baby todo or not todo

Platform	HTB
Operating System	Web-CTF
Tags	flask python

General-Information

▼ Table of Contents

▼ Notes

▼ Challenge Description

- I’m so done with these bloody HR solutions coming from those bloody HR specialists, I don’t need anyone monitoring my thoughts, or do I…?

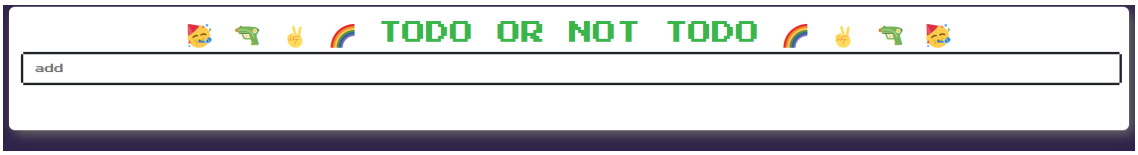
Summary

- Insufficient authentication on all API calls allows for any user to view all the previous calls or flag to the server.

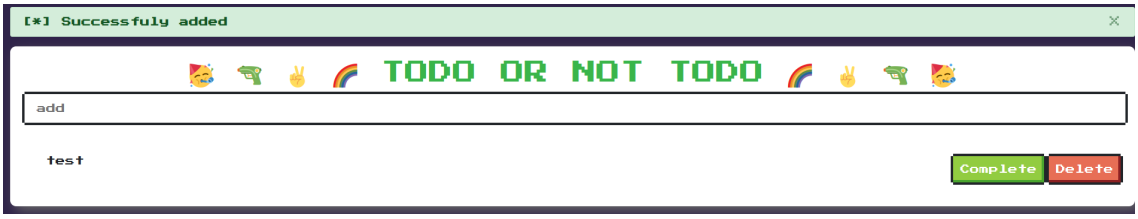
Website

▼ Looking at the website, there is just one spot for user input which doesn't sanitize the input that you write into it. Which at first would've lead me down a rabbit hole, but I checked the files for the challenge and was able to find the vulnerability in there.

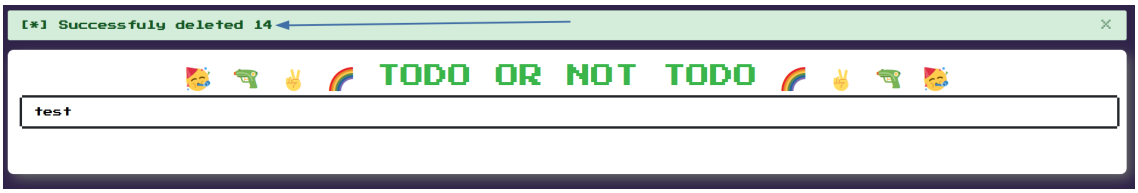
▼ Website



▼ Success image



▼ Deletion image



▼ Also when looking at the source code, a message about viewing "all" and checking the verify_integrity stand out as well.

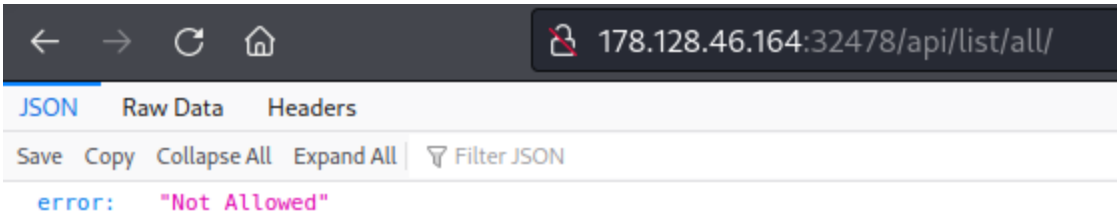
```
<script src="/static/js/main.js"></script>
<script>
// don't use getstatus('all') until we get the verify_integrity() patched
const update = () => getTasks('userb7Fc9a5F')
update()
setInterval(update, 3000)
</script>
</body>
</html>
```

Flask files

▼ When I was reviewing the `routes.py` file, the `/list/all` API route stood out because It would return all the JSON text for this “to do” list application.

```
# TODO: There are not view arguments involved, I hope this doesn't break
# the authentication control on the verify_integrity() decorator
@api.route('/list/all/')
def list_all():
    return jsonify(todo.get_all())
```

▼ Along with that strange route, the comment about checking the `verify_integrity()` function made me give it a look over as well. However, I went to try and view `/api/list/all` first, but was given a 403 error

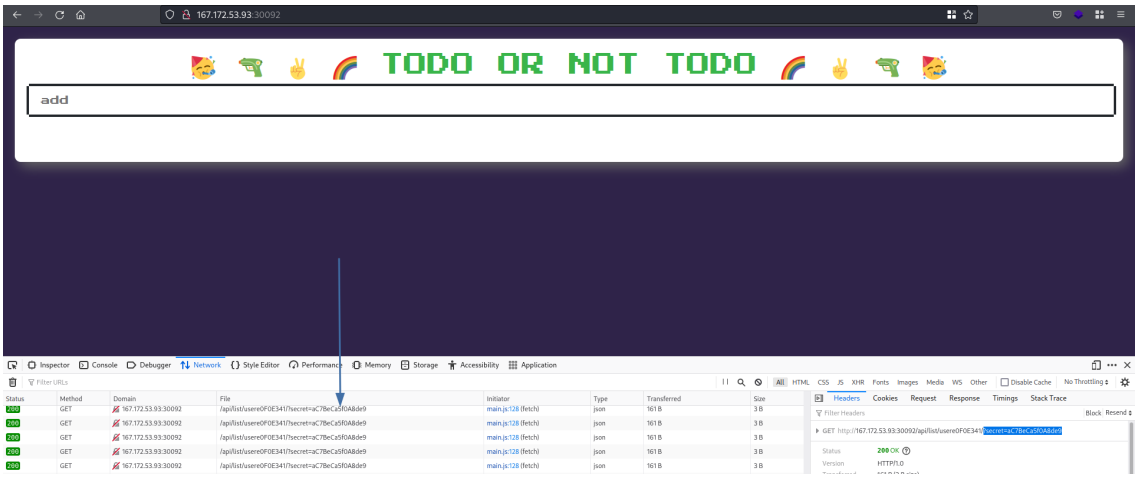


▼ Looking over the `verify_integrity()` function to verify my understanding of why the error code was caused I see that `check_integrity` is going to see if the request contains any arguments and if it does contain JSON then it process it a certain way.

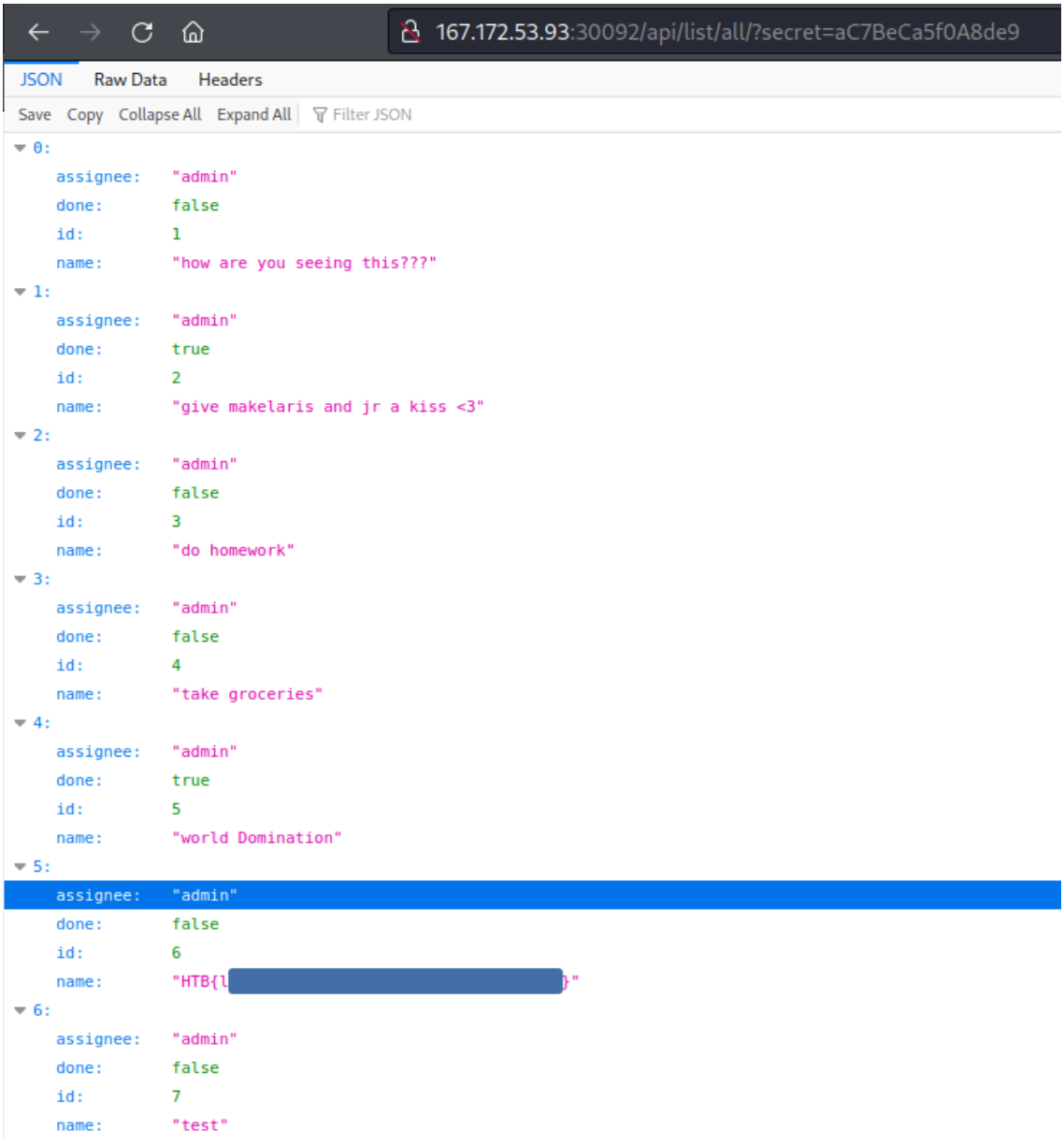
```python
def verify_integrity(func):
    def check_secret(secret, name):
        if secret != todo.get_secret_from(name):
            return abort(403)

    @functools.wraps(func)
    def check_integrity(*args, **kwargs):
        g.secret = request.args.get('secret', '') or request.form.get('secret', '')

        if request.view_args:
            list_access = request.view_args.get('assignee', '')

            if list_access and list_access != g.user:
                return abort(403)

            todo_id = request.view_args.get('todo_id', '')
            if todo_id:
                g.selected = todo.get_by_id(todo_id)

                if g.selected:
                    if dict(g.selected).get('assignee') == g.user:
                        check_secret(g.secret, g.user)
                        return func(*args, **kwargs)

                    return abort(403)

                return abort(404)

        if request.is_json:
            g.task = request.get_json()
            g.name = g.task.get('name', '')
```

▼ However, if the request has no arguments, then it will look for a `secret` or the user's secret ID that gets assigned at the beginning. Which we're shown how to append to request with `?/secret`. So, I send a request to `/api/list/all/?secret=aC7BeCa5f0A8de9` and get back the flag!

▼ Taking the `?/secret` upon navigating to the website



▼ Displaying the flag



Information Learned

- Breaking down the files and understanding what each function did until I got to the weird part really helped me identify the issue that was present.
- In hindsight this challenge left a super easy clue in the source code, however at the time I couldn't understand its full impact yet.