HELLO, I'M DANIELA 👋

@sericaia

DISCLAIMER

*I am **not an expert** on web security*

What is security about?

LET'S PLAY

"I'VE NEVER..."

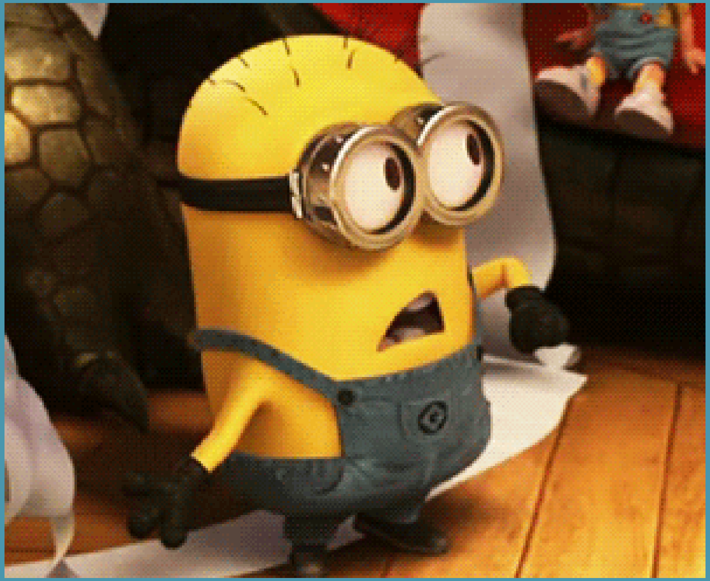
...ran in the rain
...danced on the street

I'VE NEVER "CHANGED A WEBSITE'S URL"

```
http://somewebsite.com/articles?
user=12345&role=EDITOR
```

```
http://somewebsite.com/login/admin
```

HOW DOES ALL THIS RELATES TO OWASP?



WHAT DOES OWASP STAND FOR?

Open Web Application Security Project

TOP 10 SECURITY ISSUES

TOP 10 SECURITY ISSUES 🤓

- A1:2017 - Injection
- A2:2017 - Broken Authentication
- A3:2017 - Sensitive Data Exposure
- A4:2017 - XML External Entities (XXE)
- A5:2017 - Broken Access Control
- A6:2017 - Security Misconfiguration
- A7:2017 - Cross-Site Scripting (XSS)
- A8:2017 - Insecure Deserialization
- A9:2017 - Using Components with Known Vulnerabilities
- A10:2017 - Insufficient Logging & Monitoring

IMPORTANT NOTE!

SOMETIMES EXAMPLES FIT MORE THAN ONE OWASP ISSUE

I'VE NEVER "CHANGED A WEBSITE'S URL"

```
http://somewebsite.com/articles?user=12345&role=EDITOR
```

```
http://somewebsite.com/login/admin
```

A5 - Broken Access Control

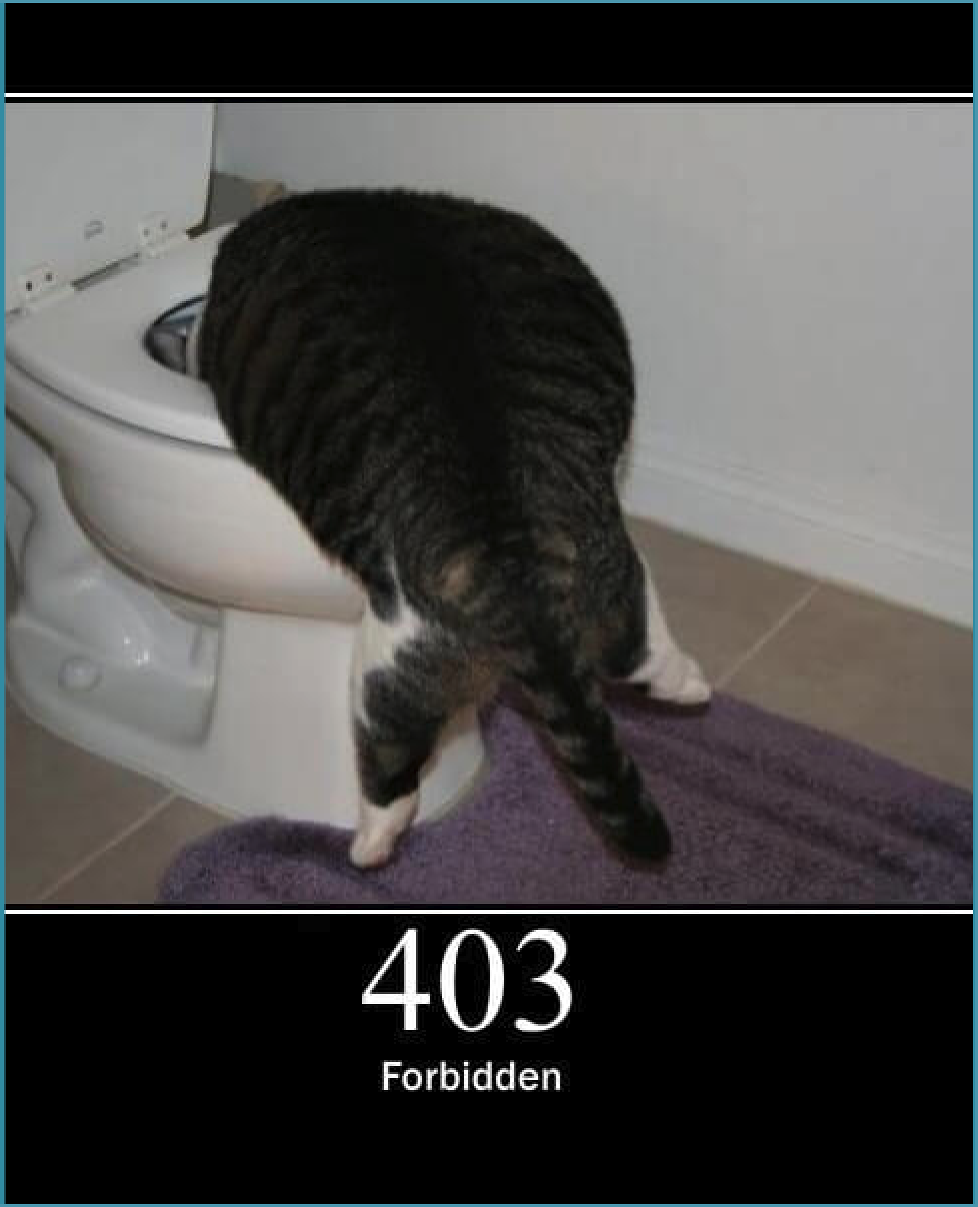
(image source https://pixabay.com/photos/glass-shattered-window-destruction-984457/)

I'VE NEVER "HACKED A WEBSITE URL TO CHECK ACCESS TO ADMIN DATA"

Use `SOME_KEY` in public and (expected) authenticated resources

```
http://somewebsite.com/admin?apiKey={SOME_KEY}
```

I'VE NEVER "UNDERSTOOD REST HTTP CODES"



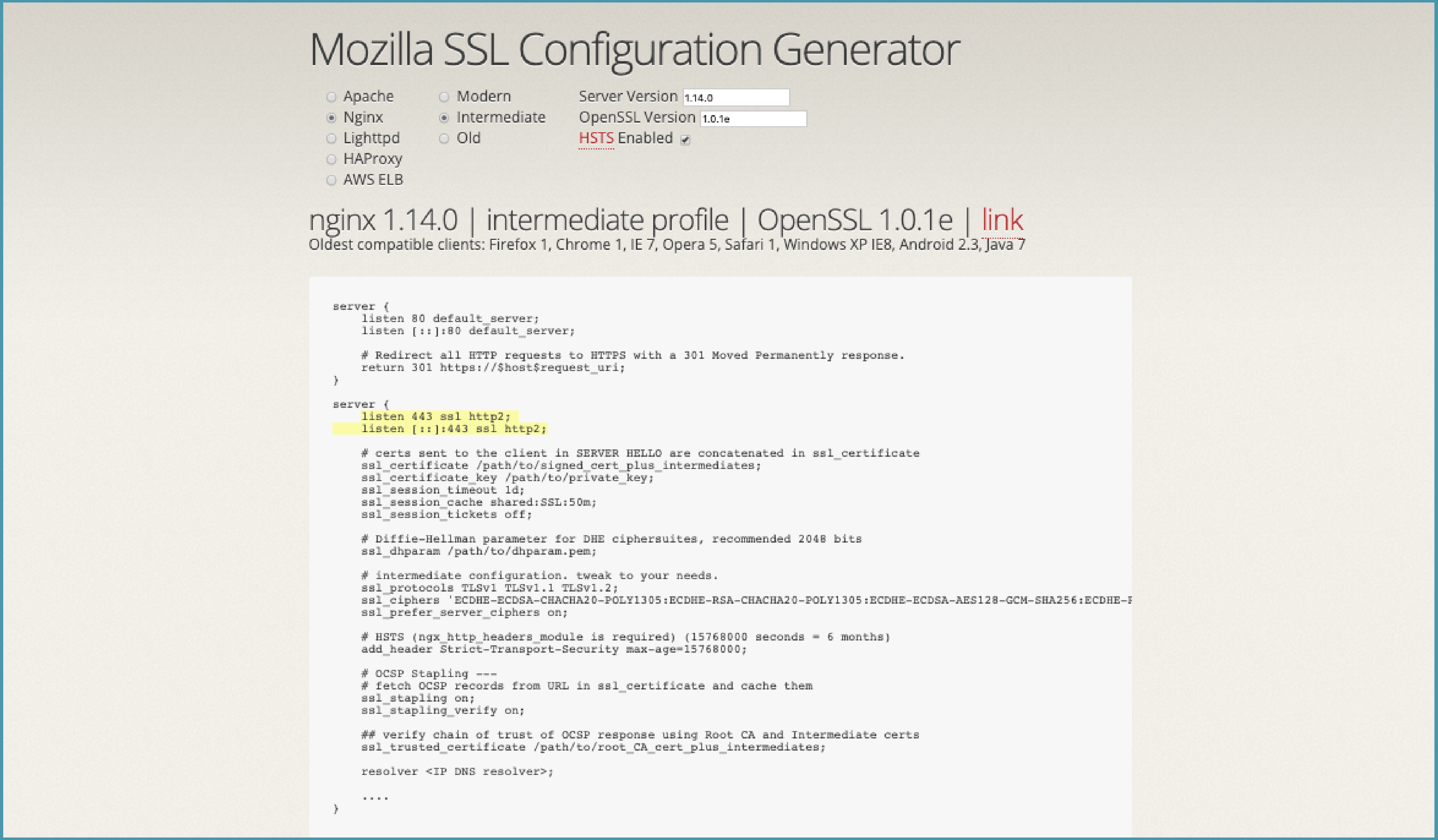
A2 - Broken Authentication

(image source https://pixabay.com/illustrations/eye-iris-biometrics-2771174/)

I'VE NEVER "USED REAL IDS IN URLS"

```
http://somewebsite.com/users/12345
```

I'VE NEVER "SET UP A CERTIFICATE"



Mozilla SSL Configuration Generator

( ) Apache [✓ Nginx] ( ) Lighttpd ( ) HAProxy ( ) AWS ELB

( ) Modern [✓ Intermediate] ( ) Old

Server Version 1.14.0
OpenSSL Version 1.0.1e
HSTS Enabled ✓

nginx 1.14.0 | intermediate profile | OpenSSL 1.0.1e | link

Oldest compatible clients: Firefox 1, Chrome 1, IE 7, Opera 5, Safari 1, Windows XP IE8, Android 2.3, Java 7

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # Redirect all HTTP requests to HTTPS with a 301 Moved Permanently response.
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    # certs sent to the client in SERVER HELLO are concatenated in ssl_certificate
    ssl_certificate /path/to/signed_cert_plus_intermediates;
    ssl_certificate_key /path/to/private_key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;

    # Diffie-Hellman parameter for DHE ciphersuites, recommended 2048 bits
    ssl_dhparam /path/to/dhparam.pem;

    # intermediate configuration. tweak to your needs.
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-
    ssl_prefer_server_ciphers on;

    # HSTS (ngx_http_headers_module is required) (15768000 seconds = 6 months)
    add_header Strict-Transport-Security max-age=15768000;

    # OCSP Stapling ---
    # fetch OCSP records from URL in ssl_certificate and cache them
    ssl_stapling on;
    ssl_stapling_verify on;

    ## verify chain of trust of OCSP response using Root CA and Intermediate certs
    ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

    resolver <IP DNS resolver>;

    ....
}
```

I'VE NEVER "USED BCRYPT WITHOUT UNDERSTANDING WHAT SALT IS"

```
const saltRounds = 10;

async function addUser(username, password) {

    const salt = await bcrypt.genSalt(saltRounds);
    const hash = await bcrypt.hash(password, salt);

    // store hash in DB
}
```

DEMO

A3 - Sensitive Data Exposure

(image source https://pixabay.com/photos/credit-card-charge-card-money-1583534/)

I'VE NEVER "LEFT A TOO DETAILED ERROR MESSAGE"

```
your-email@your-provider.com does not exist in
our database. Try again.
```

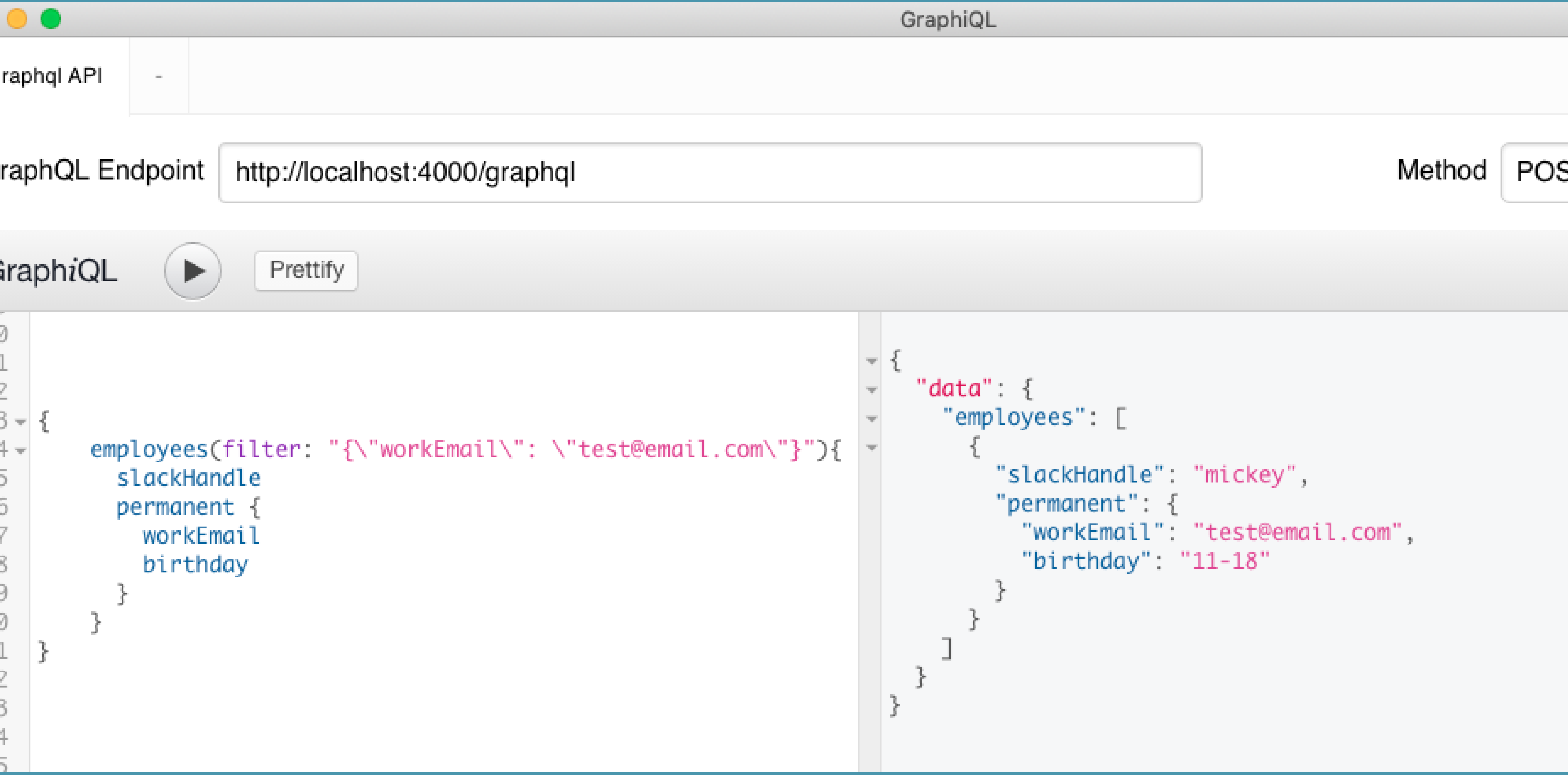
I'VE NEVER "SET THE WRONG SECURITY HEADERS"

```
Set-Cookie: "daenerys=targaryen; HttpOnly"
```

A6 - Security Misconfiguration

(image source https://pixabay.com/photos/electrical-planning-3536767/)

I'VE NEVER "FORGOT THAT USERS CAN BE MAD AND LEAVE ROOM FOR INJECTION" (1/2)

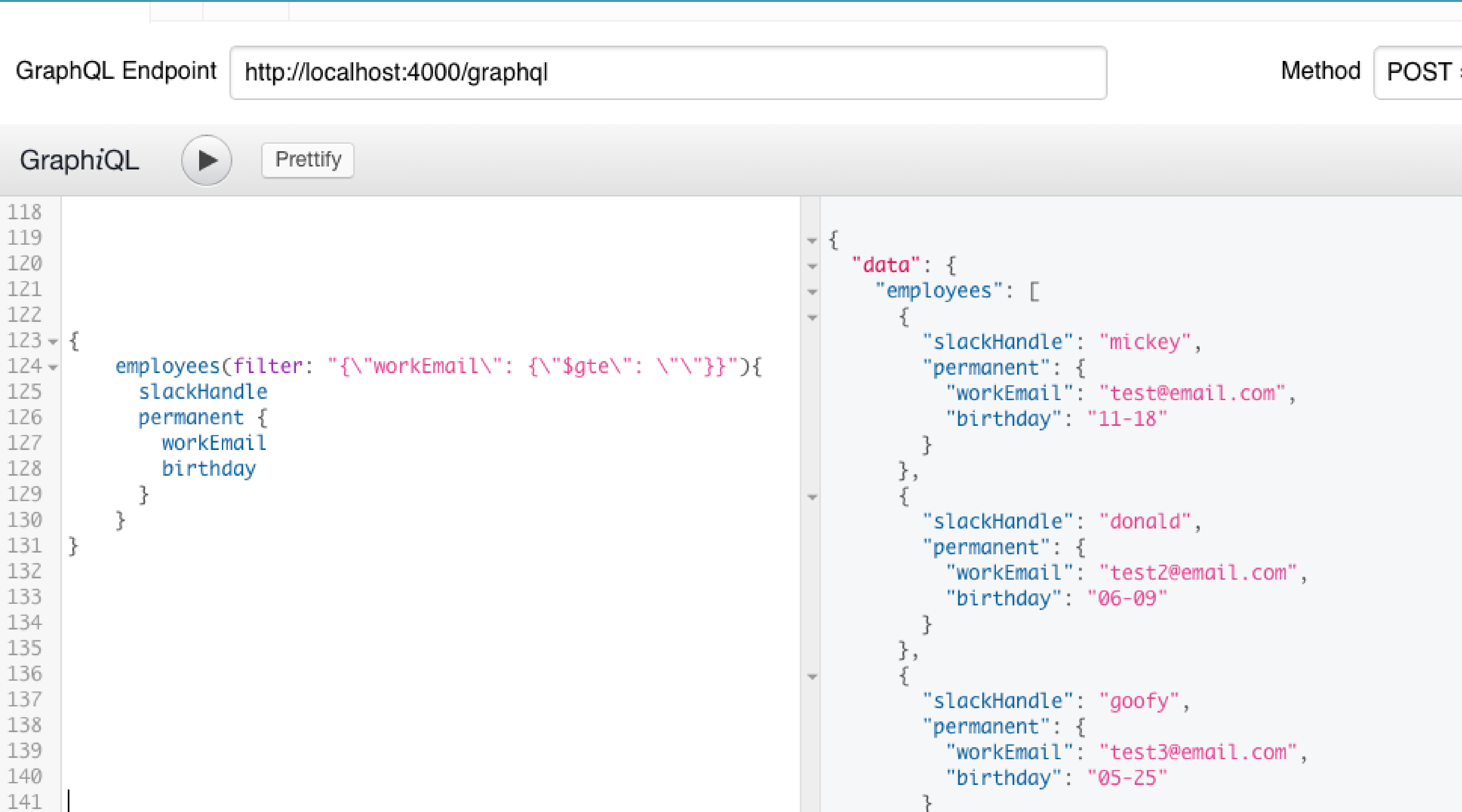


http://localhost:4000/graphql

```
{
    employees(filter: "{\"workEmail\": \"test@email.com\"}"){
      slackHandle
      permanent {
        workEmail
        birthday
      }
    }
}
```

```
{
  "data": {
    "employees": [
      {
        "slackHandle": "mickey",
        "permanent": {
          "workEmail": "test@email.com",
          "birthday": "11-18"
        }
      }
    ]
  }
}
```

I'VE NEVER "FORGOT THAT USERS CAN BE MAD AND LEAVE ROOM FOR INJECTION" (2/2)



```
{
    employees(filter: "{\"workEmail\": {\"$gte\": \"\"}}"){
      slackHandle
      permanent {
        workEmail
        birthday
      }
    }
}
```

```
{
  "data": {
    "employees": [
      {
        "slackHandle": "mickey",
        "permanent": {
          "workEmail": "test@email.com",
          "birthday": "11-18"
        }
      },
      {
        "slackHandle": "donald",
        "permanent": {
          "workEmail": "test2@email.com",
          "birthday": "06-09"
        }
      },
      {
        "slackHandle": "goofy",
        "permanent": {
          "workEmail": "test3@email.com",
          "birthday": "05-25"
        }
```

I'VE NEVER "SET LIMITS ON REQUESTS" (1/2)

`http://somewebsite.com/resetpassword`

I'VE NEVER "SET LIMITS ON REQUESTS" (2/2)

```
query {
  employees {
    name
    email
    manager {
      name
      email
      manager {
        name
        email
        manager {
          // ...
        }
      }
    }


  }
}
```

A1 - Injection

(image source https://pixabay.com/illustrations/syringe-pill-capsule-morphine-1884784/)

I'VE NEVER "ESCAPED USER INPUT"

DEMO

I'VE NEVER "IMPLEMENTED STRICT CSP RULES"

```
Content-Security-Policy: "default-src https:;"
```

DEMO

A7 - Cross-Site Scripting (XSS)

(image source https://pixabay.com/photos/camera-backpack-theft-steal-2292843/)

I'VE NEVER "FORGOT TO VALIDATE USER DATA PROPERLY"

DEMO

I'VE NEVER "GAVE EXTRA PERMISSIONS TO EXTERNAL LINKS"

```
<link
  rel="stylesheet"
  href="https://.../4.3.1/bootstrap.min.css"
  integrity="sha384-SOME_SHA"
  crossorigin="anonymous"
/>
```

A8 - Insecure Deserialization

(image source https://pixabay.com/photos/suit-business-man-business-man-673697/)

TOP 10 SECURITY ISSUES 🤓

- A1:2017 - Injection
- A2:2017 - Broken Authentication
- A3:2017 - Sensitive Data Exposure
- A4:2017 - XML External Entities (XXE)
- A5:2017 - Broken Access Control
- A6:2017 - Security Misconfiguration
- A7:2017 - Cross-Site Scripting (XSS)
- A8:2017 - Insecure Deserialization
- A9:2017 - Using Components with Known Vulnerabilities
- A10:2017 - Insufficient Logging & Monitoring

THANK YOU! 👋 QUESTIONS?

Daniela Matos de Carvalho *@sericaia*

Playing with blocks, huh?
No. Teaching him how to create STRONG PASSWORDS.
MIT IST

(image source https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiFm8T4ysDhAhUMXRoKHXbTABwQjRx6BAgBEAU&url=https%3A%2F%2Fmedium.com%2F%40chumworth%2Fteach-your-children-well-folks-29be20e29299&psig=AOvVaw2mnzR-3Nbzu6aop6zRQjS2&ust=1554815941821077)

RESOURCES

- General
  - The Open Web Application Security Project (OWASP)
  - Mozilla docs on web security
  - HTTP security report - best practices
  - Common Vulnerabilities and Exposures (CVE)
    - CVE For Hapi.js
    - CVE For Express.js
  - Blogpost: "We're under attack! 23+ Node.js security best practices"
  - Blogpost: "Web Security basics by Martin Fowler"
  - Blogpost: "A Tale of (prototype) Poisoning"
  - Free Security training for developers

RESOURCES

- YLD blogposts on web security
  - Blogpost: "Security Trivia Series: Hints on default-src CSP directive"
  - Blogpost: "Security Trivia Series: Understanding CSP's Reporting"

- Site Scanners
  - SSL labs
  - Security headers

RESOURCES (NODE.JS)

- Useful modules
  - bcrypt, save sensitive data encrypted
  - helmet, helps to set HTTP headers on Express.js apps
  - cors, middleware to implement CORS on Express.js apps
  - Hapi.js security modules
  - dompurify, helps to sanitize inputs and request params
  - eslint-plugin-security, to detect security gaps
  - passport.js, authentication Express.js middleware
  - authentication strategies in Hapi.js

RESOURCES (NODE.JS)

- Node.js Vulnerability scanning
  - npm audit
  - snyk
  - Retire.js