

Unsafe Code Lab: How Modern Web Frameworks Fail (and How to Fix Them)

Irina Iarlykanova





A Week in the Life of an AppSec Engineer

Flask: user_id = request.values.get('id')

ExpressJS: const userId = req.param('id')

FastAPI: def get_user(id: int): ...

```
Flask: user_id = request.values.get('id')
```

```
ExpressJS: const userId = req.param('id')
```

```
FastAPI: def get_user(id: int): ...
```

```
Flask: user_id = request.values.get('id')
```

```
ExpressJS: const userId = req.param('id')
```

```
FastAPI: def get_user(id: int): ...
```

Same logic, same bug?

Flask: `user_id = request.values.get('id')`

ExpressJS: `const userId = req.param('id')`

FastAPI: `def get_user(id: int): ...`

Does ?id=1&id=2 behave the same?
What about sending the id in the
body **and** the query string?



Absolutely Not!

```
Flask: user_id = request.values.get('id')
```

Type confusion

```
ExpressJS: const userId = req.param('id')
```

```
FastAPI: def get_user(id: int):...
```

Safe by default



Irina Iarlykanova



B.Sc. Student at
Maastricht University
SOC analyst at DIVD

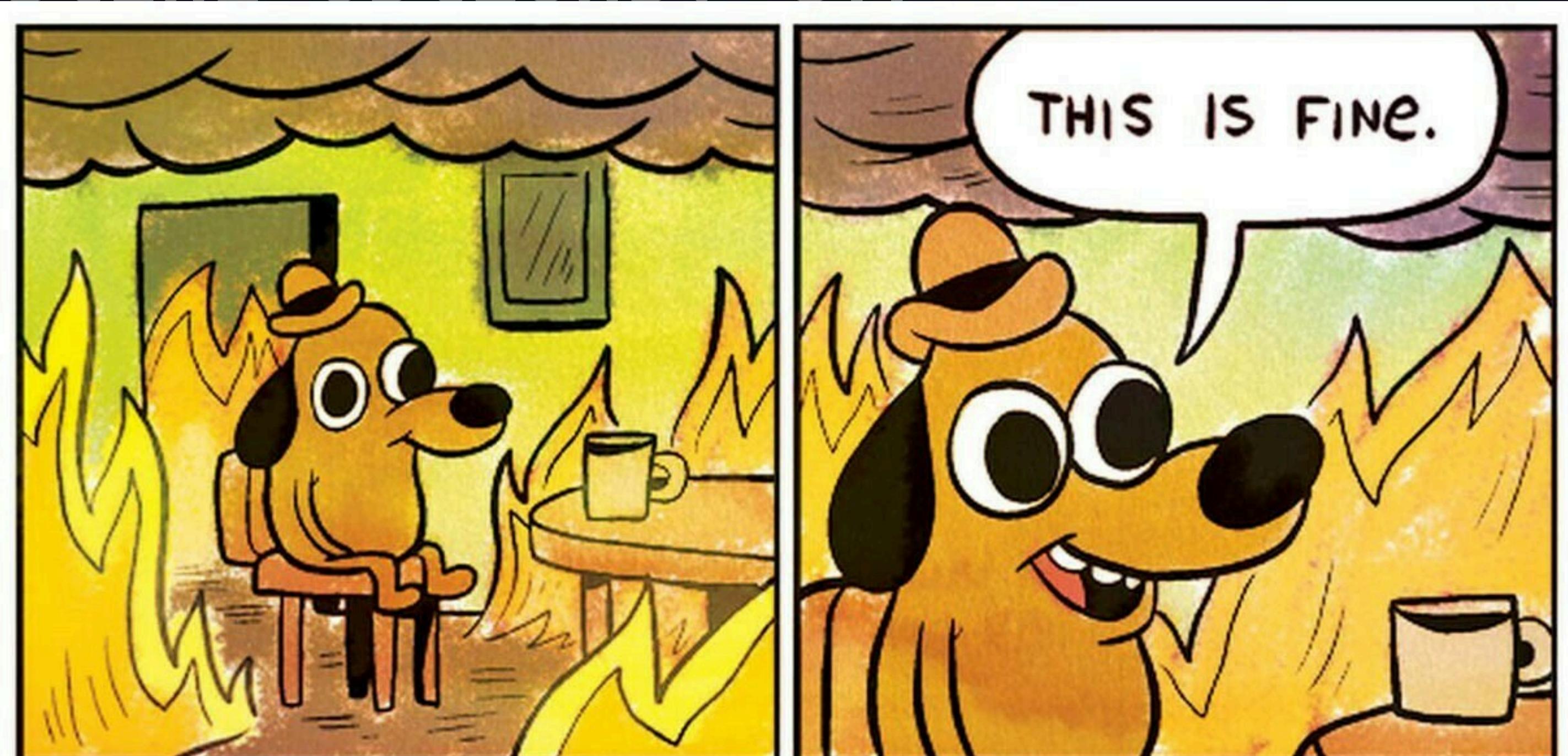
You're reviewing code in a new framework...
...where do you even start? 🤔

Existing Options: CTF

1. Learn vulnerabilities in isolation
2. Not representative
3. Unrealistic patterns
4. Rewarded for exploitation, not understanding

Existing Options: CTF

1.
2.
3.
4.



Existing Options: Blog Posts

1. One-off examples, hard to reproduce
2. Different styles, languages, versions
3. Outdated or missing fixes

The image shows four separate web snippets arranged horizontally, each from a different source:

- GitHub:** A snippet from a repository titled "Example of a Flask API vulnerable to SQL Injection ...". It includes a GitHub icon and a link to <https://github.com/guilatrova/flask-sqlinjection-vulne...>.
- DEV Community:** A snippet from a post titled "Express.js Security Best Practices". It includes a DEV icon and a link to <https://dev.to/tristankalos/expressjs-security-best-pra...>.
- Escape DAST:** A snippet from a blog post titled "How to secure APIs built with FastAPI: A complete guide". It includes an Escape DAST icon and a link to <https://escape.tech/Blog>.
- FastAPI Applications:** A snippet from a blog post titled "How to secure APIs built with FastAPI: A complete guide". It includes a FastAPI icon and a link to <https://escape.tech/Blog>.

Solution: Unsafe Code Lab

The screenshot shows a GitHub repository page for 'unsafe-code'. The repository is private and has 1 branch and 0 tags. The main branch has 143 commits from user 'Irench1k'. The commits are listed in reverse chronological order, with the most recent being 'Regenerated all READMEs' (4 days ago). Other commits include 'Add automated link validation to documentation system' and 'Add AI automation for consistent docs quality'. The repository has 2 stars, 0 forks, and 0 watching. It also has 0 releases and 0 packages published. The contributors are 'execveat' (Andrew Konstantinov) and 'Irench1k' (Irina Iarlykanova).

Commit Message	Date	Author
Regenerated all READMEs	4 days ago	Irench1k
Add automated link validation to documentation system	last week	
Add AI automation for consistent docs quality	last week	
Add automated link validation to documentation system	last week	
Now READMEs will include links to the source code after ...	4 days ago	
Regenerated all READMEs	4 days ago	
Update the typos and regenerate README.md from th ...	2 months ago	
Move to uv for python management	2 months ago	
CLAUDE.md		
README.md		
annotations.md		
pyproject.toml		
uv.lock		

1. Realistic & runnable vulnerable code
2. Modern frameworks
3. Built for AppSec & Research

Case Study: Same Bug, Different Stack

Confusion Attacks

Flask: Authorization Binding Drift # Example 1

```
def basic_auth(f):
    @wraps(f)
    def decorated_basic_auth(*args, **kwargs):
        auth = request.authorization
        if not auth or not
        authenticate(auth.username, auth.password):
            return response_401()

        g.user = auth.username
        return f(*args, **kwargs)

    return decorated_basic_auth
```

```
@bp.get("/example3/groups/<group>/messages")
@basic_auth_v2
@check_group_membership_v2
def example3_group_messages(group):
    return get_group_messages(group)
```

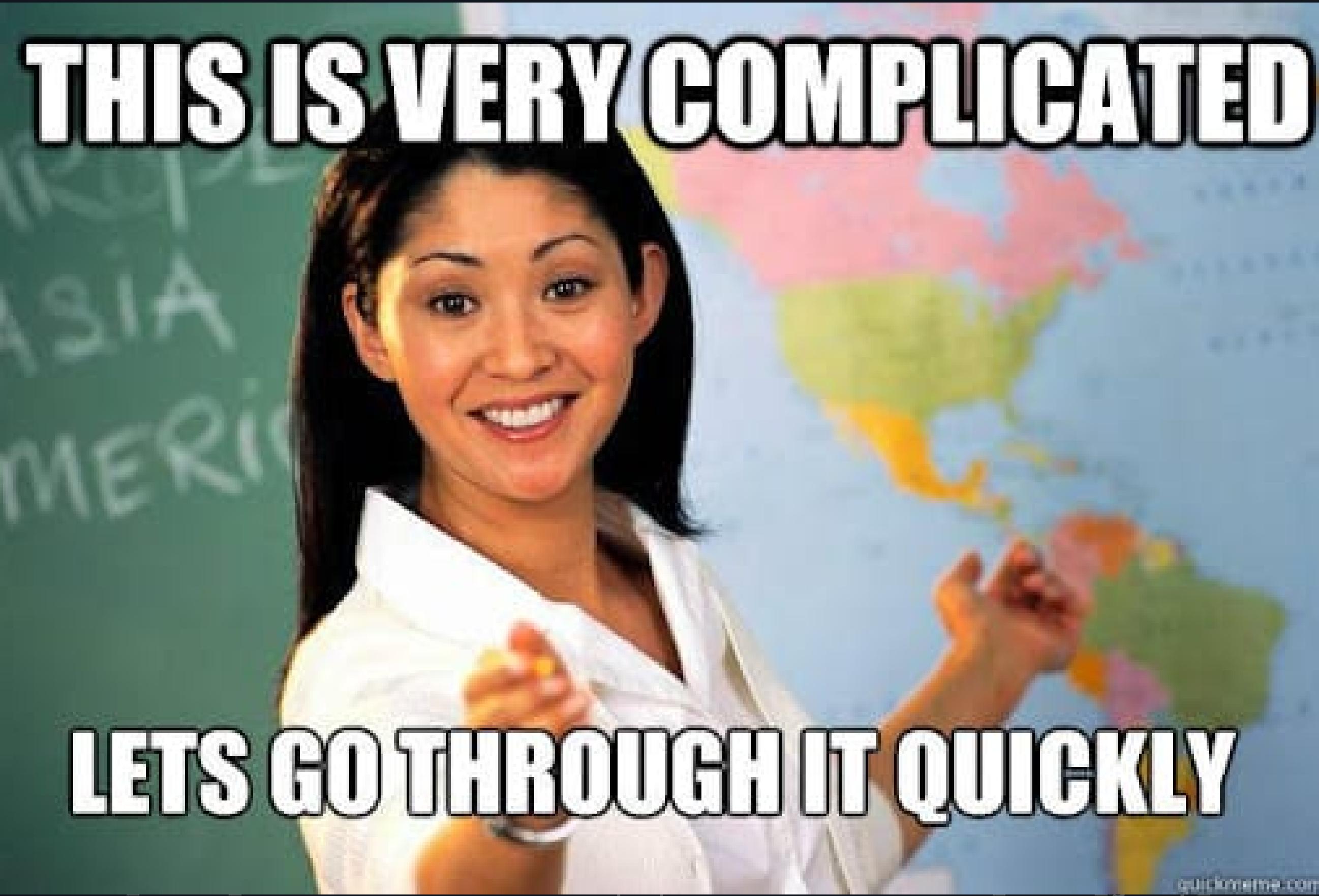
```
def check_group_membership(f):
    @wraps(f)
    def decorated_check_group_memb(*args, **kwargs):
        group = request.args.get("group") or
                request.view_args.get("group")

        if group and not is_group_member(g.user,
                                         group):
            return response_403()

        return f(*args, **kwargs)
    return decorated_check_group_memb
```

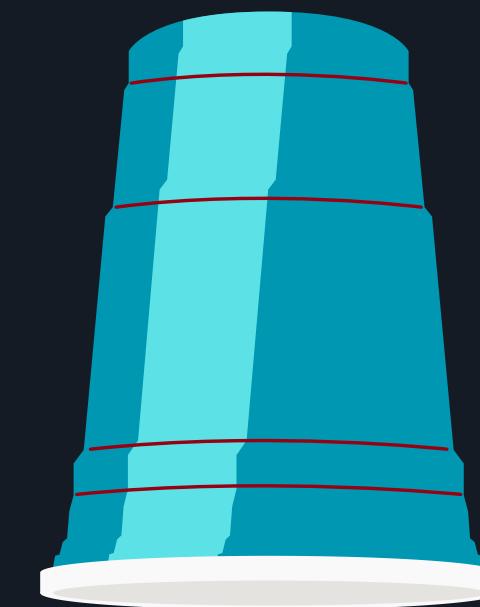
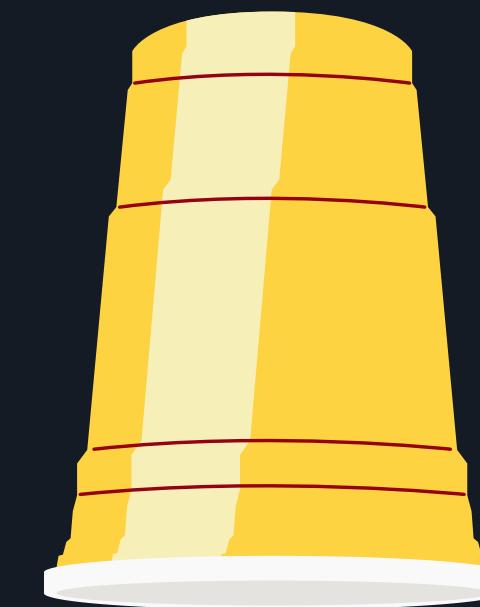
Flask:

```
def basic_auth_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not auth.username or not auth.password:
            return make_response('Missing or invalid credentials', 401, {'WWW-Authenticate': 'Basic realm="Login Required"'})
        g.user = User.query.filter_by(username=auth.username).first()
        if g.user is None or not g.user.check_password(auth.password):
            return make_response('Invalid credentials', 401, {'WWW-Authenticate': 'Basic realm="Login Required"'})
        return f(*args, **kwargs)
```



Sample 3

What's source confusion?



GET {base}/**spongebob/messages**



GET {base}/spongebob/messages?user=squidward



POST {base}/spongebob/messages?user=squidward

Content-Type: application/x-www-form-urlencoded

user=mr.krabs



Flask: Source Precedence # Example 1

```
POST /messages  
Content-Type: application/x-www-form-urlencoded  
  
user=squidward&password=clarinet123
```



```
200 OK:  
Returns messages for 'squidward'
```

Flask: Source Precedence # Example 1

```
POST /messages?user=spongebob
Content-Type: application/x-www-form-urlencoded

user=squidward&password=clarinet123
```



```
200 OK:
Returns messages for 'squidward'
```

Flask: Source Precedence # Example 1.1

```
def get_parameter(key):  
    return request.args.get(key) or request.form.get(key)
```

Flask: Source Precedence # Example 1.2

```
def list_messages(key):  
    return messages_get(request.values.get("user"))  
  
def authenticate_user():  
    return authenticate(request.form.get("user"),  
                        request.form.get("password"))
```

Flask: Source Precedence # Example 1.2

```
def list_messages(key):  
    return messages_get(request.values.get("user"))  
  
def authenticate_user():  
    return authenticate(request.form.get("user"),  
                        request.form.get("password"))
```

request.values.get("user")

=

request.args.get(key) or request.form.get(key)

Flask: HTTP Parameter Pollution # Example 2

```
POST /transfer?to=squidward&amount=50  
Authorization: Bearer <spongebob_session_token>
```



200 OK "Transfer successful"

Flask

POST
Autho

ple 2



400
Bad Request

Flask: HTTP Parameter Pollution # Example 2

```
POST /transfer?to=squidward&amount=50&amount=5000  
Authorization: Bearer <spongebob_session_token>
```



```
200 OK "Transfer successful"
```



Flask: Source Precedence # Example 3

```
GET /admin/plankton/profile  
Authorization: Bearer <plankton_token>
```



```
200 OK with Plankton's profile
```

Flask: Source Precedence # Example 3

```
GET /admin/plankton/profile?employee_name=squidward  
Authorization: Bearer <plankton_token>
```



200 OK with Squidward's profile

Flask: Authorization Binding Drift # Example 4

```
def basic_auth(f):
    @wraps(f)
    def decorated_basic_auth(*args, **kwargs):
        auth = request.authorization
        if not auth or not
        authenticate(auth.username, auth.password):
            return response_401()

        g.user = auth.username
        return f(*args, **kwargs)

    return decorated_basic_auth
```

```
@bp.get("/example3/groups/<group>/messages")
@basic_auth_v2
@check_group_membership_v2
def example3_group_messages(group):
    return get_group_messages(group)
```

```
def check_group_membership(f):
    @wraps(f)
    def decorated_check_group_memb(*args, **kwargs):
        group = request.args.get("group") or
                request.view_args.get("group")

        if group and not is_group_member(g.user,
                                         group):
            return response_403()

        return f(*args, **kwargs)
    return decorated_check_group_memb
```

Flask: Authorization Binding Drift # Example 4

```
@bp.get("/example3/groups/<group>/messages")
@basic_auth_v2
@check_group_membership_v2
def example3_group_messages(group):
    return get_group_messages(group)
```

group = **request.args.get("group") or request.view_args.get("group")**

```
auth = request.authorization
if not auth or not
authenticate(auth.username, auth.password):
    return response_401()

g.user = auth.username
return f(*args, **kwargs)

return decorated_basic_auth
```

```
group = request.args.get("group") or
request.view_args.get("group")

if group and not is_group_member(g.user,
group):
    return response_403()

return f(*args, **kwargs)
return decorated_check_group_memb
```

GET {{base}}/groups/**staff@krusty-krab.sea**/messages

Authorization: Basic **plankton@chum-bucket.sea:burgers-are-yummy**



403
Forbidden

```
GET {{base}}/groups/staff@krusty-krab.sea/messages?group=staff@chum-bucket.sea  
Authorization: Basic plankton@chum-bucket.sea:burgers-are-yummy
```



```
200 OK:  
Returns messages for 'staff@krusty-krab.sea'
```

```
get_group_messages(group)
```

```
GET {{base}}/groups/staff@krusty-krab.sea/messages?  
group=staff@chum-bucket.sea  
Authorization: Basic plankton@chum-bucket.sea:burgers-  
are-yummy
```

```
is_group_member(request.args.get("group"))
```

So, why should I care? 😵

CherryPy: Same Pattern, Different Syntax

```
class MessageController:  
    @cherrypy.expose  
    def messages(self, group):  
        group_for_auth = cherrypy.request.params.get("group")  
        if not is_group_member(cherrypy.request.login,  
                               group_for_auth):  
            raise cherrypy.HTTPError(403, "Forbidden")  
    return get_group_messages(group)
```

CherryPy: Same Pattern, Different Syntax

```
class MessageController:  
    @cherrypy.expose  
    def messages(self, group):  
        group_for_auth = cherrypy.request.params.get("group")  
        if not is_group_member(cherrypy.request.login,  
                               group_for_auth):  
            raise cherrypy.HTTPError(403, "Forbidden")  
    return get_group_messages(group)
```

...and more examples also found in Express.js, Ruby on Rails, and PHP...

Django & FastAPI

Django

SECURE BY DESIGN:

- Strictly separates parameter sources.
- Makes source confusion hard to do by accident.

FastAPI

SECURE BY DEFAULT:

- Requires explicit declaration of the parameter's source.
- A validation layer enforces this rule before your code ever runs.

Some tips on how to avoid Confusion Attacks

1. Be Specific
2. Be Consistent
3. Choose Wisely



Unsafe Code is more than that :)

	Flask	CherryPy	Django	FastAPI	ExpressJS
Confusion	✓	✓	✓	✓	✓
Policy Composition	✓	→ SOON	→ SOON	→ SOON	→ SOON
Trust Boundary Errors	✓	→ SOON	→ SOON	→ SOON	→ SOON
Cross-Component Semantics	✓	→ SOON	→ SOON	→ SOON	→ SOON

and more coming!

Why even bother?

1. Who should care?
2. How it can be used?
3. Can't I just use AI?

How to contribute?

Oh, it's so nice of you!!! :)

1. Go to our git and submit your code!!
2. Email me your questions
3. Mention us on your socials



QR code to the git repository



MANY THANKS !



irina.iarlykanova@gmail.com



<https://github.com/Irench1k/unsafe-code/>