURRENCYOBJECT,

"AccessToken": "ACCESSTOKEN",

"RefreshToken": "REFRESHTOKEN",

"BotName": "No Name Given",

"TokenRefreshDateTime": "2019-04-02T10:31:47",

"OwnerLogin": null,

"IsSuperBot": false,

"ID": 86

},

"Code": 200,

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

}

Our imitation of OAuth, is used to authenticate bots. This involves issuing a **RefreshToken** and **AccessToken**, the first is used to obtain the latter and the latter is used to prove they are who they say they are.

When you first create the **Bot** curl -X POST 'https://owlcoin.co.uk/webapi/bot' -H 'LoginID: 32' -H 'AccessToken: ACCESSTOKEN' -H '' -d '' you will receive a response (right) containing the initial **RefreshToken** and **AccessToken**.  
The **AccessToken** will be valid for 10 minutes post acquisition.

In order to acquire a new **AccessToken** we must perform the following curl -X POST 'https://owlcoin.co.uk/webapi/bot' -H 'RefreshToken: REFRESHTOKEN' -H 'BotID: 86' -d '', using the **RefreshToken** from the previous refresh. We will receive a response (right) containing our new **AccessToken** and **RefreshToken**.