

Devoir II – CEG 4399



CEG 4399 - Design of Secure Computer Sys.

Université d'Ottawa

Professeur : Amir H. Razavi

Noms et numéros des étudiants :
Gbegbe Decaho Jacques 300094197

Date de soumission: 03 November 2024

Assignment 2

Introduction

UNIVERSITY OF Ottawa Faculty of Engineering and Computer Science CSI 4139-CEG 4399 - SEC 5100 – Fall 2024 Assignment 1

Problem 1-1. Password Cracker (10 %)

You're working for the NSA and have been assigned the task of breaking into the most top-secret government agencies. To do so, you must crack as many passwords that have been hashed using the MD5 algorithm as you can. The hashes have been included in a separate file (hashes.txt) and have been divided into weak, moderate, and strong categories. Try to break as many of them as you can! Please submit three Python lists of cracked passwords for each category. Leave passwords that you are unable to break as an empty string.

For example:

Weak:

1. 1f3870be274f6c49b3e31a0c6728957f
2. fe01d67a002dfa0f3ac084298142eccd

Answer: # Weak:

```
[ 'apple', ]
```

The image shows two screenshots of a Linux desktop environment. Both screenshots feature a terminal window titled 'password_crack.py' running in a terminal emulator. The first screenshot shows the script code, which includes imports for hashlib and multiprocessing, a function to create a hash map from a dictionary file, and a function to crack hashes using the precomputed hash map. The second screenshot shows the execution of the script, where it prompts for the dictionary file ('rockyou.txt') and then prints out cracked passwords for weak, moderate, and strong categories. The terminal window has a dark theme with light-colored text.

```
import hashlib
from multiprocessing import Pool, cpu_count

# Function to load the dictionary and create a hash map
def create_hash_map(dictionary_file):
    hash_map = {}
    with open(dictionary_file, 'r') as file:
        for word in file:
            word = word.strip()
            hash_value = hashlib.md5(word.encode('utf-8')).hexdigest()
            hash_map[hash_value] = word
    return hash_map

# Function to crack hashes using the precomputed hash map
def crack_hashes(hashes, hash_map):
    cracked_passwords = []
    for i, h in enumerate(hashes, start=1):
        if h in hash_map:
            print(f"{i} - Match found: {hash_map[h]} for hash {h}")
            cracked_passwords.append(f"{i}-{hash_map[h]}")
        else:
            cracked_passwords.append(f"{i}-")
            print(f"{i} - No match for hash {h}")
    return cracked_passwords
```

```
# Specify the dictionary file
dictionary_file = 'rockyou.txt'

# Create a hash map from the dictionary
hash_map = create_hash_map(dictionary_file)

# Crack hashes using the precomputed hash map
cracked_passwords1 = crack_hashes(example_weak_hashes, hash_map)
cracked_passwords2 = crack_hashes(example_moderate_hashes, hash_map)
cracked_passwords3 = crack_hashes(example_strong_hashes, hash_map)

# Print the results as a numbered list
print("\nThe Cracked weak passwords are:", cracked_passwords1)
print("\nThe Cracked moderate passwords are:", cracked_passwords2)
print("\nThe Cracked strong passwords are:", cracked_passwords3)
```

the result are shown below :

```
The Cracked weak passwords are: ["1-'jennifer'", "2-''", "3-'facebook'", "4-''", "5-'sheep'", "6-'123456'", "7-''", "8-''", "9-''", "10-''", "11-''", "12-'secret'", "13-'password'", "14-''", "15-''", "16-'hunter2'", "17-'banana'", "18-''", "19-''", "20-'nintendo'", "21-''", "22-'qwerty'", "23-'123456'", "24-'banana'", "25-''", "26-''", "27-''", "28-'hello123'", "29-''", "30-''", "31-'12345'", "32-'qwertyuiop'", "33-'123123'", "34-''", "35-'asdf'", "36-''", "37-''", "38-'passw ord123'", "39-''", "40-''", "41-'treasure'", "42-'98765'", "43-'television'", "4 4-'qwerty1234'", "45-'donkey'", "46-'password1'", "47-'google'", "48-'abcde'", "49-'dragon'", "50-'orange'", "51-''", "52-'drowssap'", "53-''", "54-''", "55-''", "56-'abc123'", "57-'letmein'", "58-''", "59-'qwerty'", "60-''", "61-''", "62-''", "63-''", "64-'banana'", "65-''", "66-''", "67-'skeleton'"]
```

```
The Cracked moderate passwords are: ["1-''", "2-''", "3-''", "4-''", "5-''", "6-''", "7-''", "8-''", "9-''", "10-''", "11-''", "12-''", "13-'password123'", "14-''", "15-''", "16-''", "17-''", "18-''", "19-''", "20-''", "21-''", "22-''", "23 -'1a2b3c4d'", "24-''", "25-''", "26-''", "27-''", "28-''", "29-''", "30-''", "31 -''", "32-'poiuytrewq'", "33-''", "34-''", "35-''", "36-''", "37-''", "38-''", "39-''", "40-''", "41-''", "42-''", "43-''", "44-''", "45-''", "46-''", "47-''", "48-''", "49-''", "50-''", "51-''", "52-''", "53-''", "54-''", "55-''", "56-''", "57-''", "58-''", "59-''", "60-''", "61-''", "62-''", "63-''", "64-''", "65-''", "66-''", "67-'"]
```

```
The Cracked strong passwords are: ["1-''", "2-''", "3-''", "4-''", "5-''", "6-''", "7-''", "8-''", "9-''", "10-''", "11-''", "12-''", "13-''", "14-''", "15-''", "16-''", "17-''", "18-''", "19-''", "20-''", "21-''", "22-''", "23-''", "24-''", "25-''", "26-''", "27-''", "28-''", "29-''", "30-''", "31-''", "32-''", "33-''", "34-''", "35-''", "36-''", "37-''", "38-''", "39-''", "40-''", "41-''", "42-''", "43-''", "44-''", "45-''", "46-''", "47-''", "48-''", "49-''", "50-''", "51-''", "52-''", "53-''", "54-''", "55-''", "56-''", "57-''", "58-''", "59-''", "60 -''", "61-''", "62-''", "63-''", "64-''", "65-''", "66-''", "67-'"]
```

Problem 1-2. Enhancing a File Sharing Service with Attribute-Based Access Control (ABAC) (10 %)

A file-sharing service similar to Dropbox or Google Drive wishes to implement Attribute-Based Access Control (ABAC) to manage access to files and folders based on user attributes, resource attributes, and environmental conditions.

- **Design the ABAC model:** Outline the attributes you would use for both users and resources.
- **Policy examples:** Provide examples of access control policies that could be implemented using ABAC.
- **Security benefits:** Discuss the advantages of ABAC over role-based access control (RBAC) in the context of a file-sharing service.

Your submission should include policy pseudo-code and an explanation of how ABAC would handle dynamic access scenarios, such as access requests based on the time of day or the sensitivity of the document.

For the design of the ABAC model :

1. **Design the ABAC model :**

User Attributes

- a. role : Position or function within the organization
- b. Department : Department to which the user belongs
- c. Security clearance : Data access authorization level
- d. User ID : Unique identifier assigned to the user
- e. Job Function : Specific tasks or missions assigned
- f. Location : Geographical or IP-based location of the user.
- g. Active Session : Indicates whether the user is currently logged in.

Resource Attributes

- a. Document access level : Resource classification
- b. File Type : File format
- c. File Owner : User who downloaded or owns the file
- d. Creation Date : Date and time the file was created
- e. Resource ID : Unique identifier assigned to the file or folder.

Environmental Attributes

- a. Access Device : Type of device used to access the resource
 - b. Time of the connection : When the access request is made
 - c. Connection security : Indicates whether the connection is secure
 - d. Geolocation : Location where the access request comes from
-

2. Policy examples :

Policy 1:

Department-Based Access: HR users can access documents with an “Internal” sensitivity level.

```
1  Policy 1 : Department-Based Access:  
2  
3  IF user.department == "HR" AND document.sensitivity == "Internal" THEN  
4      GRANT access  
5  ELSE  
6      DENY access
```

Policy 2:

Time-Limited Access: Managers can access “Confidential” documents only during business hours.

```

8   Policy 2 : Time-Limited Access:
9
10  IF user.role == "Manager" AND document.sensitivity == "Confidential" AND current_time BETWEEN "08:00" AND "16:00" THEN
11      GRANT access
12  ELSE
13      DENY access
14

```

Policy 3:

High-Sensitivity Access Control: Users with “Top Secret” clearance can access “Top Secret” documents only if they are connected via a secure VPN.

```

15  Policy 3: High-Sensitivity Access Control:
16
17  IF user.clearance == "Top Secret" AND document.sensitivity == "Top Secret" AND user.connection_type == "Secure VPN" THEN
18      GRANT access
19  ELSE
20      DENY access

```

Policy 4:

File Owner Edit Rights: The user who owns a file can edit it.

```

22  Policy 4 : File Owner Edit Rights:
23
24  IF user.id == document.owner_id THEN
25      GRANT edit_permission
26  ELSE
27      DENY edit_permission

```

Policy 5:

Location-Based Access: Employees can only access “Sensitive” documents if they are on the organization’s premises or using the organization’s IP range.

```

29  Policy 5 : Location-Based Access:
30
31  IF user.location == "Organization Premises" OR user.ip_address IN organization.ip_range AND document.sensitivity == "Sensitive" THEN
32      GRANT access
33  ELSE
34      DENY access

```

Policy 6:

Device-Specific Access: Users can only access “Confidential” documents from a registered corporate device.

```

36  Policy 6 : Device-Specific Access:
37
38  IF user.device IN organization.registered_devices AND document.sensitivity == "Confidential" THEN
39      GRANT access
40  ELSE
41      DENY access

```

Policy 7:

Multi-factor authentication for critical files: Users attempting to access “highly confidential” documents must verify their identity via multi-factor authentication.

```

43 Policy 7 : Multi-factor authentication for critical files:
44
45 IF user.authentication_level == "Multi-factor" AND document.sensitivity == "Highly Confidential" THEN
46     GRANT access
47 ELSE
48     DENY access

```

Policy 8:

Temporary access for contractors: Contractors can only access “public” or “shared” documents and have restricted access permissions that expire after a set period of time.

```

50 Policy 8 : Temporary access for contractors:
51
52 IF user.role == "Contractor" AND document.visibility IN ["Public", "Shared"] AND current_date <= user.access_expiry_date THEN
53     GRANT access
54 ELSE
55     DENY access

```

Policy 9:

Document version control: Only team leaders can edit or overwrite existing versions of a document, while regular users can only view or comment on them.

```

57 Policy 9 : Document version control:
58
59 IF user.role == "Team Leader" THEN
60     GRANT edit_permission
61 ELSE IF user.role == "Regular User" THEN
62     GRANT view_or_comment_permission
63 ELSE
64     DENY edit_permission
65

```

Policy 10:

Approval-based access: Users requesting access to “restricted” documents must first obtain approval from a designated manager or data owner.

```

66 Policy 10 : Approval-based access:
67
68 IF document.sensitivity == "Restricted" AND user.request_approved == TRUE THEN
69     GRANT access
70 ELSE
71     DENY access

```

3. Security benefits :

- ABAC can be easily modified to accommodate new attributes and meet changing organizational needs without having to reorganize complex role structures, making it more flexible and adaptable.
- ABAC allows access policies to be written that precisely align with regulatory or internal requirements, ensuring compliance with **security** and **privacy standards**.

- ABAC allows nuanced policies to be created using a variety of attributes, unlike RBAC which primarily uses predefined roles. This allows for more contextual control, such as limiting access based on time or login type, facilitating **granular access control**.
- ABAC **easily adapts** to new resource types and attributes without requiring a complete overhaul of the access model, unlike RBAC which requires adjustments and new role definitions.
- ABAC can **grant attribute-based access** for external or temporary users with well-defined restrictions without creating new dedicated roles.
- With ABAC, administrators can manage and modify **access rules** more simply through attributes, making it easier to audit and update policies.

Handling Dynamic Access Scenarios

Location-Based Document Access :

Suppose an employee with a “Top Secret” clearance attempts to access a document marked as “Confidential” while outside the organization’s country. An ABAC policy could evaluate the user’s current location and deny access if they are outside a specified geographic boundary.

Explanation: ABAC’s ability to combine user, resource, and environment attributes enables real-time access decisions—if an employee attempts to access a document while outside a secure geographic location or is using an unregistered device, the system can dynamically deny access. This adaptive nature makes ABAC well-suited for managing complex, context-aware data access, ensuring enhanced data security in file sharing services.

```
≡ handle dynamic acces ×
1 | Pseudo-code
2 |
3 | IF user.security_clearance == "Top Secret"
4 | AND resource.document_sensitivity == "Confidential"
5 | AND environment.user_location NOT IN ("Organization's Country")
6 | DENY access;
7 |
8 | IF user.security_clearance == "Top Secret"
9 | AND resource.document_sensitivity == "Confidential"
10 | AND environment.user_location IN ("Organization's Country")
11 | ALLOW access;
12 |
```

Problem 1-3. Chosen-Ciphertext Attack (10%)

The basic RSA cryptosystem is vulnerable to Chosen-Ciphertext Attack. This means if we have a ciphertext in hand, say c_1 , corresponding to an unknown message, M_1 , and then submit a chosen ciphertext, say c_2 , and receive its corresponding message, M_2 , then we can recover the unknown message, M_1 . Clearly shows how we can do this attack. Then, briefly explain how we can prevent this type of attack.

Note: The assumption is that the attacker knows c_1 , and can choose c_2 , and receive M_2 , such that $E(M_2) = c_2$. The attacker also knows the public key (e, n) , but has no information about the private key d . Now, we would like to know how the attacker, having this info, can find M_1 .

Method of doing the attack :

By choosing a ciphertext ($(c_2 = c_1 \times r^e) \bmod n$) and submitting it, the attacker can use the decrypted response M_2 to deduce the original message M_1 .

Example :

- The attacker has a cryptogram c_1 that corresponds to an unknown message M_1 , such that $c_1 = E(M_1) = M_1^e \bmod n$. The attacker can choose another cryptogram c_2 and receive the corresponding decrypted message M_2 .
- The attacker chooses c_2 as $(c_2 = (c_1 \times r^e) \bmod n)$, where r is a random integer. This operation encrypts the message $M_1 \times r \dots$ since $c_2 = (M_1 \times r)^e \bmod n$.
- The decryption server or oracle decrypts c_2 and returns M_2 such that $M_2 = D(c_2) = M_1 \times r \bmod n$.
- The attacker, knowing M_2 and r , can compute M_1 using the modular inverse of r :
$$M_1 = M_2 \times r^{(-1)} \bmod n$$
.
The inverse $r^{(-1)}$ can be calculated using the extended Euclid algorithm.

Preventing Chosen-Ciphertext Attacks:

Padding Schemes (e.g. OAEP): Using padding schemes like OAEP in RSA adds randomness and structure to messages to prevent significant manipulation of ciphertexts.

Moving to CCA-Secure Protocols: Using cryptographic protocols like RSA-PSS that are resistant to chosen-ciphertext attacks.

Authenticated Encryption: Adding integrity checks or signatures to encrypted messages helps detect and reject altered ciphertexts.

Problem 1.4. Capture the flag (70%)

Goal: This problem introduces the concept of a CTF through the form of web vulnerabilities. Most of the vulnerabilities come from the OWASP top 10. OWASP, which stands for Open Web Application

Security Project is a project dedicated to making the web more secure. There are 7 vulnerabilities in this platform based on the OWASP top 10. Furthermore, vulnerabilities like the ones here have all been found in major production websites at some point.

Setup:

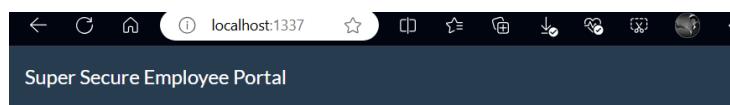
We are using docker. The following steps will have the exercise up and running in docker in no time.

Questions:

Most of the challenges in this exercise have a flag associated with them. A flag is of the format "`flag{some_text_here}`". The idea comes from the concept of a cyber security CTF in which participants hunt for vulnerabilities revealing flags. These vulnerabilities typically reflect real-world security issues.

Question 1

Register for an account. You may need to use SQL injection to help you out here. You will find a flag once your SQL injection is successful

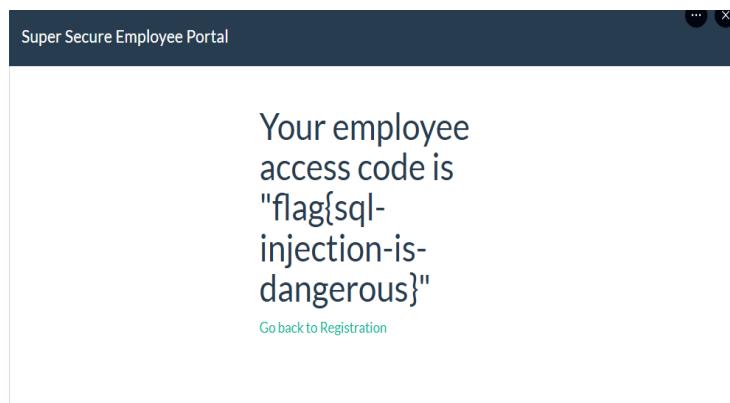


Welcome to the employee portal

get started please either Login or Register using the access code you were given during your training.



Above is the web page we were redirected to
We have a login and register option.



In order to get the flag, we have to enter the values 1111' OR 1 = 1;-- in the recovery Code Field after pressing the (forgot your code) field in the register.
After we registered we got this message above.

Super Secure Employee Portal

First name
dc

Last name
ca

Email
deca@gmail.com

Username
drive

Password

Confirm Password

Employee Access Code ([Forgot your code?](#))
flag(sql-injection-is-dangerous)

[Register](#) [Back](#)

We have to enter the value of the flag and some information in order to register on the page.

Question 2

Create a URL so that the user is redirected after logging in. There is no flag for this challenge. Just record the URL that you used.

localhost:1337/employee?page=dashboard.html

Super Secure Employee Portal Profile Messenger Logout

Hello de ca

Welcome to the employee dashboard. Here you can send messages to the administrator as well as modify your profile. We like to keep it simple for our employees to use!

To get the user to go to another page, like the dashboard, we changed the url to another
The URL page : localhost:1337/employee?page=dashboard.html
This is the page we were directed to after logging in

Question 3

Find a stored cross-site scripting vulnerability and have an alert pop up on the dashboard page. Once successful, a second alert will pop up.

localhost:1337/employee?page=dashboard.html<script>window.alert("")</script>

localhost:1337 indique

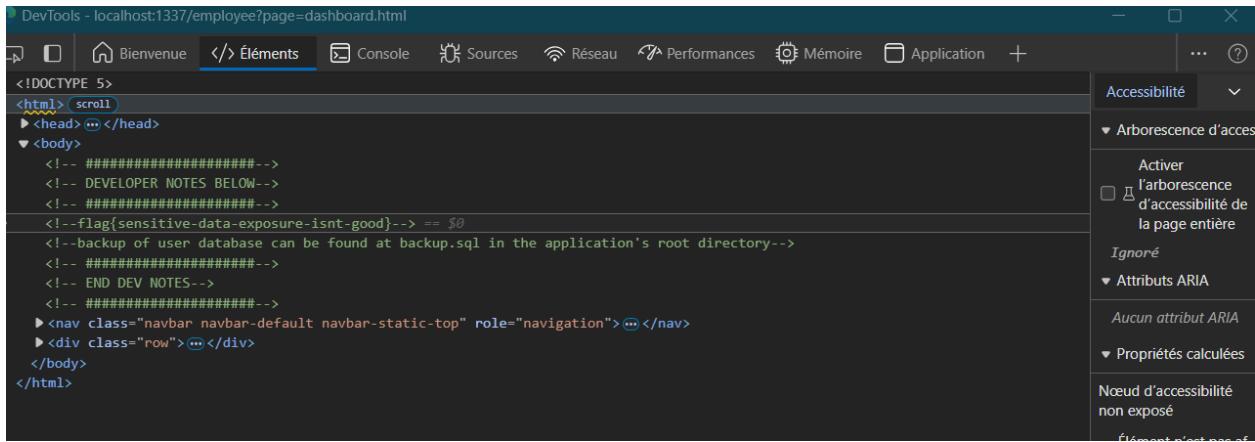
OK

For this vulnerability we added javascript to the url which produced a pop up.

[http://localhost:1337/employee?page=dashboard.html<script>window.alert\(""\)</script>](http://localhost:1337/employee?page=dashboard.html<script>window.alert()

Question 4

A piece of information is exposed on the employee dashboard. Find it and the flag that comes with it.



The screenshot shows the DevTools Element tab for a page at localhost:1337/employee?page=dashboard.html. The page content includes developer notes and an accessibility panel on the right.

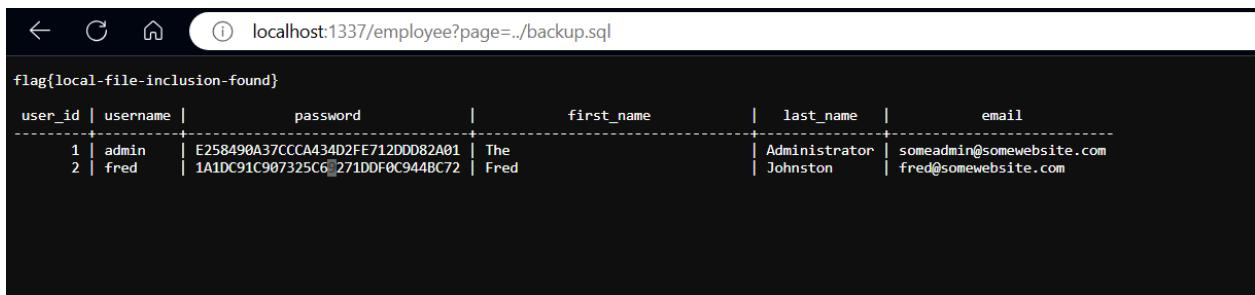
```
<!DOCTYPE 5>
<html> scroll
  ><head> ... </head>
  ><body>
    <!-- #####-->
    <!-- DEVELOPER NOTES BELOW-->
    <!-- #####-->
    <!--flag{sensitive-data-exposure-isnt-good}--> == $0
    <!--backup of user database can be found at backup.sql in the application's root directory-->
    <!-- #####-->
    <!-- END DEV NOTES-->
    <!-- #####-->
  ><nav class="navbar navbar-default navbar-static-top" role="navigation">... </nav>
  ><div class="row">... </div>
</body>
</html>
```

The accessibility panel on the right shows an expanded tree under "Arborescence d'accès".

For this question, we inspected the dashboard page and found sensitive elements as well as an indicator.

Question 5

Based on the information exposed in question 4, are you able to find a file on the system that should not be accessible?



The screenshot shows a browser window with the URL localhost:1337/employee?page=../backup.sql. The page displays a table with user data, indicating a local file inclusion vulnerability.

flag{local-file-inclusion-found}					
user_id	username	password	first_name	last_name	email
1	admin	E258490A37CCCA434D2FE712DD82A01	The	Administrator	someadmin@somewebsite.com
2	fred	1A1DC91C907325C6271DDF0C944BC72	Fred	Johnston	fred@somewebsite.com

To get the database, all I had to do was go to the page:
localhost:1337/employee?page=../backup.sql

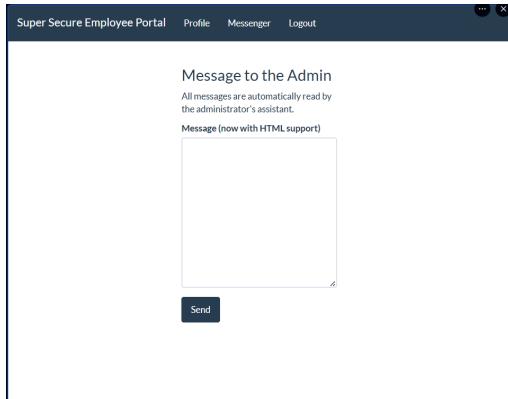
Question 6

Using the result of question 5, you need to **figure out** the password of **Fred**. This is another form of sensitive data exposure.

The password of **Fred** is **1A1DC91C907325C69271DDF0C944BC72**

Question 7

Find a way to **change** the password of the **admin** account. It will be more difficult than figuring out Fred's one... There might be a way through the messaging system. After all, the admin's assistant does automatically read the messages.



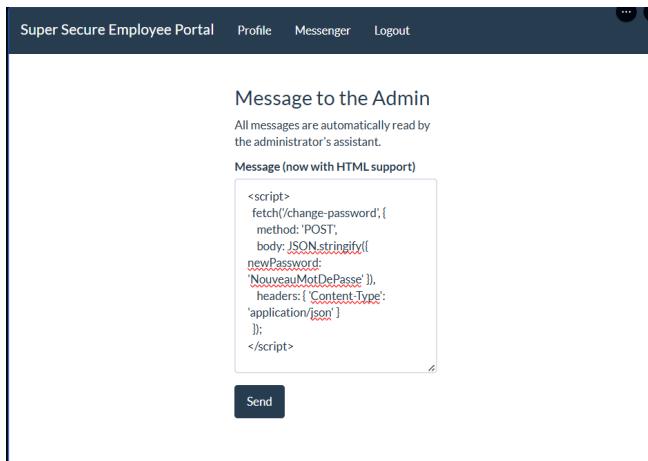
Message sent successfully to the administrator!

Message to the Admin

All messages are automatically read by the administrator's assistant.

Message (now with HTML support)

Send



We launched the script through the message window in order to change the password of the admin Fred. But we failed to change it.