**In-Band Data Extraction**

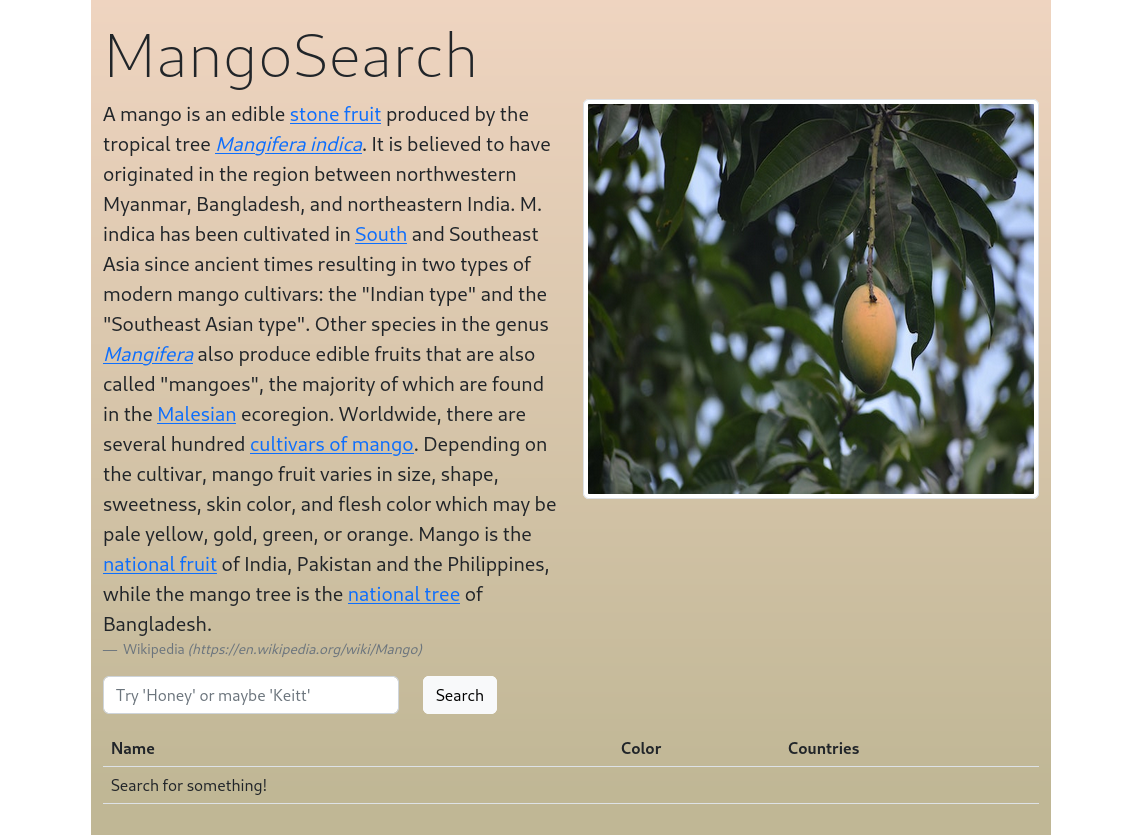
**Theory**

In traditional SQL databases, in-band data extraction vulnerabilities can often lead to the entire database being exfiltrated. In MongoDB, however, since it is a non-relational database and queries are performed on specific collections, attacks are (usually) limited to the collection the injection applies to.

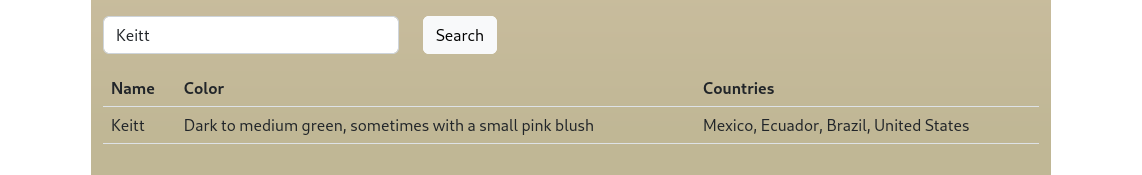
**MangoSearch**

In this section, we will take a look at MangoSearch. This application is vulnerable to in-band data extraction.

The website itself is very basic: A quote from Wikipedia. An image of a Mango. A search area where you can find facts about the various types of mangoes.



We can try searching one of the recommended types to see what request is sent and what sort of information is returned.



We can see that the search form sends a GET request where the search query is passed in the URL as ?q=<search term>. Similarly to the previous section, this is URL-encoded data, so keep in mind that any NoSQL queries we want to use will have to be formatted like param[$op]=val.

On the server side, the request being made will likely query the database to find documents that have a name matching $\_GET['q'], like this:

Code: javascript

db.types.find({

name: $\_GET['q']

});

We want to list out information for all types in the collection, and assuming our assumption of how the back-end handles our input is correct, we can use a RegEx query that will match everything like this:

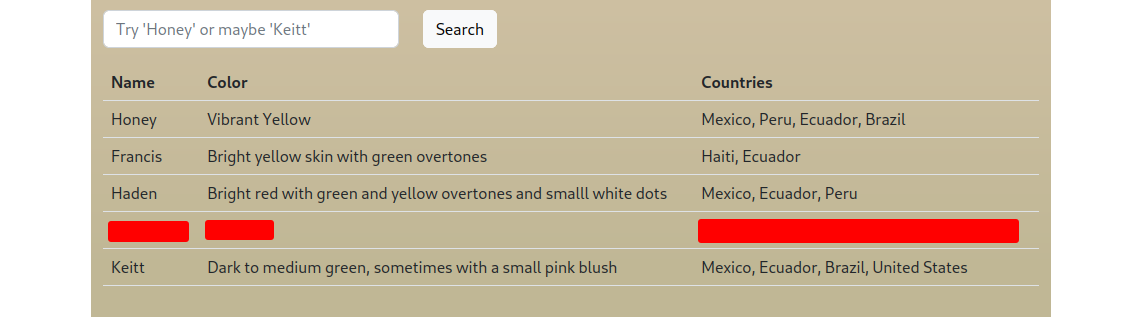
Code: javascript

db.types.find({

name: {$regex: /.\*/}

});

Upon sending the new request, we should see that all mango types and their corresponding facts are listed.



**Alternative Queries**

* name: {$ne: 'doesntExist'}: Assuming doesntExist doesn't match any documents' names, this will match all documents.
* name: {$gt: ''}: This matches all documents whose name is 'bigger' than an empty string.
* name: {$gte: ''}: This matches all documents whose name is 'bigger or equal to' an empty string.
* name: {$lt: '~'}: This compares the first character of name to a Tilda character and matches if it is 'less'. This will not always work, but it works in this case because Tilda is the [largest printable ASCII value](https://www.asciitable.com/), and we know that all names in the collection are composed of ASCII characters.
* name: {$lte: '~'}: Same logic as above, except it additionally matches documents whose names start with ~.