Update Every Review: NoSQL Injection

NoSQL databases are not immune to injection just because they don't use Structured Query Language. They still run queries, and those queries are still text that can be interpreted as code, which is the idea at the core of injection attacks of all kinds.

Ideas

You can post reviews on products.
You can edit your existing reviews.
Wouldn't it be ... unfortunate ... if you could update every review at once?

Look at those requests for places you might inject some NoSQL keywords or special characters.

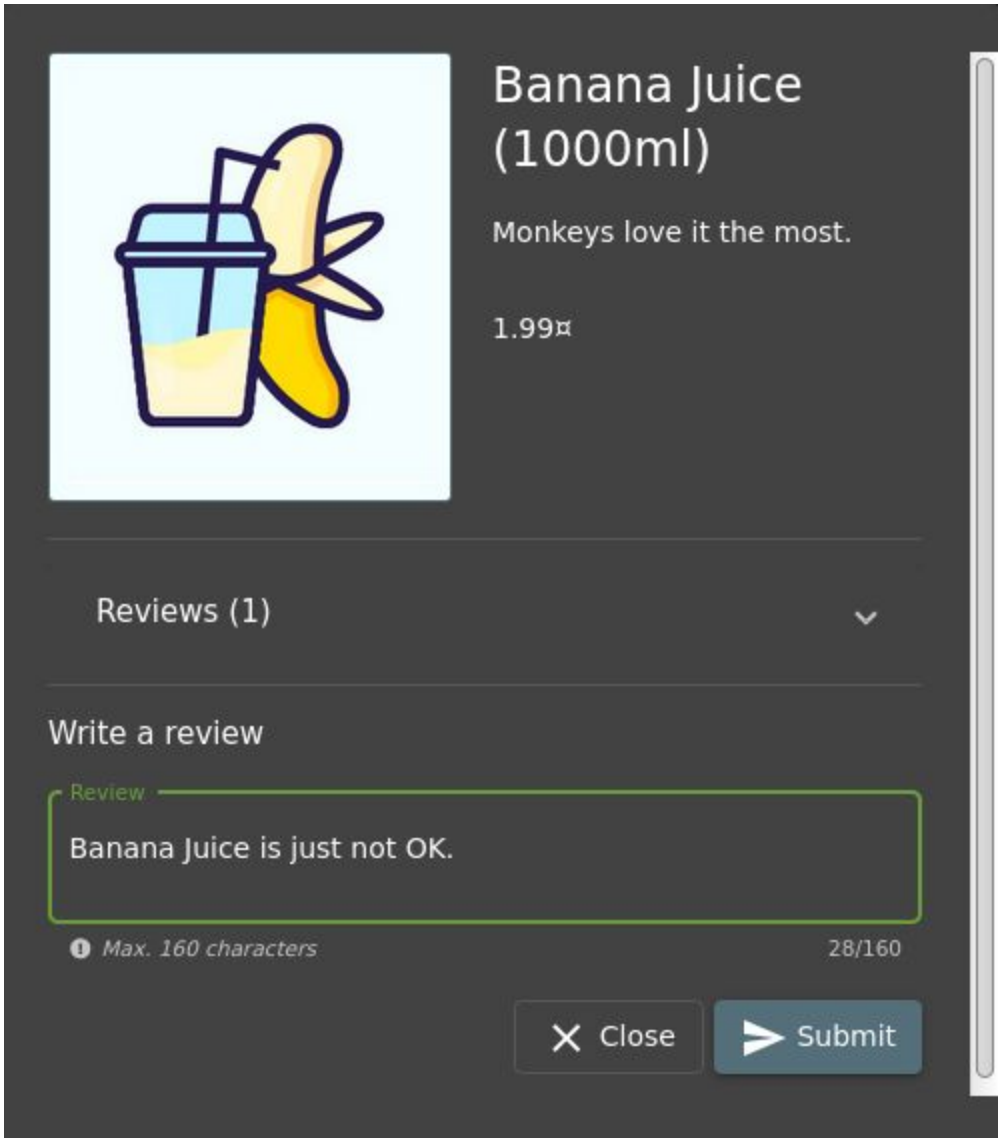
If you want to find everything in an inventory with a status of 'D':
```
db.inventory.find({ status: 'D' } );
```

...if you want to find everything with a status that is not 'D':
```
db.inventory.find({ status: {"$ne":  'D'} } );
```

Walk-Thru

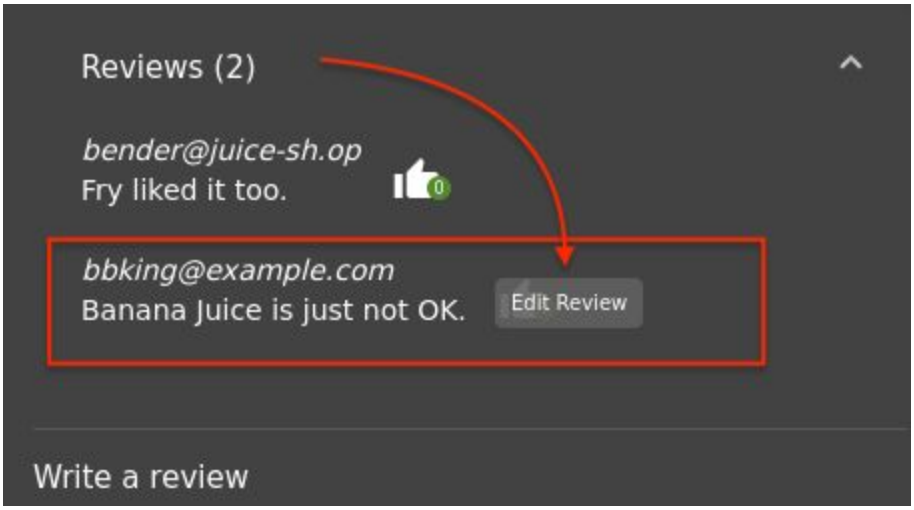
Make sure you have Firefox set to use your Burp Suite as a proxy, and that the Proxy > Intercept pane says "Intercept is off"

1. Log in as your user
2. Choose a product and submit a review saying whatever you like.



Submit Any Review

3. Click the "Reviews" drop-down link to see reviews on the product, then mouse over your review and click to edit it.



You Can Edit Your Reviews!

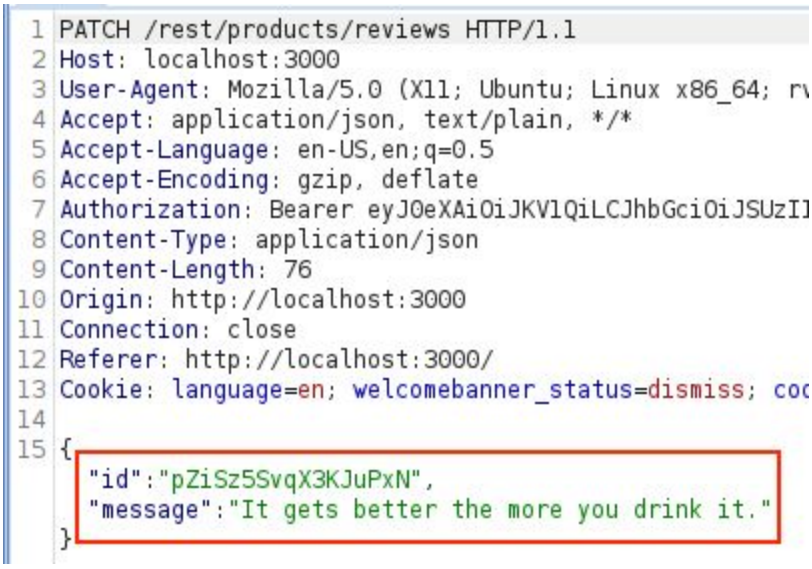
4. Click "Submit" to save your edits

That's Better

5. Look at Burp's Proxy History for the PUT request and the PATCH request to `/rest/products/6/reviews` that contain your reviews. (You may have a different product number of course)

6. Notice that the PUT request, which created the review, includes a "message" and an "author" where the PATCH request, which modified it, includes an "id" and a "message"

PUT Request: Message Text and Author's Email



PATCH Request: Message ID and Message Text

7. Send the "PATCH" request to Burp Repeater and click "Send" there to make sure it works.

8. Edit the "id" value to include a single quote or curly braces and send that.

9. Notice you get no response, or you get an error in response. This is your clue that you broke some back-end processing

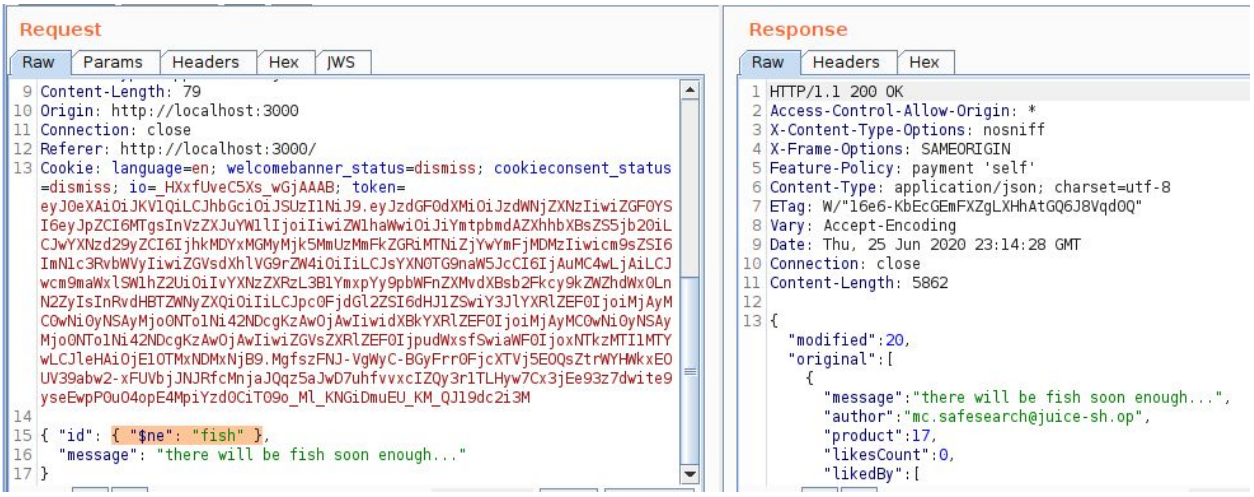
10. Replace the string value of the "id" parameter with a JSON object that will evaluate to true in its place. Imagine what the NoSQL query might look like if it's making an update. Check the MongoDB reference manual or the slides just before this lab in the course for possible operators.
https://docs.mongodb.com/manual/reference/operator/query/
The code that handles this challenge in Juice Shop is included at the bottom of this walkthrough if you want to review it for ideas. In a real pentest, you may be allowed to see the server-side source code. Never hurts to ask!

Think about what the id parameter would need to contain in order to update every review in the database.

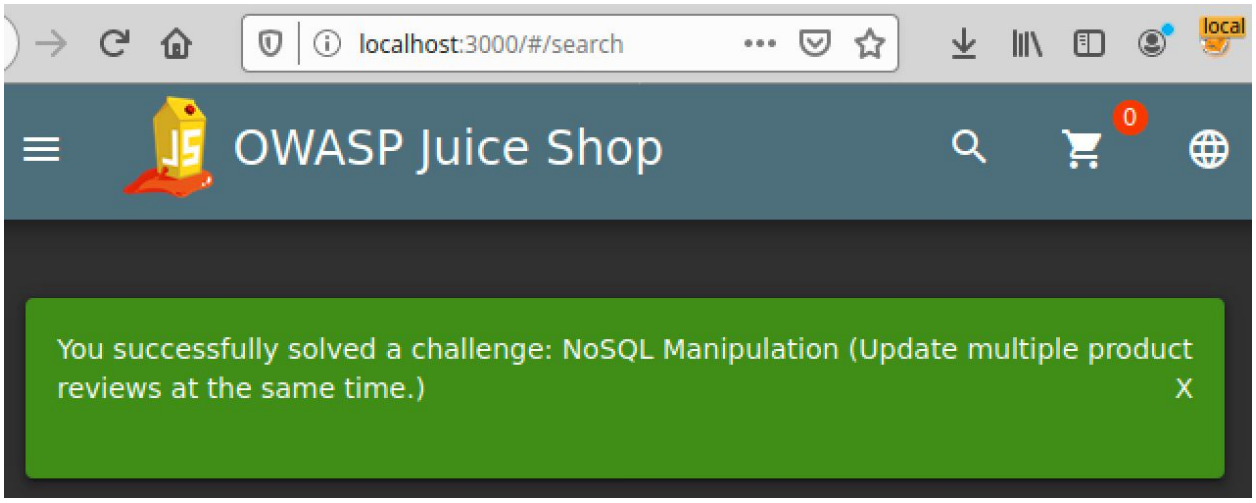
`{ "$ne" : "fish" }` might do it, because it's a safe bet that no reviews have "fish" as their id value.

11. Submit the following as the body of the PATCH request:

```
{"id": { "$ne": "fish" },"message":"now there are fish."}
```



12. Return to the browser view to receive your reward



Juice Shop Source Code Reference

This is the code that's running behind this challenge:
juice-shop/routes/updateProductReviews.js

```
const utils = require('../lib/utils')
const challenges = require('../data/datacache').challenges
const db = require('../data/mongodb')
const insecurity = require('../lib/insecurity')

module.exports = function productReviews () {
  return (req, res, next) => {
    const user = insecurity.authenticatedUsers.from(req)
    db.reviews.update(
      { _id: req.body.id },
      { $set: { message: req.body.message } },
      { multi: true }
    ).then(
      result => {
        utils.solveIf(challenges.noSqlReviewsChallenge, () => { return
result.modified > 1 })
        utils.solveIf(challenges.forgedReviewChallenge, () => { return user
&& user.data && result.original[0].author !== user.data.email &&
result.modified === 1 })
        res.json(result)
      }, err => {
        res.status(500).json(err)
      })
```

```
  }
}
```

MongoDB Query Selectors Reference

https://docs.mongodb.com/manual/reference/operator/query/#query-selectors

Name	Description
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.