Soft

Информация по хосту приложений.

- Установка BookStack на VH
- Защита сайта на базе WordPress
- Nginx \ Apache

Установка BookStack на VH

Инфо:

Установка BookStack на сервере виртуального хостинга может оказаться проблематичной так как у разных провайдеров используются различные конфигурации, ограничения и т.д. и т.п.

Имея root доступ к серверу на который будет выполняться установка оказывается не такой сложной задачей.
В примере будет выполнена установка BookStack на сервер с панелькой **HestiaCP**, для пользователя *solaire*.

Установка:

1. Скачиваем последнюю сборку BookStack в ./public_html сайта:

```
it clone https: //github. com/BookStackApp/BookStack. git --branch release --single-branch
```

Переносим все файлы из диры приложения в public_html:

```
ind ./BookStack/ -mindepth 1 -maxdepth 1 -exec mv -t ./ {} +
```

Пользователю, в качестве примера *solaire* нужно предоставить доступ к composer. Для го логинимся от root и вызываем команду:

```
usr/local/hestia/bin/v-add-user-composer solaire
```

сле чего нужно запустить:

```
omposer install --no-dev
```

Добавим группу веб-сервера для работы с файлами приложения и установим омендуемые права:

```
hown -R solaire: www-data public_html
hmod -R 755 public_html
```

```
chmod -R 775 public_html/storage public_html/bootstrap/cache public_html/public/uploads
```

5. Обратно, от пользователя *solaire* копируем файл с конфигурацией. Указываем там реквизиты БД и почту:

```
p .env. example .env
hmod -R 640 .env
im .env
```

Генерируем ключ приложения. Он автоматически будем записан в .env файл: `php artisan generate`

В настройках панельки HestiaCP нужно поменять пользовательский document root на blic, то есть путь будет: `/home/solaire/web/book. domain. com/public_html/public`

Обновляем БД: `php artisan migrate`

Done. Можно войти в панель администратора по реквизитам:

```
dmin@admin. com
assword
```

Защита сайта на базе WordPress

Сайт, работающий на платформе WordPress, требует обеспечения безопасности, поскольку является одной из наиболее популярных и широко используемых систем управления контентом.

Ряд уязвимостей WordPress включает:

- Уязвимости плагинов и тем: Неправильно разработанные или устаревшие плагины и темы могут содержать уязвимости.
- Слабые пароли и учетные записи: Использование слабых паролей или повторение паролей для различных учетных записей делает сайт уязвимым для атак перебора паролей или подбора.
- Необновленный WordPress: Необновленная версия WordPress может содержать известные уязвимости, которые могут быть использованы злоумышленниками.
- SQL-инъекции и межсайтовые сценарии (XSS): Уязвимости, которые позволяют злоумышленникам внедрять и выполнять вредоносный код на сайте.
- Вредоносные программы: Злоумышленники могут использовать вредоносные программы для внедрения вредоносного кода на веб-сайт.

Наиболее распространенные причины взлома WordPress:

- Получение нежелательного доступа к административной панели или учетным записям.
- Внедрение вредоносного кода на сайт.
- Перехват данных пользователей, таких как логины и пароли.
- Загрузка вредоносных файлов на сервер.
- Выполнение атаки отказа в обслуживании (DDoS) для отключения сайта.

> Обеспечение безопасности WordPress-сайта включает регулярные обновления, использование надежных плагинов и тем, сложные пароли, использование защитных плагинов и мониторинг безопасности.

Для обеспечения безопасности и защиты от уязвимостей следует принять базовые меры:

1. Уязвимости плагинов и тем:

- Поддерживайте все плагины и темы в актуальном состоянии, регулярно обновляя их до последних версий.
- Устанавливайте плагины и темы только из надежных и проверенных источников, чтобы уменьшить риск внедрения вредоносного кода.

2. Слабые пароли и учетные записи:
   - Используйте сильные и уникальные пароли для всех учетных записей, включая административные.
   - Регулярно меняйте пароли и избегайте повторного использования паролей.
   - Рассмотрите возможность использования плагинов для управления паролями и реализации двухфакторной аутентификации.

3. Необновленный WordPress:
   - Регулярно проверяйте наличие обновлений для WordPress и его компонентов (плагины, темы).
   - Включите автоматические обновления, чтобы гарантировать, что ваш сайт будет иметь последние исправления уязвимостей.

4. SQL-инъекции и межсайтовые сценарии (XSS):
   - Используйте фильтры данных и экранируйте пользовательский ввод для предотвращения внедрения вредоносного кода.
   - Используйте параметризованные запросы и подготавливайте данные перед внесением их в базу данных.

5. Вредоносные программы:
   - Установите надежный антивирус и анти-вредоносное программное обеспечение на сервере.
   - Регулярно сканируйте веб-сайт на наличие вредоносного кода.
   - Ограничьте загрузку файлов на сервер и проверяйте загружаемые файлы на предмет вирусов или вредоносного кода.

Рекомендуемые настройки прав доступа для файлов и папок в WordPress:

1. Права доступа для файлов: 644
   - Владелец (Owner): Чтение и запись
   - Группа (Group): Чтение
   - Остальные пользователи (Others): Чтение
2. Права доступа для папок: 755
   - Владелец (Owner): Чтение, запись и выполнение
   - Группа (Group): Чтение и выполнение
   - Остальные пользователи (Others): Чтение и выполнение

Команды для массового изменения прав на папки и файлы:

```
find ./ -type d -exec chmod 755 {} \;
find ./ -type f -exec chmod 644 {} \;
```

Дополнительные рекомендации:

- Права доступа к файлам конфигурации (например, wp-config.php) должны быть установлены на 400 или 440, чтобы ограничить доступ к конфиденциальным данным, таким как данные базы данных.
- Если ваш хостинг поддерживает это, рекомендуется использовать suPHP или FastCGI, что позволит вам установить права доступа для файлов и папок 644 и 755 соответственно, и при этом сохранить безопасность.
- Избегайте установки прав доступа 777 для файлов и папок, так как это дает полные права на запись для всех пользователей, что может создать риск безопасности.

Отключение возможности выполнить PHP файлы в определенных каталогах:

Используя файл .htaccess:

Создайте файл .htaccess в каталоге, где вы хотите отключить выполнение PHP файлов, например в `./wp-content/uploads/`.

В файл .htaccess добавьте следующий код:

```
emoveHandler .php .phtml .php3
emoveType .php .phtml .php3
```

можно использовать этот код:

```
Files *.php>
eny from all
/Files>
```

от код отключит обработку файлов с расширением .php, .phtml и .php3 как исполняемых скриптов в выбранном каталоге.

Отключение индексирования и просмотр каталогов:

пользуя файл .htaccess:

утри файла .htaccess добавьте следующий код:

```
ptions -Indexes
```

Этот код запрещает серверу отображать список файлов и папок, если в URL указан каталог без конкретного файла.

Nginx \ Apache

Немного о веб-серверах.

Nginx и **Apache** являются двумя из наиболее популярных веб-серверов, и каждый из них имеет свои особенности и преимущества. Вот основные различия и случаи, когда лучше использовать каждый из них:

1. Архитектура и производительность:
   - Apache: Apache является модульным веб-сервером, который использует множество процессов или потоков для обработки запросов. Он хорошо подходит для обработки динамического контента, такого как скрипты PHP, и может интегрироваться с различными модулями и расширениями. Однако Apache может быть менее эффективным в обработке большого количества одновременных подключений.
   - Nginx: Nginx является событийно-ориентированным сервером, оптимизированным для обработки большого количества одновременных соединений. Он эффективно обрабатывает статический контент, такой как HTML, CSS и изображения. Кроме того, Nginx также может служить прокси-сервером, балансировщиком нагрузки или кэширующим прокси, что делает его хорошим выбором для обработки статического контента и проксирования запросов к другим серверам.

2. Конфигурация:
   - Apache: Apache имеет более гибкую и понятную конфигурацию, основанную на файлах `.htaccess` и конфигурационных файлах сервера. Это позволяет настраивать поведение сервера на уровне отдельных директорий и файлов, что полезно для хостинга с множеством веб-сайтов с разными требованиями.
   - Nginx: Конфигурация Nginx основана на блоках `server` и `location` в конфигурационных файлах. Она менее гибкая в сравнении с Apache, но обычно более производительная и легковесная. Однако, для изменения конфигурации Nginx требуется перезапуск или перезагрузка сервера.

3. Проксирование и балансировка нагрузки:
   - Apache: Apache может работать в режиме прокси-сервера, но его производительность в этой роли может быть ниже, особенно при обработке большого количества одновременных запросов. Apache также поддерживает балансировку нагрузки с помощью модуля mod_proxy_balancer.
   - Nginx: Nginx изначально разработан для работы в качестве прокси-сервера и обратного прокси. Он обладает высокой производительностью

и масштабируемостью в роли прокси и может эффективно распределять запросы между несколькими серверами в балансировке нагрузки.

Обработка подключений.

Apache использует модель "один процесс или поток на подключение". Это означает, что для каждого входящего подключения Apache создает новый процесс или поток, который обрабатывает запрос. Когда количество одновременных подключений увеличивается, Apache создает все больше процессов или потоков, что может привести к значительному потреблению памяти и ресурсов системы. Каждый процесс или поток занимает определенное количество памяти, что может привести к исчерпанию ресурсов сервера при очень большом количестве подключений.

Nginx использует событийную модель и асинхронный подход к обработке подключений. Вместо создания отдельного процесса или потока для каждого подключения, Nginx использует неблокирующий ввод-вывод (non-blocking I/O) и мультиплексирование событий для эффективной обработки множества подключений одновременно. Он может эффективно обслуживать множество запросов с помощью небольшого числа рабочих процессов или потоков, что приводит к более экономному использованию ресурсов сервера.

Например:

Apache	Nginx + Apache
Каждый новый запрос от клиента инициирует создание нового процесса или потока в Apache для обработки этого запроса.	Nginx работает как проксирующий сервер, принимая все входящие запросы от клиентов.
Создание и уничтожение процессов/потоков требует времени и ресурсов операционной системы.	Nginx быстро и эффективно обрабатывает статический контент, так как он может возвращать его из своего кэша или передавать напрямую клиенту без обращения к Apache.
При большом количестве одновременных запросов возникает нагрузка на систему в виде создания и управления множеством процессов/потоков, что может вызывать задержки и снижать производительность.	Когда Nginx обнаруживает запрос на динамический контент, он перенаправляет его к Apache для обработки.
	Apache, в свою очередь, обрабатывает эти динамические запросы, генерирует страницы, выполняет скрипты и обращается к базе данных при необходимости.
	Ответы от Apache возвращаются обратно в Nginx, который передает их клиентам.

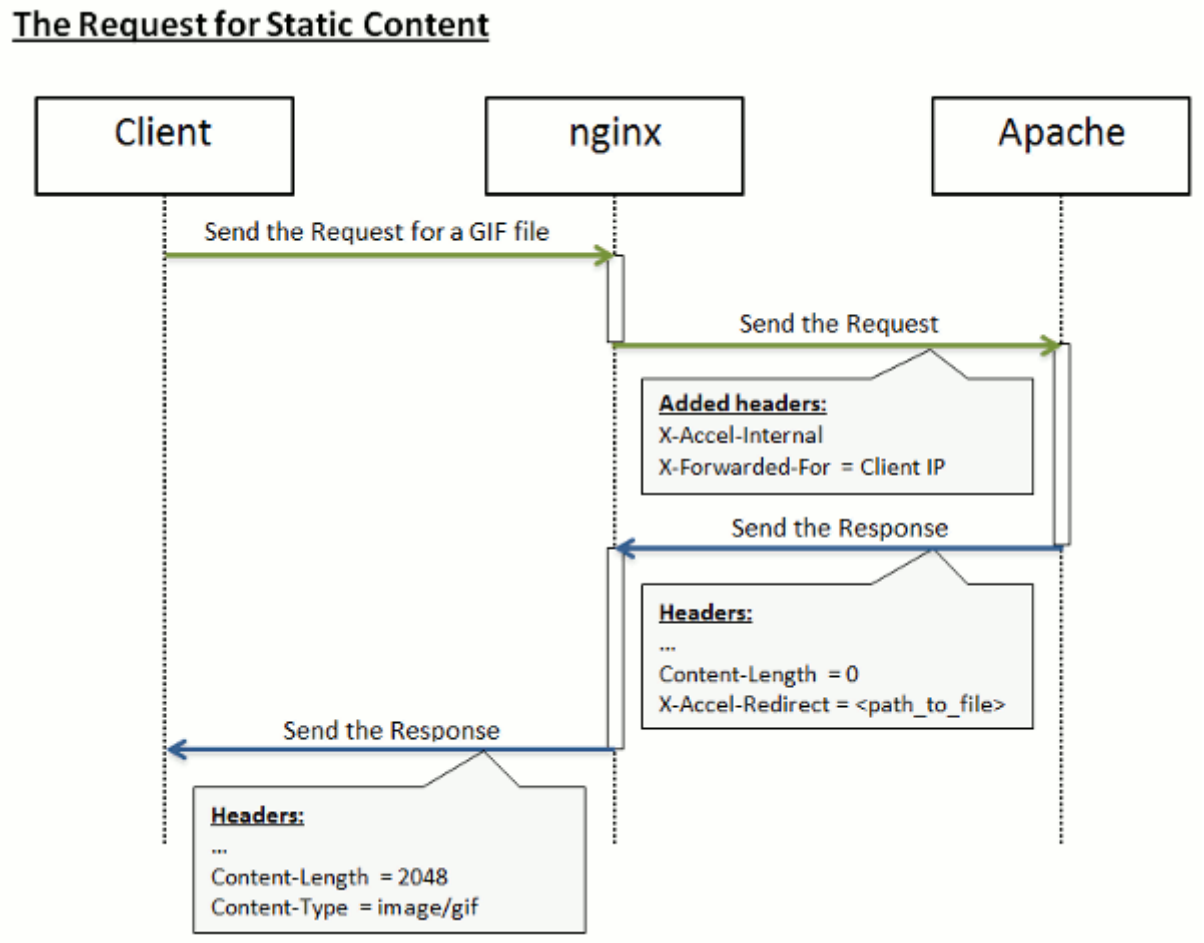
Такой подход позволяет эффективнее обрабатывать множество подключений по сравнению с просто использованием Apache, поскольку:

- Nginx способен обрабатывать большое количество одновременных подключений благодаря своей асинхронной модели событийного цикла и использованию неблокирующего ввода-вывода.
- Nginx кэширует статический контент и быстро отдает его клиентам, что уменьшает нагрузку на Apache и ускоряет доставку контента.
- Apache, в свою очередь, может сосредоточиться на обработке динамического контента, что позволяет ему эффективно выполнять скрипты и работать с базой данных.

Техническое преимущество комбинации Nginx + Apache.

- Увеличивается максимальное количество одновременных подключений к одному сайту.
- Сокращается потребление процессорного времени и памяти на сервере. Этот эффект будет наиболее ощутим для сайтов с большим объемом статического контента (фотогалереи, видеохостинги и так далее).
- Оптимизируется обслуживание посетителей с низкой скоростью соединения (GPRS, EDGE, 3G и т.д.). Например, допустим, что клиент со скоростью подключения 10 КБ/с запрашивает некий сценарий PHP, который генерирует ответ размером 100 КБ. Если на сервере не установлен nginx, то этот ответ доставляется веб-сервером Apache. В течение всех 10 секунд, необходимых для доставки ответа, Apache и PHP продолжают потреблять полный объем системных ресурсов для поддержания этого открытого подключения. Если же nginx установлен, Apache перенаправляет этот ответ ему (соединение между nginx и Apache очень быстрое, так как оба находятся на одном сервере) и высвобождает системные ресурсы. Благодаря тому, что nginx потребляет меньше памяти, общая нагрузка на систему сокращается. Если у вас много таких медленных подключений, использование nginx позволит вам значительно повысить производительность сайтов.

The Request for Static Content



The Request for Dynamic Content

