Goals

> We want to forge a signature where the role is `Admin` instead of `Guest`

We are given a form where we can sign up a new user which we can trigger via a POST request:

```
POST /register HTTP/1.1
Host: 945c05c8eb1051f7e1c15d5f-auth-token.challenge.master.cscg.live:31337
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:97.0) Gecko/20100101
Firefox/97.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Origin: https://945c05c8eb1051f7e1c15d5f-auth-
token.challenge.master.cscg.live:31337
Dnt: 1
Referer: https://945c05c8eb1051f7e1c15d5f-auth-
token.challenge.master.cscg.live:31337/register
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

email=XXX&first-name=XXX&last-name=XXX&spam=off
```

Furthermore, we have the code which explains how the server does create a signature. Below is the important part, where the token is a JSON object which has to contain the fields `email, fullname, role, signature`.

```
def _sign_token(token):
  # Fetch secret key from temp file. Generate it the first time.
  keypath = '/tmp/secure-secret.key'
  if not os.path.exists(keypath):
    with open(keypath, 'w') as keyfile:
      keyfile.write(hexlify(os.urandom(16)).decode())

  # Compute HMAC over token fields.
  sign_data = token['email'] + token['role'] + token['fullname']
  with open(keypath, 'r') as keyfile:
    key = unhexlify(keyfile.read().strip())
    return hmac.new(key, sign_data.encode(), 'sha256').hexdigest()
```

Observations

1. `fullname = first-name + ' ' + 'last-name'`, not in the code
2. Mail format is not checked by the server, can be anything
3. The signature is only computed over the **values of the fields without the keys**
4. The order is `token['email'] + token['role'] + token['fullname']` such that we can abuse the input for role

Signature Forge

First of all, we create a valid signature for

```
email=&first-name=Admin&last-name=&spam=off
```

which returns us the following jwt like token (where the signature should be outside ...):

```
{"email": "", "fullname": "Admin ", "role": "Guest", "signature":
"68a967fcc65b6db04959b3b21542ce1808d7228ec5495ab9c17080fe46c94737"}
```

By default, we are assigned the role `Guest`. The signature is computed as follows:

```
SIG({"email": "", "fullname": "Admin ", "role": "Guest"}) = SIG("" +
"Guest" + "Admin ") = SIG("GuestAdmin ")
```

To obtain the flag, we get `/admin` with a valid signature and a token with the role of `Admin`. We take the json from above and do some cut and paste

```
{"email": "Guest", "fullname": " ", "role": "Admin", "signature":
"68a967fcc65b6db04959b3b21542ce1808d7228ec5495ab9c17080fe46c94737"}
```

When we compute the signature for this token, we get:

```
SIG({"email": "Guest", "fullname": " ", "role": "Admin"}) = SIG("Guest" +
"Admin" + " ") = SIG("GuestAdmin ")
```

Therefore, we are greeted with a flag 😀

What we learn

Always compute the signature over the whole object, including keys. However, one may not reinvent the wheel when we have the possibility to use JWT based token.