Protégez votre site en utilisant la technique du « Pot de Miel »

Dans tous les sites Web interactifs, il y a une partie dites “sensible” comme par exemple la partie administrative qui permet d'avoir le contrôle de la quasi-totalité du site.Dans cet article, nous allons apprendre à créer un pot de miel (ou honeypot), plus exactement une fausse partie administrative qui permet d'apprendre comment le pirate s'y prend pour exploiter les vulnérabilités de votre site, mais aussi le décourager/stopper dans la continuation de son acte de piratage, un peu comme le principe du pot de miel.

Cet article explique :

- L'utilité d'un pot de miel.
- Comment mettre en place un pot de miel sur son site, plus exactement une fausse partie administrative.
- Mise en place d'une fausse partie administrative sur un CMS. Exemple avec Joomla.
- Idée d'autres pots de miel et explication des deux types distincts de pots de miel.

Ce qu'il faut savoir :

- Connaissance de base en PHP.
- Notions en POO.
- Connaître les enjeux et les conséquences de la sécurité informatique.

En général, quand un pirate informatique essaye de s'introduire dans un site Web, il va essayer de passer outre la sécurité de la connexion du panneau d'administration soit en exploitant une éventuelle faille de sécurité (injection SQL, …), soit avec un bot qui va tenter toutes les combinaisons de login possibles pour ainsi y accéder et y faire ce qu'il désire.

Une première technique de protection consiste à modifier le nom et l'emplacement du dossier d'administration pour compliquer la tâche du pirate.
Au lieu d'avoir un dossier nommé "admin", "administration" ou encore "admin.php", vous pouvez lui donner un nom difficile à trouver, mais cela n'empêche pas les pirates à adapter une stratégie pour découvrir le nom du dossier.

Alors, tout en gardant le vrai dossier d'administration caché, nous allons simuler une fausse partie d'administration qui va attirer les pirates par cette proie facile qui va non seulement les faire perdre du temps précieux à essayer de se connecter au lieu de chercher la vraie page d'administration qui la rend ainsi immunisé par les attaquants, mais également nous permettre d'analyser la manière qu'ils utilisent pour exploiter les éventuelles vulnérabilités du site Web.

Vous l'avez compris, dans ce tutoriel, je vais donc vous expliquer comment créer une fausse partie administrative qui va servir de pot de miel.

Pour ce faire, renommez votre vrai dossier d'administration de votre site par un nom difficile à deviner mais facile à retenir (ex : /dossier-secret-admin-difficile-a-trouver/).
Puis créer la fausse partie administrative dans un dossier sous le nom "admin" avec un fichier index.php qui demandera l'identifiant de l'administrateur et gérera les traces du pirate qui tentera de se loger dans l'administration de votre site (listing 1).

N'oubliez pas que le design de cette fausse interface d'administration doit ressembler au maximum à votre site, afin que le pirate ne ce doute pas que c'est une fausse page de connexion.

Listing 1. Fichier index.php qui va servir comme fausse page de connexion

```
<?php
/**
 * This page is a fake admin login page of site.
 * Your real login page it just another URL (e.g. http://your-site.com/_my-secret-admin-page/).
 */
require '../../_inc/Sniff.class.php'; // Include the class.
$bErr = false; // Initialize the error variable.

if(isset($_POST['usr'], $_POST['pwd']))
{
    sleep(6); // Security against brute-force attack and this will irritate the hacker...
    $bErr = true; // Display an error message indicating that the login is incorrect.
    new Sniff($_POST['usr'], $_POST['pwd']); // // Class declaration with initialization values connection.
}

/**
 * Checks if the IP address is banned.
 */
if(Sniff::isIpBlock())
{    /**
     * Redirect to the index page.
     * Instead, you can display a message indicating that the user has been banned or another redirection.
                */
    header('Location: ../');
    exit;
}
?>

<!DOCTYPE html>
<html>
  <head>
     <title>Honeypot | Standalone example</title>
```

```
    <link rel="stylesheet" href="./../static/css/general.css" />
    <!-- Your Analytics Code here (e.g. Google Analytics: http://www.google.com/analytics/, Piwik: http://piwik.org) -->
  </head>
  <body>
    <div id="container">

      <header>
        <h1>Honeypot Example</h1>
      </header>

      <h2 class="blue">Admin Panel</h2>
      <form class="center" action="index.php" method="post">
        <fieldset>
          <legend>Login</legend>
          <?php if($bErr) echo '<p class="center warning  block">Your username or password was incorrect. Please try again.</p>'; ?>
          <label for="usr">Username:</label>
          <input type="text" name="usr" id="usr" value="admin" onfocus="if('admin' == this.value) this.value='';" onblur="if('' == this.value) this.value = 'admin';" required="required" />
          <label for="pwd">Password:</label>
          <input type="password" name="pwd" id="pwd" required="required" />
          <div class="center"><button type="submit" name="submit">Login</button></div>
        </fieldset>
      </form>

      <footer>
        <p>By <strong><a href="http://ph-7.github.com">pH7</a></strong> &copy; 2012.</p>
      </footer>

    </div>
  </body>
</html>
```

Nous allons maintenant récupérer les informations envoyer par le formulaire de connexion et les stocker

dans un fichier de logs ou les envoyer par email, ainsi vous serez immédiatement averti quand quelqu'un essayera de se connecter.

Vous pouvez également ajouter un code de suivi analytique pour avoir de plus amples informations (provenance du pays, ville, temps resté sur la page, etc.) sur les personnes qui tentent de se connecter à la fausse partie d'administration de votre site.

Enfin, j'ai placé la fonction sleep() qui permet d'endormir le script pendant quelques secondes afin de se sécuriser contre les attaques à force brute causé par d'éventuels bots, mais également pour agacer le pirate.

Et maintenant voici la classe "Sniffer" (listing 2) qui va permettre de récupérer la plupart des actions de l'attaquant, ainsi que l'éventuelle possibilité de bannir automatiquement les personnes qui tentent de se connecter dans la partie administrative (car souvent ces visiteurs n'apportent rien de positif à votre site).

Dans cette classe, nous créons cinq constantes.
Les deux premières concernent le rapport d'email, la suivante est l'activation du bannissement automatique d'adresse IP, enfin les deux dernières sont les chemins du fichier de log et du fichier de la liste des adresses IP bannis du site.
Puis nous créons le constructeur qui initialise les variables de login entrées via l'interface d'administration.
Après, nous initialisons et écrivons le message de log avec les méthodes setLogMsg() et writeFile(), enfin nous envoyons un email avec à l'aide de la méthode sendMessage() si la constante EMAIL_REPORT est vraie et nous bannissons l'adresse IP du spammeur à l'aide de la méthode blockIp() si la constante AUTO_IP_BLOCK est vraie.

Listing 2. La classe qui permet d'analyser et de stocker les comportements du pirate

```
<?php
class Sniff
{

    /**
     * Your informations here.
     */

        // TRUE = enable sending email to each someone tries to connect to admin login.
    const EMAIL_REPORT = true;

        // Email address where reports will be sent emails if Sniff::EMAIL_REPORT is TRUE.
    const EMAIL_ADDRESS = 'you@your-domain.com';
```

```
    /**
     * Settings of application.
     */

    // TRUE = Automatically banned all those who attempt to log into the admin.
    const AUTO_IP_BLOCK = false;

    // Path where will be stored log files.
    const LOG_PATH = '../../_data/logs/attackers/';

    // Path from the list of IP addresses that are banned from the site.
    const BAN_IP_FULL_PATH = '../../_data/bans/ip.txt';

    private $_sUsername, $_sPassword, $_sIp, $_sContents;

    /**
     * Constructor.
     *
     * @param string $sUsername The Username of the Admin Login.
     */
    public function __construct($sUsername, $sPassword)
    {   // Initializes login variables.
        $this->_sUsername = $sUsername;
        $this->_sPassword = $sPassword;

        // Creates the log message and adds it to the list of logs.
        $this->setLogMsg()->writeFile();

        // Sends the email report.
        if(self::EMAIL_REPORT) $this->sendMessage();

        // Blocks IP address.
        if(self::AUTO_IP_BLOCK) $this->blockIp();
```

```
    }

    /**
     * Check if the IP address is banned.
     *
     * @return boolean Returns true if the ip is banned, otherwise returns false.
     */
    public static function isIpBlock()
    {
        if(is_file(self::BAN_IP_FULL_PATH))
        {
            $aIpBans = file(self::BAN_IP_FULL_PATH);

            foreach($aIpBans as $sIp)
            {
                $sIp = trim($sIp);
                if(0 == strcmp(self::getIp(), $sIp)) return true;
            }
        }

        return false;
    }

    /**
     * Return the IP address of a client.
     *
     * @return string
     */
    public static function getIp()
    {
        if (!empty($_SERVER['HTTP_X_FORWARDED_FOR']))
        {
            $sIp = $_SERVER['HTTP_X_FORWARDED_FOR'];
        }
```

```
        elseif (!empty($_SERVER['HTTP_CLIENT_IP']))
        {
            $sIp = $_SERVER['HTTP_CLIENT_IP'];
        }
        else
        {
            $sIp = $_SERVER['REMOTE_ADDR'];
        }

        return preg_match('/^[a-z0-9:.]{7,}$/', $sIp) ? $sIp : '0.0.0.0';
    }

    /**
     * Build the log message.
     *
     * @return object this
     */
    protected function setLogMsg()
    {
        $sReferer = (!empty($_SERVER['HTTP_REFERER'])) ? $_SERVER['HTTP_REFERER'] : 'NO HTTP REFERER';
        $sAgent = (!empty($_SERVER['HTTP_USER_AGENT'])) ? $_SERVER['HTTP_USER_AGENT'] : 'NO USER AGENT';
        $sQuery = (!empty($_SERVER['QUERY_STRING'])) ? $_SERVER['QUERY_STRING'] : 'NO QUERY STRING';

        $this->_sIp = self::getIp();

        $this->_sContents =
        'Date: ' . date('Y/m/d') . "\n" .
        'IP: ' . $this->_sIp . "\n" .
        'QUERY: ' . $sQuery . "\n" .
        'Agent: ' . $sAgent . "\n" .
        'Referer: ' . $sReferer . "\n" .
        'LOGIN - Username: ' . $this->_sUsername . ' - Password: ' . $this->_sPassword . "\n\n\n";
```

```
        return $this;
    }

    /**
     * Write a log file with the hacher informations.
     *
     * @return object  this
     */
    protected function writeFile()
    {
        $sFileName = $this->_sIp . '.log';
        $sFullPath =  self::LOG_PATH . $sFileName;
        file_put_contents($sFullPath, $this->_sContents, FILE_APPEND);

        return $this;
    }

    /**
     * Blocking IP address.
     *
     * @return object this
     */
    protected function blockIp()
    {
        $sContent = $this->_sIp . "\n";
        file_put_contents(self::BAN_IP_FULL_PATH, $sContent, FILE_APPEND);

        return $this;
    }

    /**
     * Send an email.
     *
```

```
    * @return boolean Returns TRUE if the mail was successfully accepted for delivery, FALSE otherwise.
    */
    protected function sendMessage()
    {
        $sHeaders = "From: \"{$_SERVER['HTTP_HOST']}\" <{$_SERVER['SERVER_ADMIN']}>\r\n";
        return mail(self::EMAIL_ADDRESS, 'Reporting of the Fake Admin Honeypot', $this->_sContents, $sHeaders);
    }


}
```

Pour plus d'informations sur cette classe, consultez l'URL du code source de cet article dans la section des liens.

Utiliser cette technique dans un CMS

Cette technique est parfaitement utilisable dans un CMS (même conseillé), en effet, les CMS sont généralement beaucoup plus à risque de ce genre d'attaque, car la personne qui veut hacker votre site va d'abord examiner attentivement les codes source du script en question afin de connaître son architecture et la répartition des fichiers ainsi que l'éventuelle morceau de code qui comporte une faille de sécurité qui pourrait être exploitable.

Dans cet exemple, nous allons utiliser le célèbre CMS Joomla 2.5, mais il devrait être facile d'adapter le code avec un autre CMS.
Pour Joomla, nous allons utiliser une technique légèrement différente, car Joomla n'autorise pas que l'on renomme le dossier "administrator", donc nous allons laisser tout le contenu dans ce dossier.

Par contre, nous allons renommer le fichier "index.php" de ce dossier par "_joomla_index.inc.php" et ajouter ce code defined('IS_INDEX') or die; tout au début du fichier, juste après la balise <?php, cette ligne permet d'interdire qu'une personne y accède sans être passé par le fichier de routage d'accès que nous allons créer par la suite.

Ainsi, uniquement les personnes qui connaissent le dossier top secret d'administration auront une variable de session qui permettra d'avoir accès à la page de connexion originale, les autres auront une fausse page de connexion.

Listing 3. Fichier qui servira de routeur pour gérer la bonne page de connexion

```
<?php
define('IS_INDEX', 1);
```

```
session_start();
if (!empty($_SESSION['joomla_admin_sess']) && $_SESSION['joomla_admin_sess'] ==
'VOTRE_MOT_SECRET')
{
    require '_joomla_index.inc.php'; // OK, the URL from where the person is the URL custom
administration.
}
else
{
    require '_honeypot_index.inc.php'; // The fake admin interface.
}
?>
```

Toujours dans le même dossier, créer un fichier “_honeypot_index.inc.php” avec le contenu du listing 4.

Il s’agit de la fausse page de connexion incluant la classe d'écoute, etc.

Listing 4. Fausse page de connexion

```
<?php
/**
 * Architecture and design reproduced of Joomla version 2.5.6
 */
defined('IS_INDEX') or die; // Security check

session_start();

/**
 * Generate a random token name field of the login form.
 */
if(empty($_SESSION['login_token']))
    $_SESSION['login_token'] = md5(uniqid(mt_rand(), true));

/**
 * Gets the root URL.
```

```
 * It is useful to get the URL to reproduce exactly the same source code as the original Joomla
administration.
 *
 * @return string
 */
function get_url()
{
    // URL Association for SSL and Protocol Compatibility
    $sHttp = (!empty($_SERVER['HTTPS']) && strtolower($_SERVER['HTTPS'] == 'on')) ? 'https://' :
'http://';


    return $sHttp . $_SERVER['HTTP_HOST'] .
dirname(dirname(htmlspecialchars($_SERVER['PHP_SELF'])));
}


/**
 * Gets the root relative URL.
 * It is useful to get the URL relative to reproduce exactly the same source code as the original Joomla
administration.
 *
 * @return string
 */
function get_relative_url()
{
    return dirname(dirname(htmlspecialchars($_SERVER['PHP_SELF'])));
}


/**
 * This page is a fake admin login page of site.
 * Your real login page it just another URL (e.g. http://your-site.com/_my-secret-admin-page/).
 */
require '../../_inc/Sniff.class.php';


$bErr = false; // Default value
```

```
if(isset($_POST['username'], $_POST['passwd']))
{
    sleep(6); // Security against brute-force attack and this will irritate the hacker...
    $bErr = true;
    new Sniff($_POST['username'], $_POST['passwd']);
}

/**
 * Check if the IP address is banned.
 */
if(Sniff::isIpBlock())
{
    header('Location: ../'); // Go to index.
    exit;
}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb" lang="en-gb" dir="ltr" >
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="generator" content="Joomla! - Open Source Content Management" />
  <title>Honeypot Joomla Example - Administration</title>
  <link href="<?php echo get_relative_url() ?>/administrator/templates/bluestork/favicon.ico"
rel="shortcut icon" type="image/vnd.microsoft.icon" />
  <link rel="stylesheet" href="templates/system/css/system.css" type="text/css" />
  <link rel="stylesheet" href="templates/bluestork/css/template.css" type="text/css" />
  <style type="text/css">
html { display:none }
  </style>
  <script src="<?php echo get_relative_url() ?>/media/system/js/mootools-core.js"
type="text/javascript"></script>
  <script src="<?php echo get_relative_url() ?>/media/system/js/core.js" type="text/javascript"></script>
  <script type="text/javascript">
```

```
function keepAlive() {	var myAjax = new Request({method: "get", url: "index.php"}).send();}
window.addEvent("domready", function(){ keepAlive.periodical(840000); });
window.addEvent('domready', function () {if (top == self) {document.documentElement.style.display =
'block'; } else {top.location = self.location; }});
  </script>



<!--[if IE 7]>
<link href="templates/bluestork/css/ie7.css" rel="stylesheet" type="text/css" />
<![endif]-->


<script type="text/javascript">
   window.addEvent('domready', function () {
      document.getElementById('form-login').username.select();
      document.getElementById('form-login').username.focus();
   });
</script>
</head>
<body>
   <div id="border-top" class="h_blue">
      <span class="title"><a href="index.php">Administration</a></span>
   </div>
   <div id="content-box">
         <div id="element-box" class="login">
            <div class="m wbg">
               <h1>Joomla! Administration Login</h1>


<div id="system-message-container">
</div>
                     <div id="section-box">
         <div class="m">
            <form action="<?php echo get_relative_url() ?>/administrator/index.php" method="post"
id="form-login">
   <fieldset class="loginform">
```

```
		<label id="mod-login-username-lbl" for="mod-login-username">User Name</label>
		<input name="username" id="mod-login-username" type="text" class="inputbox" size="15" />

		<label id="mod-login-password-lbl" for="mod-login-password">Password</label>
		<input name="passwd" id="mod-login-password" type="password" class="inputbox" size="15" />

		<label id="mod-login-language-lbl" for="lang">Language</label>
		<select id="lang" name="lang"  class="inputbox">
	<option value="" selected="selected">Default</option>
	<option value="en-GB">English (United Kingdom)</option>
</select>

		<div class="button-holder">
			<div class="button1">
				<div class="next">
					<a href="#" onclick="document.getElementById('form-login').submit();">
						Log in</a>
				</div>
			</div>
		</div>

	<div class="clr"></div>
	<input type="submit" class="hidebtn" value="Log in" />
	<input type="hidden" name="option" value="com_login" />
	<input type="hidden" name="task" value="login" />
	<input type="hidden" name="return" value="aW5kZXgucGhw" />
	<input type="hidden" name="<?php echo $_SESSION['login_token'] ?>" value="1" />	</fieldset>
</form>
		<div class="clr"></div>
	</div>
	</div>

			<p>Use a valid username and password to gain access to the administrator backend.</p>
```

```
            <p><a href="<?php echo get_url() ?>">Go to site home page.</a></p>
            <div id="lock"></div>
        </div>
    </div>
    <noscript>
        Warning! JavaScript must be enabled for proper operation of the Administrator backend.
</noscript>
  </div>
  <div id="footer">
    <p class="copyright">
      <a href="http://www.joomla.org">Joomla!&#174;</a> is free software released under the <a href="http://www.gnu.org/licenses/gpl-2.0.html">GNU General Public License</a>.    </p>
  </div>
</body>
</html>
```

Enfin, créez le vrai dossier d'administration portant le nom de votre choix, mais difficile à deviner et créez-y un fichier index.php contenant le code du listing 5.

Listing 5. Fichier qui initialise la session pour la page de connexion Joomla

```
<?php
define('IS_INDEX', 1);

session_start();
$_SESSION['joomla_admin_sess'] = 'VOTRE_MOT_SECRET'; // Joomla Login is OK
header('Location: ../administrator/');
?>
```

Gardez à l'esprit qu'en général il suffit de renommer le réel dossier d'administration par un nom difficile à deviner et de créer une fausse partie administrative portant le nom de l'ancien (ex : admin/) puis de modifier une variable ou une constante qui est le chemin de cette partie administrative (généralement elle se trouve dans le fichier de configuration de votre site).

L'exemple précédent est différent, car Joomla ne permet pas que le dossier "administrator" soit renommé sous un autre nom.

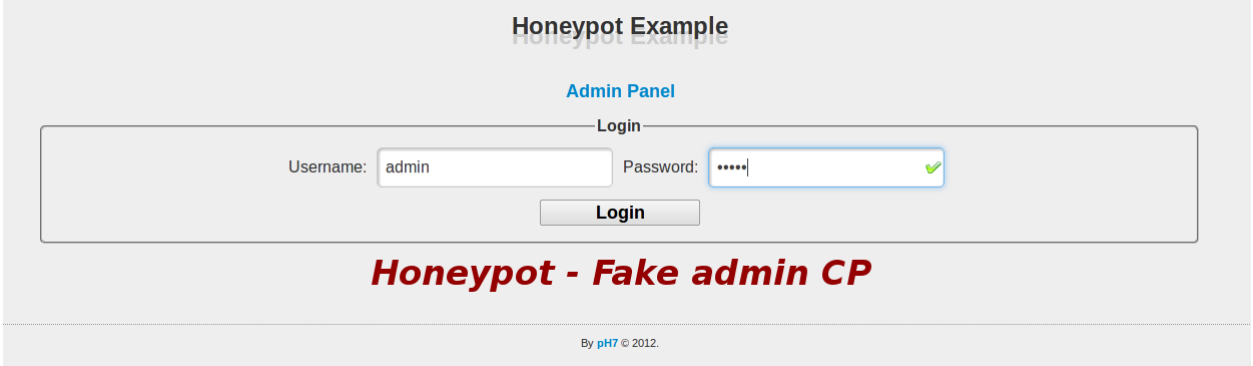

Figure 1. Exemple de la fausse page de connexion.

Les différentes techniques de pots de miel

Dans ce tutoriel, nous avons utilisé le pot de miel pour faire croire aux pirates informatiques ou aux spammeurs (robots, etc.) qu'il s'agissait de la page de connexion de la partie administrative, mais en réalité il s'agit d'un piège.

Mais cette technique de pots de miel peut s'utiliser pour d'autres choses.
Par exemple, pour créer un faux captcha qui sera invisible à l'œil nu pour les internautes, mais visible pour les robots de spam. Il serait également intéressant d'examiner le comportement et la technique utilisée par ce bot pour, par exemple améliorer notre vrai captcha pour qu'il soit plus efficace.

Nous pouvons également bannir ces robots par son adresse IP de la même manière que nous l'avons fait dans la classe « Sniff ».
Un autre exemple est une adresse email de spam (aussi connu sous le nom d'e-mail piège ou e-mail trap) qui est utilisé que pour recevoir du spam. Cette technique est souvent considérée comme un pot de miel à spam.
Un projet sous le nom de **Project Honey Pot** à également été créé pour lutter contre le spam et ainsi collecter toutes les adresses email de spam ou pour d'autres fins similaires, comme le courrier en vrac et la fraude par courriel.
Ces exemples sont loin d'être les seuls, car la technique du pot de miel est très vaste et n'est pas uniquement destiné pour le domaine du Web.
Pour de plus amples informations, j'ai ajouté un article Wikipedia dans la section des liens.

Catégories de pot de miel

Avant de terminer cet article, je vous propose de découvrir en grandes lignes, les deux différents types de pots de miel.

Pots de miel à faible interaction:

Ce sont les plus simples à mettre en place et les plus sécurisés de la famille des pots de miel.

Leurs but est de récolter un maximum d'informations tout en offrant un minimum de privilèges aux attaquants afin de limiter au maximum les risques de sécurité.

Pots de miel à forte interaction:
Ils peuvent être considérés comme le côté extrême, car ils reposent sur le principe de l'accès à de véritable interactions et fonctionnalités du service concerné.

Les risques sont beaucoup plus importants que pour les pots de miel à faible interaction.

Il apparaît donc nécessaire de sécuriser au maximum l'architecture du service afin que l'attaquant ne puisse s'en prendre et accéder à d'autres fonctionnalités et services qui ne lui sont pas destinés.

Conclusion

– Cette technique permet de mieux connaître les intentions et les tentatives d'hacking sur votre site.

Par exemple, maintenant vous allez recevoir un email ou un message dans les fichiers de logs à chaque fois qu'une personne tente de se connecter, vous pouvez également voir si la personne utilise le même nom d'utilisateur ou mot de passe, si tels est le cas, cela veut dire que quelqu'un d'autre connait votre passe de connexion, vous devez donc vous prémunir en changeant votre mot de passe.

– Elle permet également de faire perdre du temps précieux à la personne qui essaye d'hacker votre site et ainsi diminuer le risque de tentative d'hacking ailleurs (sur votre site ou sur un autre).

– Vous pouvez bloquer les adresses IP des personnes mal intentionnés afin de diminuer encore davantage les éventuelles risques de piratage sur votre site, même si cette pratique reste discutable, en effet vous pouvez facilement modifier votre adresse IP via des serveurs proxy, de plus beaucoup d'opérateur Internet fournissent une IP dynamique (adresse qui change périodiquement).

– Enfin, les bots qui essayeront d'utiliser toutes les combinaisons possibles pour accéder à l'administration du site, serons également bannit si vous avez activé la protection de bannissement d'IP automatique.

Sur Internet

- *http://github.com/pH-7/Honeypots* – La totalité du code source abordée dans cet article.
- *http://github.com/pH-7/fake-admin-honeypot-V1.1* – Module d'une fausse partie administrative dont vous pouvez vous inspirer du code source.

- *http://projecthoneypot.org* – Project Honey Pot est un réseau de récolte d'adresse IP de spam afin de lutter contre le courrier non désiré.

http://fr.wikipedia.org/wiki/Honeypot – Explication brève des principes de fonctionnement du pot de miel.

- http://www.tracking-hackers.com/papers/honeypots.html – *Article intéressant sur la définition et l'explication du pot de miel (en anglais).*

À propos de l'auteur – Pierre-Henry Soria

Pierre-Henry Soria travaille dans le développement Web depuis plusieurs années,
il se spécialise sur l'accessibilité, le référencement ainsi que la sécurité informatique. Il est l'auteur entre autre du site d'actualité francophone ***01Script*** ainsi que du CMS social open source ***pH7CMS*** et de ***IDzUp*** qui fournit des portails communautaires de hautes qualités pour les entreprises.
Il travaille également sur les parseurs et analyseurs lexicaux et syntaxiques en langages de bas niveaux.

Vous pouvez le contacter par e-mail à : *phs@hizup.net*