Question 3

a) IMAGES

b)

Voici la formule de l’entropie qui donne la complexité en nombre de bits

En prenant comme hypothèse que la longueur maximale du mot de passe est de 8 caractères voici la complexité selon les alphabètes :

|  |  |  |
| --- | --- | --- |
| **Alphabète** | **Formule** | **Entropie (bits)** |
| [a-zA-Z] |  |  |
| [a-zA-Z0-9] |  |  |
| Table ascii |  |  |

\*Il s’agit d’une valeur théorique car il y a des caractères qui sont impossibles de taper sur le clavier. Nous utilisons 256 pour démontrer que plus le nombre est grand plus l’entropie augmente.

c)

En analysant le tableau ci-haut, il est facile de remarquer que plus l’alphabète est diversifié, plus l’entropie maximale augmente.

d)

En regardant les résultats de John the ripper, on s’aperçoit que les mots de passe n’ont pas un dictionnaire élargie, qu’ils sont courts et qu’ils ne sont pas formé de façon aléatoire.

Voici 3 critères qui augmente l’entropie d’un mot de passe :

1. Diversité du dictionnaire.
2. Longueur du mot de passe.
3. Mot de passe aléatoire.

Question 5 Injection de SQL (SQLi)

1. Proposez une façon de vous loguer sur le site web avec le compte : *gigi*

Pour se connecter avec le compte *gigi* il suffit d’entrer ses informations dans les champs suivant :

|  |  |
| --- | --- |
| Champs | Valeurs entrées |
| user | gigi |
| password | ' OR '1 |

Ainsi requête donne le résultat :

*select mem\_code from MEMBRES where mem\_login = '****gigi****' and mem\_pwd = '****' OR '1****'*

La requête SQL est toujours vrai grâce à la condition OR*'1'.* Donc le mot de passe n’est jamais pris en compte pour se connecter.

1. Proposez une façon pour passer à travers de la partie identification du site en supposant que vous ne connaissez aucun nom de compte

Pour se connecter sans connaitre les identifiants il suffit de rentrer une condition encore toujours vraie comme par exemple :

|  |  |
| --- | --- |
| Champs | Valeurs entrées |
| user | ' OR '1' OR 'true |
| password | x |

*select mem\_code from MEMBRES where mem\_login = '****' OR '1****'* ***OR 'true****' and mem\_pwd = '****x****'*

Il s’agit du même principe que la question précédente.

1. Quelles failles dans le code avez-vous utilisées pour l’attaque 1 ? Et pour l’attaque 2 ?

Les variables en PHP ne sont pas traitées et sont utilisées intégralement dans la requête. Donc, il suffit de mettre un guillemet et on peut commencer à insérer d’autres conditions et d’autres actions.

1. Corrigez les failles dans le code que l’admin de PolyVideo vous a envoyé et mettez le code corrigé dans le rapport.

Il faut nettoyer les variables avec la méthode mysql\_real\_escape\_string() qui ajoute les caractère « \ » devant les caractères suivant : NULL, \x00, \n, \r, \, ', " et \x1a

extract($\_POST);

$Secure\_login = mysql\_real\_escape\_string($login);

$Secure\_pass = mysql\_real\_escape\_string($pass);

$req = "select mem\_code from MEMBRES where mem\_login = '$Secure\_login' and mem\_pwd = '$Secure\_pass'";

$result = mysql\_query($req) or

die ("Error : the SQL request".$req."is not valid: ".mysql\_error());

list($mem\_code) = mysql\_fetch\_array($result);

Question 5 Cross Site Scripting (XSS)

1. Comment avez-vous effectué l’attaque ?

À la suite de la connexion nous avons tenté de modifier nos informations dans la section « Modification Information Personnel ». Dans le champs nom nous avons entré cette ligne de commande :

<script type="text/javascript">document.location.href="http://195.34.45.30/hacked.html"</script>

1. Quelles ont été les failles que vous avez utilisées ? Comment les corriger ?

La faille que nous avons utilisée c’est que dans les champs POST du html de la section « Modification Information Personnel » permettent d’insérer du code JavaScript. Une fois la page chargée le code JavaScript s’exécute. Une façon de corriger le problème de XSS est d’utiliser une méthode qui se nomme « HTML Escaping ». Cette technique convertie les caractères potentiellement dangereux dans leur abréviation. Par exemple :

|  |  |
| --- | --- |
| Caractère | Abréviation |
| & | &amp; |
| < | &lt; |
| > | &gt; |
| " | &quot; |
| ' | &#x27; |
| / | &#x2F; |

Ainsi le rendu HTML sera exactement ce que l’usager a entré mais ne sera pas exécuté comme du JavaScript.

Question 6

1. Donnez la séquence exacte de caractères à entrer. Expliquez brièvement comment votre « hack » fonctionne.

Avec exactement 60 caractères aléatoires dans le champ username l’entré au site est toujours garantie. Il en est de même avec 100, 140 et 180 caractères aléatoires. Notre hack fonctionne en écrasant la valeur du nom d’utilisateur par une valeur arbitraire et en écrasant la valeur du mot de passe par le caractère de fin de ligne. Ainsi lorsque l’on se connecte avec par exemple 60 caractères, le nouveau nom d’utilisateur "root" est remplacé par les 20 derniers caractères entrés et le mot de passe est remplacé par “\0“ qui est ajouté lorsque l’on tente de se connecter. Comme le montre le tableau suivant, les 40 premiers caractères servent à écraser les tableaux user\_name et password.

Voici comment est représenté la mémoire du programme.

|  |  |
| --- | --- |
| Emplacement mémoire | Données |
| [0-19] | user\_name[20] |
| [20-39] | password[20] |
| [40-59] | users[0][0] ("root") |
| [60-79] | users[0][1] ("98765") |
| [80-99] | users[1][0] ("moi") |
| [100-119] | users[1][1] ("allo") |
| [120-139] | users[2][0] ("abc") |
| [140-159] | users[2][1] ("motdepasse") |
| [160-179] | users[3][0] ("") |
| [180-199] | users[3][1] ("") |

1. Que faudrait-il changer dans le programme pour enlever ce problème de sécurité?

Premièrement en mettant un nombre de caractère maximum dans le champ username et password. Il serait aussi possible d’ajouter une variable canarie. Par exemple, en ajoutant une chaine de caractère directement après la déclaration du password et en vérifiant l’intégrité de cette variable avant chaque tentative de connexion.