УДК 004.49

И.В. Кобзев, К.Э. Петров, С.В. Калякин

Харьковский национальный университет внутренних дел, Харьков

ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ САЙТА, ПОСТРОЕННОГО НА СИСТЕМЕ УПРАВЛЕНИЯ КОНТЕНТОМ WORDPRESS

Безопасность является одной из базовых концепций веб-технологий в настоящее время. WordPress является одним из наиболее оптимальных решений при выборе платформы для разработки сайтов, В статье рассматриваются вопросы защиты веб-сайта, построенного на системе управления контентом WordPress. Рассмотрены методы несанкционированного доступа и предложены варианты защиты подобных сайтов-блогов.

Ключевые слова: *система управления контентом, Wordpress, безопасность, защита, блог, запрос.*

Введение

Wordpress является популярной системой управления контентом для блогов, который используется на множестве сайтов. Открытая блог-платформа WordPress преодолела важный рубеж: она используется уже на более чем 70 миллионах сайтов. Около половины из этих сайтов размещены на WordPress.com.

Кроме того, более 358 миллионов людей просматривают более 2,5 миллиарда страниц на WordPress.com каждый месяц, а в среднем за день пользователи WordPress.com создают около 500 тысяч новых записей и 400 тысяч новых комментариев, согласно официальной статистике WordPress [1]. На данный момент актуальной версией CMS Wordpress является версия 3.3.1 [2].

Постановка задачи. Системы управления контентом (CMS), как и многие другие виды программного обеспечения достаточно уязвимы. В отличие от CMS собственной разработки, если злоумышленники или хакеры находят уязвимые места в одной конкретной тиражируемой CMS (не имеет значения платной или бесплатной) – возникает угроза взлома всех подобных CMS данного производителя. При этом, чем более распространённой является система и чем чаще она применяется на популярных сайтах, тем больше денег и усилий злоумышленники инвестируют в поиск её проблемных мест.

Кроме того, большинство современных CMS состоят из множества модулей, и большинство уязвимых мест связаны с плагинами, которые обычно написаны и протестированы на безопасность хуже, чем основной код системы.

Используя различные недостатки CMS, злоумышленники стараются извлечь для себя выгоду за счёт чужих сайтов и посетителей. Например, на страницах сайта может быть размещён код, который заражает компьютеры посетителей вредоносными программами. Кроме того, на сайте со взломанной CMS может, без ведома веб-мастера, публиковаться контент сомнительного содержания или осуществляться автоматическое перенаправление пользователей на другие ресурсы с таким контентом. В результате страдает репутация сайта, и количество его посетителей может существенно уменьшиться.

По данным Яндекса, доля CMS WordPress на популярных русскоязычных сайтах достигает 64%. Кроме этого, Яндекс предоставил подробную статистику взломов из которой видно, что лидерами по частоте взломов являются сайты на CMS DLE - 50% выборки взломанных сайтов. Сайты, работающие на CMS WordPress и Joomla!, занимают соответственно по 19% выборки каждый. [3].

Основным недостатком WordPress, как и любой другой открытой бесплатной платформы для блогов, является ее доступность для злоумышленников. В статье описываются некоторые способы несанкционированного доступа к сайтам, которые используют CMS WordPress и проблемы обеспечения безопасности таких сайтов.

Проблемы обеспечения безопасности блога и пути их решения

Рассмотрим более подробно способы защиты от несанкционированного доступа к системе, базирующейся на использовании CMS WordPress.

Попытка несанкционированного доступа через URL. Через URL-базированные внедрения, хакеры пытаются найти «слабые» места web-сайта, используя запросы, которые при нормальном администрировании хостинга должны выдавать ошибку, но выполняются. Например, в процессе выполнения запроса http://mybllog.com/try/to/exploit/%2F/config можно получить доступ к файлам конфигурации блога.

Одним из способов защиты от такого рода запросов является использование файла .htaccess в качестве Firewall. Таким образом имеется возможность задать набор правил, которые будут автоматически блокировать запросы содержащие строковые переменные в адресной строке.

Например, использование скобок в HTTP-запросе говорит о том, что кто-то пытается обнаружить «дыру» в системе. В данном случае рекомендуется генерировать страницу «403 Forbidden - доступ запрещен» следующим образом. В файл .htaccess необходимо ввести строку RedirectMatch 403 [].

Таким образом будут заблокированы любые запросы, содержащие квадратные скобки.

Наиболее полное описание защиты WordPress через файл .htaccess можно найти у Jeff Starr, который использует 5G Firewall [4].

```
# 5G:[REQUEST STRINGS]
<IfModule mod_alias.c>
RedirectMatch 403 (https?|ftp|php)\://
RedirectMatch 403 /(cgi|https?|ima|ucp)/
RedirectMatch 403 (\=\\\'|\=\\%27|/\\\'/?|\)\.css\()$
RedirectMatch 403
(\,|//|\)\+|/\,/|\{0\}|\(/\(|\.\.\.|\+\+\+|\|)
RedirectMatch 403
\.(cgi|asp|aspx|cfg|dll|exe|jsp|mdb|sql|ini|rar)$
RedirectMatch 403
/(contac|fpw|install|pingserver|register)\.php
RedirectMatch 403
(base64|crossdomain|localhost|wwwroot)
RedirectMatch 403
(eval\(|\_vti\_|\(null\)|echo.*kae)
RedirectMatch 403 \.well\-known/host\-meta
RedirectMatch 403 /function\.array\-rand
RedirectMatch 403 \)\;\$\(this\)\.html\(
RedirectMatch 403 proc/self/environ
RedirectMatch 403 msnbot\.htm\)\.\_
RedirectMatch 403 /ref\.outcontrol
RedirectMatch 403 com\_cropimage
RedirectMatch 403 indonesia\.htm
RedirectMatch 403 \{\$itemURL\}
RedirectMatch 403 function\(\)
RedirectMatch 403 labels\.rdf
```

Firewall имеет модульную структуру, что дает возможность удалить любую строку без нарушения функциональности системы

Еще одним способом защиты системы является ограничение доступа к директориям сервера. На многих серверах существует возможность просмотреть содержимое каталогов используя простой HTTP-запрос примерно такого содержания: http://myblog.com/wp-content/uploads/2012/02/. С помощью данного запроса к сайту, использующему WordPress, можно получить содержание каталога со всеми загрузками, осуществленными с февраля 2012 года. Закрыть доступ в конкретный каталог при отсутствии индексного файла можно следующим образом.

За листинг файлов отвечает директива Indexes (показывать посетителю список файлов, если в выбранном каталоге нет файла index.html или его аналога). Необходимо создать в данном каталоге файл .htaccess, который содержит одну строку: Options – Indexes.

После этого при попытке доступа к данному каталогу будет появляться сообщение «403 Forbidden - доступ запрещен».

Если же понадобиться разрешить просматривать список файлов, но чтобы при этом часть файлов определенного формата не отображалась, то необходимо использовать директиву IndexIgnore directive. Например:

IndexIgnore *

или

indexIgnore *.jpg *.gif *.png

В первом случае, даже при полном доступе к каталогу, не будет видно ни одного файла. Во втором – будут показаны все файлы за исключением трех видов графических: *.jpg *.gif *.png.

Запретить доступ к отдельным файлам или группам файлов можно используя директиву deny from all.

```
<Files config.php>
Deny from all
</Files>
```

В данном случае для всех пользователей без прав администратора будет закрыт доступ к файлу config.php

Помимо указанного выше можно ограничить доступ к сайту с отдельных или диапазонов IP-адресов, поставить пароли на конкретные каталоги, использовать перенаправление (redirect) только при запросе определенных страниц или только посетителей с определенным IP-адресом, запретить скачивание файлов и т.п.

Защита от внедрения в административную часть.

Следующая разновидность проблем исходит от пользователей, которые попадают в административную часть системы. Используя HTTP-запрос http://mybllog.com/admin/scripts/delete_user.php?user_id=11 можно получить доступ к списку зарегистрированных пользователей системы и удалить их. Таким образом, если доступ к директории http://myblog.com/admin/scripts/ не закрыт и отсутствует проверка прав пользователей, то зарегистрированный пользователь с ID равным 11 будет удален из списка пользователей системы.

В данном случае надо выяснить две вещи. Имеет ли податель запроса права на выполнение подобных действий? Понимает ли он, последствия своих действий?

WordPress имеет функциональные возможности проверить оба условия перед тем, как подобного вида запрос будет выполнен.

Например, только администратор системы имеет права на выполнение функций редактирования контента сайта. То есть если происходит попытка редактирования/изменения контента необходимо удостовериться, что пользователь имеет права на такие действия.

Предположим, что HTTP-запрос на удаление комментария имеет следующий вид: http://mybllog.com/admin/scripts/delete_comment.php?comment_id=11

Скрипт WordPress проверяет залогинен ли в данный момент пользователь с правами администратора. Допустим «хакер» выкладывает подобную гиперссылку на свой сайт: <ahref="http:// mybllog.com/admin/scripts/delete_comment.php?comment_id=11">Вы выиграли 1000000!.

Перейдя по данной ссылке простой пользователь ничего не увидит, но если вдруг это сделает администратор mybllog.com, то комментарий будет удален.

Проверка авторизации в WordPress. WordPress имеет мощную систему авторизации известную как «Роли и возможности – Roles and capabilities». Возможности – это основа всей системы, роли – способ сгруппировать возможности вместе [5].

Пользователь с правами SuperAdmin имеет права на выполнение любых действий в системе. Administrator – тот, кто имеет доступ к панели администрирования. Editor, в отличие от Author, имеет право редактировать любые, а не только свои записи. Contributor может писать и редактировать свои статьи, но не публиковать их. Subscriber может только редактировать свой профиль.

WordPress обеспечивает легитимность выполнения действий своими авторизованными пользователями следующим образом.

```
if(current_user_can("delete_users")) {
wp_delete_user(11);
}
else {
die("Невозможно выполнить это действие…");
}
```

Функция current_user_can() использует один аргумент для выполнения действий. Имеется возможность с правами «editor» удалять пользователей следующим образом:

```
if(current_user_can("editor")) {
wp_delete_user(11);
}
else {
die("Вы можете удалить");
}
```

Следующие две функции дают возможность проверять права пользователей.

```
if(user_can(11, "управление ссылками")) {
echo "Пользователю 11 позволено редактировать ссылки";
}
else {
echo "Пользователь 11 не может редактировать ссылки ";
}
if(author_can(1000, "update_themes")) {
echo "Автору статьи #1000 разрешено изменить тему";
}
else {
echo " Автору статьи #1000 не разрешено изменить тему ";
}
```

Функция user_can() проверяет роли и возможности пользователя. Первый аргумент – ID-пользователя, а второй – проверяемые роли и возможности.

Проверка правильности протоколов WordPress. Одним из способов данного вида проверки является анализ $_SERVER['HTTP_REFERER']. Это глобальная переменная показывает с какой страницы был совершён переход на данную. Переменная показывает полный путь к странице, с которой был осуществлен переход. Если «хакер» каким-либо способом получит доступ к переменной, в которой хранится адрес сайта, то он сможет получить полный доступ к базе данных системы.

В WordPress защита основывается на использовании случайного идентификатора, который генерируется для каждой операции и необходим для ее выполнения. Данный идентификатор также называют маркером доступа – nonce (маркер).

Такой маркер в WordPress может генерироваться в двух местах: формах и гиперссылках.

Маркер в форме используется следующим образом.

```
<form id="form1" method="post" action="myscript.php">
<h1>Введите слово</h1>
<input type='text' name='n1'>
<?php wp_nonce_field( 'wp_test') ?>
</form>
```

Данный фрагмент сгенерирует скрытое поле ввода содержащее маркер, который будет отправлен вместе с данными формы. wp_nonce_field() вызывает другую функцию – wp_original_referer_field(), которая вставляет поле с именем _wp_original_http_referer. Это поле хранит адрес страницы, с которой пользователь перешел на страницу с формой.

В результате скрытое поле будет иметь следующий вид: <input type="hidden" id="_wpnonce" name="_wpnonce" value="d6641w9964">.

Необходимо проверять наличие и значение маркера перед тем как разрешать пользователям выполнять какие либо действия. Один из способов заключается в следующем:

```
if
(!wp_verify_nonce($_POST['_wpnonce'],"wp_test') ) {
die('не верно.');
```

```
}
else {
awesome_word_inserter($_POST["n1"]);
}
```

Здесь используется функция wp_verify_nonce(), которая необходима для проверки корректности маркера. Эта функция имеет два параметра: первый – значение поля маркера, второй – действие, которое производится.

В некоторых случаях можно использовать маркеры в ссылках. Например,

http://blog.com/admin/scripts/deletethatthing.php?thing_id=66

Для генерации маркера ссылки можно использовать следующий подход:

```
$base_url =
"http://blog.com/admin/scripts/deletethatthing.php?thing_id=66";
$nonce_url = wp_nonce_url( $base_url, "thing-deleter_nonce");
echo "<a href='".$nonce_url."'>Удалить</a>";
```

В результате имеем: http://blog.com/admin/scripts/deletethatthing.php?thing_id=66&_wpnonce= d6641w9964.

Проверяем маркер аналогично первому способу:

```
if
(!wp_verify_nonce($_GET['_wpnonce'],'thingdeleter_nonce') ) {
die('Oops, your nonce didn\'t verify. So there.');
}
else {
delete_that_thing($_GET["thing_id"]);
}
```

Ссылка, удаляющая комментарии должна размещаться на той же странице:

```
$nonce_url =
wp_nonce_url("http://blog.com/scripts/delete_comment.php?comment_id=1000", "delete_comment_nonce");
echo "<a href='".$nonce_url."'>удалить комментарий</a>";
```

Теперь приведем непосредственно скрипт:

```
if
(wp_verify_nonce($_GET['_wpnonce'],'delete_comment_nonce') AND current_user_can("edit_comment")) {
wp_delete_comment($_GET["comment_id"]);
}
else {
die(Нет прав на удаление.');
}
```

Безопасность данных. Дополнительный уровень безопасности необходим для обеспечения целостности особо важных данных системы при удалении чего-либо из базы данных системы. Особое внимание следует уделить защите от внедрения SQL-кода (так называемый SQL-injection). SQL-injection – один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода. Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность «хакеру» выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на сервере, который подвергается атаке [6].

Например, администратор, введя в поле запроса имя пользователя получает его email.

```
SELECT user_email FROM wp_users WHERE user_login = 'guest'
```

«Хакер» внедряет SQL-код вида ' OR 1=1 'в форму и получает запрос следующего вида:

```
SELECT user_email FROM wp_users WHERE user_login = '' OR 1=1 ''
```

После выполнения такого запроса «хакер» получает все адреса электронной почты, если не было введено имя пользователя.

Для недопущения таких внедрений в первую очередь необходимо проверять правильность введенных данных.

При использовании WordPress наиболее предпочтительным методом защиты является использование класса $wpdb. Используя класс wpdb можно производить всевозможные операции с базой данных WordPress, предотвращая SQL-injection, зная принципы управления этим классом [7].

```
// Выполнение запроса
$wpdb->query("DELETE FROM wp_users WHERE user_id = 11");
// Получение первого столбца данных
$posts = $wpdb->get_col("SELECT post_title FROM wp_posts WHERE post_status = 'опубликовано' ORDER BY comment_count DESC LIMIT 0,10");
// Получение строки данных
$post = $wpdb->get_row("SELECT * FROM wp_posts WHERE ID = 1000");
// Получение набора строк и столбцов
$posts = $wpdb->get_results("SELECT ID, post_title, post_date FROM wp_posts WHERE post_type = 'опубликовано' ORDER BY post_date DESC LIMIT 0, 12 ");
// Получение уникального значения
$author_id = $wpdb->get_var("SELECT post_author FROM wp_posts WHERE ID = 222");
// Добавление записи
$wpdb->insert("wp_postmeta", array("post_id" => 4444, "meta_key" => "favorite_count", "meta_value" => 322 ), array("&percnt;d", "&percnt;s", "&percnt;d"));
// Обновление записи
$wpdb->update("wp_postmeta", array("meta_value" => 323), array("meta_key" => "favorite_count", "post_id" => 4444), array("&percnt;d"), array("&percnt;s", "&percnt;d"));
```

Методы insert() и update() являются вспомогательными.

Более простым способом является использование метода escape():

```
$data = $wpdb->escape($_POST[about_me]);
$wpdb->query("UPDATE wp_usermeta SET meta_value = '$data' WHERE meta_key = 'description' AND user_id = 100 ");
```

Помимо вышесказанного, сохранность данных можно обеспечить, защитив их с помощью SSL. SSL представляет собой протокол шифрования, который используется в различных сетях, таких как Интернет. Использование SSL связано лишь с одной проблемой: не все компании, предлагающие услуги хостинга, поддерживают данный вид шифрования. Именно поэтому включение SSL может не дать никаких результатов.

Также можно использовать различные полезные плагины, которые дают возможность надежно защитить блог от «хакерских» атак. Приведем несколько примеров.

BulletProof Security - позволяет защитить блог от кросс-серверных скриптов, SQL-injection, base64, CSRF, а также надежно сохранить жизненно важные файлы сервера, такие как .htaccess и php.ini.

Login Lock Down – записывает IP-адреса при неудачных попытках входа в систему. Если число попыток входа было превышено, IP-адрес блокируется на заданный промежуток времени. Это прекрасное средство для предотвращения брутфорса.

Block Bad Queries (BBQ) – плагин, который препятствует выполнению злонамеренных запросов eval и base64. Он обязательно необходим в случае, если не установлена защита BulletProof Security [8].

Login LockDown — ставит ограничение на количество попыток входа в панель администратора Допустим если злоумышленник ввел подряд 3 раза неправильно логин и пароль, то до следующей попытки ему нужно подождать 30 минут. Время и количество попыток можете устанавливать сами. Плагин прост в установке.

WordPress File Monitor — плагин отслеживает все изменения блога, связанные с добавлением/удалением/изменением файлов.

WordPress Database Backup — отправляет копию вашей базы данных к вам на почту ежедневно (интервал времени можно задать самому).

Выводы

Сфера информационной безопасности – актуальнейший вопрос современности. Защита сайта от «хакеров» становится глобальной проблемой, над разрешением которой работают специалисты всего мира. Бизнес, построенный в агрессивной среде Интернет, уязвим и подвержен нападениям со стороны конкурентов и недоброжелателей.

Единого инструмента для устранения всех угроз несанкционированного доступа к сайтам построенным на основе систем управления контентом – просто не существует. Обеспечение защиты сайта – это комплексная задача.

Предложенные в статье подходы к решению этой нетривиальной задачи помогут администраторам и пользователям сайтов организовать защиту контента от несанкционированного доступа.

Список литературы

1. Stats wordpress.com. [Електронний ресурс]. Режим доступу URL: http://en.wordpress.com/stats/.

2. Русский wordpress. [Електронний ресурс]. Режим доступу URL: http://ru.wordpress.org/.

3. Безопасный поиск Яндекса. [Електронний ресурс]. Режим доступу URL: http://safesearch.ya.ru/.

4. 5G Firewall Beta. Security by Jeff Starr. [Електронний ресурс]. Режим доступу URL.

5. Roles and capabilities.. [Електронний ресурс]. Режим доступу URL: http://codex.wordpress.org/Roles_and_Capabilities.

6. Внедрение SQL-кода/ Материал с Википедии — свободной энциклопедии. [Електронний ресурс]. Режим доступу URL: http://ru.wikipedia.org/wiki/Внедрение_SQL-кода

7. Класс WordPress для работы с Базой Данных (wpdb class). [Електронний ресурс]. Режим доступу URL: http://wp-kama.ru/id_178/klass-wordpress-po-rabote-s-bazoy-dannyih-wpdb-class.html

8. Безопасность в WordPress: как защитить свой блог от хакеров. [Електронний ресурс]. Режим доступу URL: http://oddstyle.ru/wordpress-2/stati-wordpress/bezopasnost-v-wordpress-kak-zashhitit-svoj-blog-ot-xakerov.html.

Поступила в редколегію 28.03.2012

Рецензент: д-р техн. наук, проф. С.Г. Удовенко, Харьковский национальный университет радиоэлектроники, Харьков.

ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ САЙТУ, ЩО ПОБУДОВАНИЙ НА СИСТЕМІ УПРАВЛІННЯ ВМІСТОМ WORDPRESS

І.В. Кобзев, К.Е. Петров, С.В. Калякін

Безпека є однією з базових концепцій веб-технологій в даний час. WordPressє одним з найбільш оптимальних рішень при виборі платформи для розробки сайтів, У статті розглядаються питання захисту веб-сайту, побудованого на системі управління контентом WordPress. Розглянуто методи несанкціонованого доступу та запропоновано варіанти захисту подібних сайтів-блогів.

Ключові слова: *система управління контентом, Wordpress, безпека, захист, блог, запит.*

SECURING CONTENT MANAGEMENT SYSTEM WORDPRESS WEB-SITE

I.V. Kobzev, K.E. Petrov, S.V. Kalyakin

Security has become a primary concept on the Web in the past few years. WordPress is one of the best solutions when choosing a platform for the development of sites This article deals with the protection of the web site, built on content management system WordPress. The methods of unauthorized access are reviewed and proposed options for protection of such sites- blogs.

Keywords: *control the system by content, Wordpress, safety, defence, blog, query.*