AlienVault Python/Django Coding Test

Goals:

- Part I: Consume JSON from an HTTP endpoint and transform it into an API request in Django Rest Framework (http://www.django-rest-framework.org/). Some of the values will be straight 1-1 mappings while others will require aggregation.
- Part II: Construct a simple Django model to track users accessing the page and also allow this to be read from the API.

Notes about the base files (ask us if you haven't received them):

- **Urls.py:** Django checks this first for routes. Create routes for /api/threat/ip/x.x.x.x and /api/traffic.
- **Views.py:** urls.py calls a view from views.py. In this case, the views are subclassed from DRF APIView. For this exercise, the view is a simple call to generate an IPDetails object, located in threat.py
- **Threat.py**: IPDetails data object does the work. It takes a raw argument, and makes a call to the reputation datastore, and then stores the results in the data object. IPDetails does double duty here as both the plain data object that can be serialized and the data interface.
- **Serializers.py**: The DetailsSerializer maps IPDetails to JSON output. You'll find many references to other serializers beyond serializers.Field() in the DRF framework. Ignore that for now since we are reading data back only and not writing. You will need to use serializers.Field(many=true) for activity output, however.

The JSON source is here:

Known Malicious:
http://reputation.alienvault.com/panel/ip_json.php?ip=69.43.161.174

Known Safe: (Will return blank result, doesn't error-check IP value so do this beforehand)
http://reputation.alienvault.com/panel/ip_json.php?ip=8.8.8.8

Part I: Simple JSON Transform to API Call

1) Make a simple Web API using Django Rest Framework (http://www.django-rest-framework.org/) in Python that can take any IP as input. URL format: /api/threat/ip/x.x.x.x as the format. Provide reasonable error-checking for IP input (empty, invalid, etc.)
2) Map the information from reputation.alienvault.com to this output API

Reputation API	**Your New API**	**No reputation result?**
Address	Address	Input

_id.$id	Id	“”
Reputation_val	Reputation_val	0
The value of the earliest first_date.sec in activites[]	first_activity	null
The value of the latest last_date.sec in activities[]	Last_activity	null
Activities	(* see activity mapping below, empty set if no reputation result)	[]
Enumeration of every unique value of activity.name in activities[]	Activity_types	[]
--	Is_valid: true if valid input, false otherwise	true if valid input, false otherwise

* Activities should be mapped as follows:

Reputation.Activities	**New api activity**
Name	Activity_type
First_date.sec	First_date
Last_date.sec	Last_date

Ignore all other JSON values for the purpose of this exercise.

3) See right column in first table on how to handle an error where no results are returned. I’ve included an is_valid field to set true or false on bad input. Feel free to add any API fields you think are relevant (error codes, etc.)

4) On a user access to the API, determine if it is their first access by looking to see if they have a cookie set for a unique user ID (call it AlienvaultID). If they do not have a cookie set, set it to a random string to 12 characters and set it to expire in one year. (Yes, this isn’t totally scientific but it simulates the behavior of many marketing automation packages.)

Part II: Track Users and Return Traffic Info at API Endpoint

1) Create a Django model to track simple statistics about who accesses the API via the web.
    a) IP Address
    b) Timestamp (epoch)
    c) API endpoint accessed (“/api/threat/ip/1.2.3.4”)
    d) The value of AlienvaultID set in #4 in the previous section
2) On each visit to the API endpoint above, record statistics about the user visiting.
3) Create another API endpoint using DRF at /api/traffic/ that does a simple dump of all the traffic stored in the model above sorted by AlienvaultID.

Structure for JSON:

```
[
        {
                “alienvaultid”: 123456789012,
                “visits”: [
                        { “address”: “1.2.1.34”, “timestamp”: 3828188382, “endpoint”:
                “/api/details/1.2.3.4”},
                        { “address”: “1.2.1.34”, “timestamp”: 3828188389, “endpoint”:
                “/api/details/2.3.4.5”}
                ]
        },
        {
                “alienvaultid”: 123456789013,
                “visits”: [
                        { “address”: “5.31.3.6”, “timestamp”: 3828188390, “endpoint”:
                “/api/details/1.2.3.4”},
                        { “address”: “28.11.4.5”, “timestamp”: 3828188396, “endpoint”:
                “/api/details/2.3.4.5”}
                ]
        }
]
```

Other Stuff:

- Python 2.7 preferred. Environment setup (Python + Django + DRF) is up to you.
- Disable all authentication for your web API. Anon access is fine.
- There is no need to write an API consumer for this exercise. Meter the traffic in part 2 using simple calls to the API endpoint in Part I.

DELIVERABLE:

- Two working demonstrations on URLs of your choice. Deploy somewhere that can be publicly accessed.

- GitHub/Dropbox/zip/etc. of your code.

Let us know when you parse through this what your expected delivery date is. Thanks and looking forward to your work!