

Строим крепость

Фленов Михаил aka Horrific <http://www.vr-online.ru>

Безопасный сайт сделать очень сложно даже профессионалу, потому что постоянно приходится думать о безопасности. Если думать о женщинах и пиве, то ничего безопасного не получится. А что делать хостинговым компаниям для защиты себя и своих клиентов? Ведь если через дыру какого-то сайта взломают весь сервер, то это будет серьезная проблема и удар по престижу хостера! Спасение утопающих дело рук самих утопающих (хостер может только помогать) и сейчас мы изучим средства личной гигиены и личной безопасности, которые помогут от нелепостей и повысят иммунитет к злоумышленникам.

Предисловие

Чтобы победить микробов под крышкой кресла в комнате с буквами М или Ж, необходимо использовать Коммит. Чтобы микробов не было во рту, используют зубную пасту. А что делать, для защиты от микробов WEB серверу? Тут уже таким простым решением не обойтись, необходим целый комплекс мер, позволяющих построить настоящую защиту.

The screenshot shows the Apache Software Foundation website as it appeared in the early 2000s, viewed through Microsoft Internet Explorer. The browser's address bar shows <http://www.apache.org/>. The website features a blue header with the Apache logo and the text "The Apache Software Foundation" and <http://www.apache.org/>. The main content area is divided into several sections: "Apache Projects" with a list of various projects like HTTP Server, Ant, APR, etc.; "Welcome!" with a paragraph about the foundation's mission; "Support the Apache Software Foundation" with links to contribute, buy gear, or donate; "Latest News" with a headline about Apache Harmony becoming a top-level project; "Tapestry Wins Duke's Choice Award" with a photo of the award; "projects.apache.org Now Open"; and "Links about the Apache People". The right sidebar contains sections for "Foundation" (FAQ, Licenses, etc.), "Foundation Projects" (Conferences, Infrastructure, etc.), "How it works" (Introduction, Meritocracy, etc.), "Get Involved" (Mailing Lists, Version Control, etc.), "Download" (from a mirror), and "Search apache.org" with a search box and a "Go" button. The bottom of the browser window shows the "Internet" status bar.

Почему именно комплекс? Дело в том, что должно быть несколько уровней защиты. Один уровень всегда может дать сбой:

- в нем могут найти дыру;
- можно ошибиться в настройке;
- хакер может обойти уровень защиты.

В хорошо продуманной системе, даже если в сценарии будет ошибка, хакер не сможет далеко продвинуться и ничего узнать. Недавно мне пришлось тестировать один WEB сайт и, не смотря на наличия откровенной ошибки SQL Injection, я ничего, кроме версии, юзера и базы данных узнать не смог. При попытке изменить или удалить данные так же ничего не происходило, а меня отбрасывало на страничку с ошибкой безопасности хостера. Правильная настройка сервера, отличное распределение прав в базе данных, мониторинг и некоторые другие действия позволяют защитить сервер даже при наличии откровенных ошибок безопасности.

Настройку сервера я разделю бы на две части:

1. Тщательное конфигурирование сервера и открытых сервисов
2. Установка дополнительного софта для мониторинга или софта для создания дополнительных уровней защиты.

Конфигурирование сервера

Все начинается с тщательного конфигурирования сервера. Что я под этим понимаю? После установки ОС, все параметры установлены по умолчанию и так, как посчитал правильным разработчик дистрибутива. Тебе необходимо просмотреть все параметры и включить то, что нужно и самое главное - запретить то, что не будешь использовать. Лучше просматривать абсолютно все параметры, особенно, если ты устанавливаешь Linux сервер. Дело в том, что в зависимости от дистрибутива настройки в конфигурационных файлах могут очень сильно отличаться. Ты можешь думать, что какое-то действие запрещено, а разработчик твоего дистрибутива или новой версии ядра почему-то посчитал по-другому и изменил конфигурацию.

Мы постоянно встречаемся с тем, что в новых версиях ОС или ядер какие-то параметры становятся рекомендуемыми к использованию, а какие-то наоборот выходят из обращения. Необходимо отслеживать изменения в моде и контролировать конфигурацию.

JMC Research - Juan M. Casillas Web Site - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Go Links

Address http://www.jmcresearch.com/projects/jail/

JMCResearch

Knowledge for our questions
fun for everyone!

Main Weblog Journal Projects Bio Contact Login Misc

Search

more articles

Table of contents

- Introduction
- How Jail works
- Jail internals
- Configuring Jail
 - Setup Entries
 - Create directories
 - Adding users
 - Adding software
- Install HOWTO
- Download**
- Top Sections**
- Libros (10)
- Escalada (22)
- Load Balancer (18)
- LibTool (17)
- Jail (1)
- Top project downloads**
- Jail (16119)
- Load Balancer (3471)
- eventmonitor (2261)
- phptemplater (1536)
- Libxml (1271)
- Top active projects**
- libnetcomm (6602)
- Jail (6197)
- Libxml (4036)
- eventmonitor (4036)
- phptemplater (3661)
- Exits**
- Jail at Sourceforge
- Jail Mailing list**
- Jail Slashdotted
- Jail "Barrapunteo"
- Jail at Freshmeat
- Jail in the WWW
- Go back to projects
- Poll (79 votes)**
- What kind of templates you use?**
- ☐ None
- ☐ String replacements
- ☐ Custom

Introduction to Jail

Basic concepts and supported platforms

Introduction to Jail

Basic concepts and supported platforms


Jail Chroot Project is an attempt of write a tool that builds a chrooted environment. The main goal of Jail is to be as simple as possible, and highly portable. The most difficult step when building a chrooted environment is to set up the right libraries and files. Here, Jail comes to the rescue with a tool to automagically configures & builds all the required files, directories and libraries. Jail is licensed under the GNU General Public License.

Jail program has been written using C, and the setup script has been written using a bash script and perl. Jail has been tested under Linux (Debian 2.1 & 2.2, RedHat 6.1, 6.2 and 7.0 and Caldera Openlinux 7.0), Solaris (2.6), IRIX (6.5) and FreeBSD 4.3. Some people has contributed to jail with patches and ideas. Thanks to all of them.

Jail supports lots of interesting features:

- Runs on Linux, Solaris, IRIX and freeBSD (tested) and should run in any of the flavours of these operating systems.
- Modular design, so you can port Jail in an easy way.
- Support for multiple users in a single chrooted environment.
- Fully customizable user shell.
- Support for multiple servers: telnetd, sshd, ftpd...
- Easy to install thanks to the environment creation script.
- Should work in any UNIX.
- Ease of porting.
- Allows run any kind of program as a shell.

An html version of the mailing list has been added to the web site. Now you can read all the user contributions, ideas and patches here.



Jail now uses SourceForge. You can reach it at <http://jail.sourceforge.net>. We have a mailing list with archive and more!

Published at 26/09/2003 17:19:59
Last Updated 09/12/2003 20:25:07
By Juan M. Casillas

Download

Latest version is 1.9a
(32.9 Kb, tar.gz file)

> from JMCResearch

Quick links

- Jail Mailing list
- Jail install HOWTO
- Bug Report
- Feature Request
- CVS Repository
- Local CVS Repository
- Ask your questions

Sections

Jail

December's articles

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

More articles

Search

Search

Other versions

- *XML version
- *Printable version

Конфигурируя сервисы и ОС необходимо запрещать все, что ты не будешь использовать. На сервере не должно работать ничего лишнего. Все, не используемые возможности, должны быть строго настроено запрещены.

После установки всех ключиков и параметров, двигаемся дальше. Такую банальную тему как необходимость сменить все пароли по умолчанию - пропускаем. Это и ежику в тумане понятно, что все пароли должны быть пуленепробиваемыми.

База данных

Следующий уровень защиты - это права доступа в базе данных и в файловой системе. База данных - это самое важное хранилище, а для атаки SQL-Injection это основной инструмент для проникновения на сервер и сбора информации. Никогда в жизни сценарии не должны работать от имени привилегированного пользователя. Учетная запись, должна иметь только те права, которые ей необходимы для работы сценария и ни капли больше.



Любой доступ к системной информации должен запрещаться. В 99% случаев при работе с MySQL сценариям не нужен доступ к системной базе mysql, где хранится вся информация о пользователях, правах доступах и объектах сервера баз данных. Доступа не должно быть не только на запись или обновление, но и на просмотр данных. Очень часто даже простой взгляд на системные данные может дать хакеру слишком многое. При работе с базой данных MS SQL Server большинство использует уже существующую по умолчанию роль public. Громадная ошибка, ведь этой роли доступно слишком многое. Никогда не используй готовое, а создавай собственную роль и пользователя.

Модули WEB сервера

По конфигурации ОС и базы данных написано достаточно много мануалов, и рассмотреть абсолютно все просто нереально. Сказанное позволит тебе понять, в какую сторону двигаться и где искать дополнительную информацию.

Следующий этап обеспечения безопасности более интересный - модули для Apache. Они реально позволяют построить эффективную защиту от атак типа SQL-Injection, XSS и многих других. Но просто так установить модуль и решить все проблемы невозможно. Необходимо тщательное конфигурирование, и максимальный запрет всего, что не используется и может

нанести вред серверу.

mod_security

Несмотря на то, что безопасность WEB сервера в основном зависит от сценариев, которые выполняются на сервере и программистов, которые пишут эти сценарии, есть возможность защитить сервер вне зависимости от сценариев и их безопасности. Отличное решение данной проблемы бесплатный модуль для Apache под названием mod_security.

Принцип работы модуля схож с сетевым экраном, который встроен в ОС, только в данном случае он специально разработан для работы с протоколом HTTP. Модуль анализирует запросы пользователей и выносит свое решение о возможности пропустить запрос к WEB серверу или нет, на основе правил, которые задает администратор. Правила определяют, что может быть в запросе, а что нет. Недаром в заголовке официального WEB сайта красуется надпись: ModSecurity (mod_security) - Open Source Web Application Firewall (Сетевой экран WEB приложений с открытым исходным кодом).

В запросах, которые направляет пользователь на сервер, содержится URL адрес, с которого необходимо взять документ или файл. Что можно задать в правилах модуля, чтобы сервер стал безопаснее? Рассмотрим несколько простейших примеров:

- в строке URL не должно быть никаких обращений к файлу /etc/passwd, а значит, пути к этому файлу не должно быть в URL адресе;
 - через URL не должен передаваться JavaScript код. Если у тебя где-то передается он, то это серьезная ошибка и глупость. Такие вещи нужно передавать методом post. Так что, если запретить `<script>` и производные, то можно облегчить себе жизнь. Заметь, что только облегчить, ведь есть куча производных плюс кодирование содержимого строки. Хакер может замутить такой запрос, который обойдет фильтры, но усложнить ему жизнь мы просто обязаны;
 - в URL не должно быть одинарных кавычек, а все параметры, передаваемые сценарию, при использовании в запросах обязательно должны быть заключены в одинарные кавычки.
- Таким образом, модуль mod_security проверяет на основе заданных фильтров адреса URL, и если он нарушает правила, то запрос отклоняется.

Итак, модуль mod_security можно взять с сайта <http://www.modsecurity.org>. После установки в файле httpd.conf можно будет использовать дополнительные директивы, фильтрации запросов. Рассмотрим наиболее интересные из них:

- SecFilterEngine On - включить режим фильтрации запросов;
- SecFilterCheckURLEncoding On - проверять кодировку (URL encoding is valid);
- SecFilterForceByteRange 32 126 - разрешается использовать символы с кодами только из указанного диапазона. Символы, коды которых менее 32 принадлежат служебным символам типа перевод каретки, конец строки. Большинство из этих символов невидимы, но для них существуют коды, чтобы можно было обрабатывать нажатия соответствующих клавиш на клавиатуре. Но как же тогда хакер может ввести невидимый символ в URL? Это действительно невозможно, но вполне реально ввести их код. Например, чтобы указать символ с кодом 13, необходимо указать в URL адресе %13. Символы менее 32 и более 126 являются недопустимыми для адреса, поэтому вполне логично такие пакеты не пропускать до WEB сервера;
- SecAuditLog logs/audit_log - с помощью этого параметра задается файл журнала, в котором будет сохраняться информация об аудите;
- SecFilterDefaultAction "deny,status:406" - параметр задает действие по умолчанию. В данном случае указан по умолчанию запрет (deny);
- SecFilter xxx redirect:http://www.webkreator.com - если фильтр срабатывает, то пользователя переадресуют на сайт <http://www.webkreator.com>;
- SecFilter yyy log,exec:/home/apache/report-attack.pl - если фильтр срабатывает, то будет выполнен сценарий /home/apache/report-attack.pl;
- SecFilter /etc/password - в запросе пользователя не должно быть обращения к файлу /etc/passwd. Таким же образом нужно добавить запрет к обращению и к файлу /etc/shadow;
- SecFilter /bin/lis - в запросе пользователя не должно быть обращения к программам. В

данном случае запрещается команда ls, которая может позволить хакеру увидеть содержимое каталогов, если в сценарии есть ошибка. Необходимо запретить обращения к таким командам как cat, rm, cp, ftp и др;



- SecFilter "\.\/" - классическая атака, когда в URL указываются символы точек. Их не

должно быть там.

- SecFilter "delete[[:space:]]+from" - запрет текста delete ... from, что чаще всего используется в SQL запросах для удаления данных. Такой текст очень часто используется в атаках типа SQL injection. Помимо этого, рекомендуется установить следующие три фильтра:

1. SecFilter "insert[[:space:]]+into" - используется в SQL запросах для добавления данных;
2. SecFilter "select.+from" - используется в SQL запросах для чтения данных из базы;
3. SecFilter "<(.\n)+>" и SecFilter "<[[:space:]]*script" - позволяют защититься от XSS атак.

Мы рассмотрели основные методы, которые могут повысить безопасность WEB сервера. Таким образом, можно защищать даже сети из серверов.

mod_rewrite

Модуль mod_rewrite не является решением всех проблем, потому что при неправильной настройке, хакер легко может обойти защиту и взломать сайт. Примеры тому уже были на страницах][акера и пару свежих примеров я описал в книге "Web сервер глазами хакера". Но при правильной настройке, сайт становится не пробиваемым.

Что такое mod_rewrite? Это модуль, который позволяет преобразовывать URL адреса из одного вида в другой? Сколько раз мы видели большие порталы, в которых все страницы имеют расширение .html и при этом сайты динамические. Это же не реально построить на статичных html страницах большой портал, а тем более динамический! На самом деле под статикой прячутся PHP, или любые другие сценарии, просто пользователь видит дымовую завесу, которая прячет внутренности исполнения. Это достигается как раз с помощью модуля mod_rewrite, который пускает хакеру пыль в глаза, а заодно, позволяет сделать очень эффективные фильтры.

Mod_rewrite можно достаточно часто встретить у хостеров, но то, что он установлен, еще ни о чем не говорит. Необходима конфигурация, без которой модуль работать не будет.

Конфигурация происходит через файл .htaccess. Чтобы включить модуль, можно добавить следующие три строки:

```
RewriteEngine on
Options +FollowSymLinks
RewriteBase /abc
```

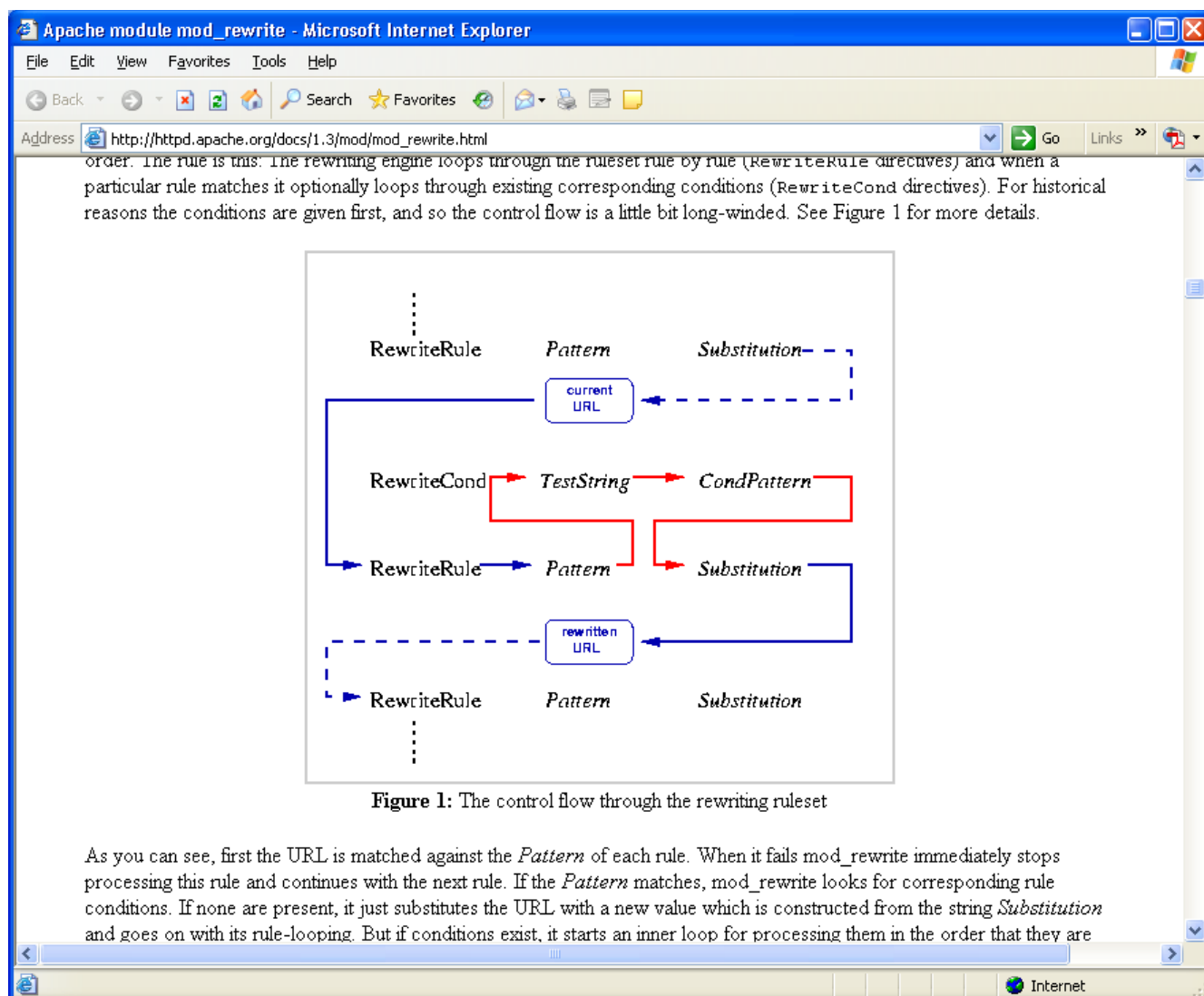
В первой строке мы включаем сам модуль с помощью установки значения директивы RewriteEngine в On. Следующая строка включает опцию FollowSymLinks. Эта опция разрешает преобразование символьных ссылок. Без этого дальнейшие телодвижения бессмысленны.

Следующая опция RewriteBase достаточно интересна, потому что позволяет изменять базовый URL. Допустим, что мы конфигурируем WEB директорию /documents/article и соответственно в ней и лежит наш .htaccess. Если RewriteBase установить в /erunda, то на WEB странице мы должны будем использовать URL /erunda/filename.php. Получив такой URL, сервер преобразует его в /documents/article/filename.php.

Вот мы уже и сделали первый шаг к безопасности - спрятали директорию /documents/article, где реально находится сценарий, а показали полную ерунду, т.е. /erunda. Не зная реального положения сценариев, хакеру будет сложнее обойти защиту, которую мы будем мутить далее.

Ну а теперь самое вкусное - правила преобразований того, что будет видеть пользователь. Для этого используется директива RewriteRule. Эта директива имеет следующий вид:

RewriteRule виртуал реал



Первый параметр - это виртуал, т.е. то, что WEB сервер получает в качестве URL, т.е. то, что преобразовывается, а реал - это реальный файл и его параметры, в которые нужно преобразовать.

Чтобы лучше было понятно, рассмотрим работу директивы на примере. Допустим, что у тебя есть сценарий `news.php`, который отображает определенную новость. Идентификатор новости передается через параметр `id`. То есть реальный URL для просмотра новости под номером 1 будет выглядеть так:

`http://www.твой_сервер.ru/news.php?id=1`

Чтобы хакер не видел этого, делаем следующую маскировку:

`RewriteRule ^news_([0-9]*).htm news.php?id=$1`

Теперь, для просмотра новости необходимо будет набрать в URL:

`http://www.твой_сервер.ru/news_1.html`

Получив такой виртуальный URL, модуль `mod_rewrite` преобразует его в реальный

`http://www.твой_сервер.ru/news.php?id=1` и корректно выполнит.

Теперь посмотрим, как мы добились такого счастья, разобрав директиву по частям:

- `RewriteRule` - начало правила;

- `^news_([0-9]*).htm` - шаблон, определяющий, как будет выглядеть URL для пользователя.

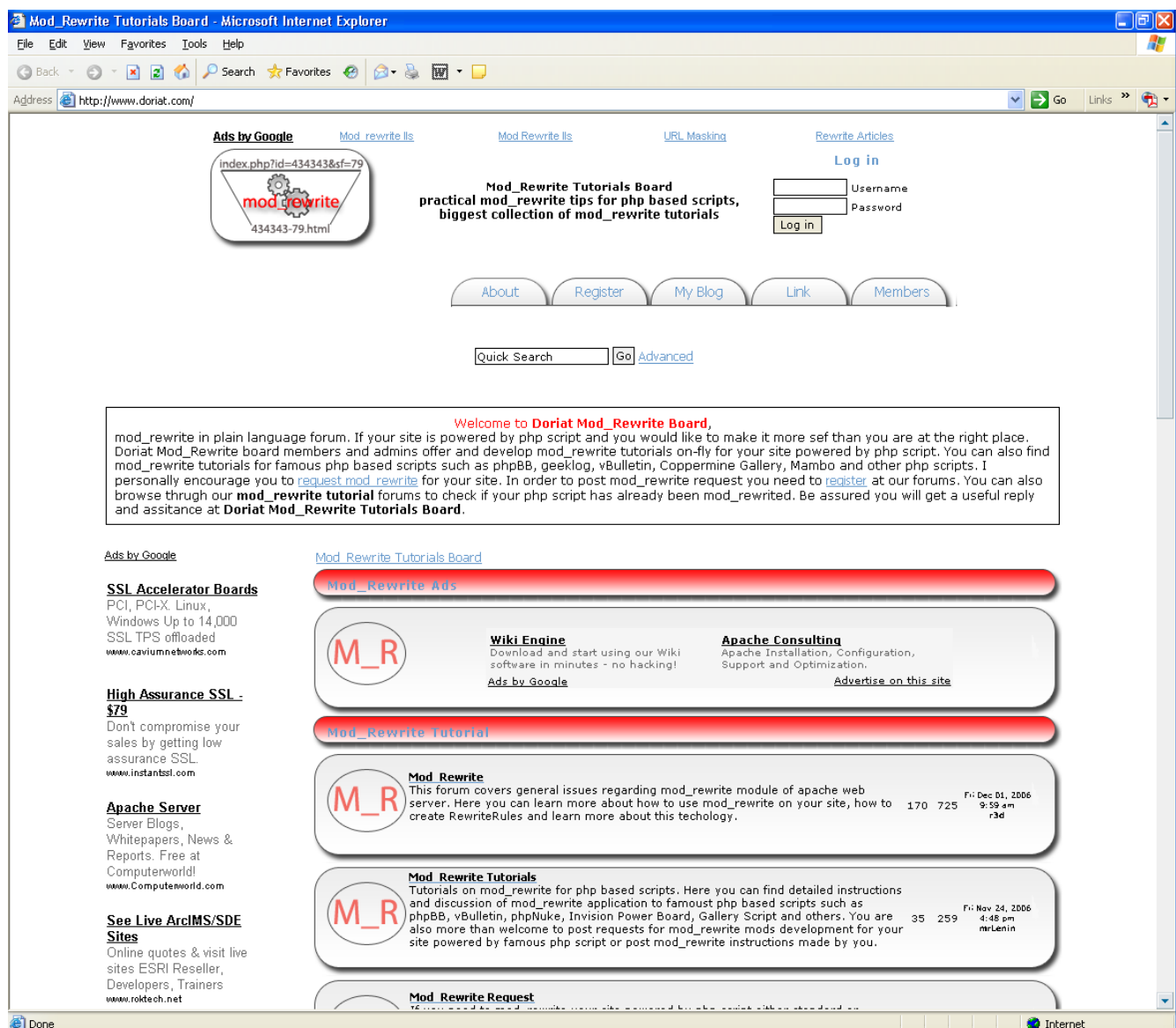
В самом начале стоит символ крыши. Нет, это не бандиты, которые будут предоставлять нам крышу, это такой символ (^), означающий начало строки. После этого идет текстовая статическая часть (`news_`), которая будет выделять наш шаблон из массы других. Ведь у нас

могут быть на сайте еще и статьи, форум и много чего еще интересного. После этого, в скобках указываем шаблон, того, что может быть в этом месте URL адреса. Так как идентификатор новости - это число, то шаблон выглядит так: [0-9] - любые символы от нуля до девяти, и после этого стоит звездочка, которая говорит, что цифр может быть несколько. После шаблона снова идет статическая часть.

- news.php?id=\$1 - во что нужно превратить виртуальный адрес. Здесь нужно заметить, что \$1 соответствует значению, вырезанному по первому шаблону (у нас он единственный). Благодаря шаблону, мы убиваем двух зайцев, и еще одного разрываем в клочья. Первый заяц заключается в том, что мы прячем реальный php сценарий. Второй заяц - это то, что хакер не видит имя параметра. Ну а третий заяц, которого просто порвало - это то, что хакер не сможет передать сценарию ничего, кроме чисел. Шаблон [0-9] вырезает любые буквы и символы, а значит, инъекция становится невозможной.

Вот тут нужно остановиться и заметить одну важную вещь - если хакер вычислит реальный сценарий и его параметр, то мы не то, что зайцев не убьем, мы даже муху не испугаем.

Обратившись к news.php напрямую, хакер обходит все правила mod_rewrite.



Чтобы хакер не смог обратиться к физическим файлам на сервере, можно выполнить одно из следующих условий:

1. Запретить прямой доступ к PHP сценариям, чтобы хакеру пришлось использовать виртуальную пыль, которую мы напустили;
2. Никогда не называть сценарии понятными именами. Главная страница никогда не должна иметь имя index.php или main.php, назови ее лучше enter_to_my_private.php или как-нибудь еще хуже. За новости не должен отвечать сценарий news.php, сценарий лучше назвать my_99545_news.php. Та же песня и с параметрами - забудь про id, sid, index, start, page и т.д.

Ну и, конечно же, на mod_rewrite надейся, а сам не плошай. Проверяй все своими руками и в сценарии, чтобы предотвратить любую атаку. Это вне комментариев и должно выполняться вне зависимости от установленных дополнительных средств контрацепции.

Мы рассмотрели простой пример с новостями, когда через параметры передается число. А что если нужно передавать строку? Да без проблем, просто пишем следующий шаблон:

```
RewriteRule ^news_([a-z0-9]*).htm news.php?id=$1
```

Здесь шаблон позволяет передавать не только числа, но и буквы от а до z. Это тоже не опасно, главное не разрешать использовать символы одинарной кавычки, запятые, тире и так далее. Буквы и цифры не несут в себе опасности, самое страшное - это символы.

Описать весь модуль mod_rewrite нереально, потому что у него очень много возможностей. Модуль очень гибкий, как Кабаева, но и этого уже достаточно для построения базовой, но надежной безопасности.

Итого

При правильном конфигурировании сервера и при помощи дополнительных модулей, можно защититься даже от самых ужасных сценариев, которые просто кишат ошибками SQL Injection или XSS. Если сервер взломали, то виноват не только программист, но и администратор и наоборот. Программеры и админы обязаны работать в унисон и дополнять друг друга, и помогут друг другу.

С другой стороны, никто не должен надеяться на дядю. Программист не должен надеяться на админа, что тот настроит сервер на максимальную безопасность и тот выдержит натиски врага. И администратор не должен надеяться на код, и на то, что программист не совершит ошибок.

Linux окружение

Конечно же, если ты работаешь с Linux и на одном сервере должно находиться несколько сайтов, то каждый из них желательно разместить в своей среде chroot. В отдельный chroot вместе с директорией сайта необходимо убирать и WEB сервер. Этот же финт нужно делать и тогда, когда на одном железе вериться не только WEB, но и другие сервисы, например, почта. В этом случае, взломав сайт, хакер не сможет получить доступ к почте. Да, мы будем конфигурировать так, чтобы хакер не смог ничего увидеть даже при наличии ошибки в сценарии и chroot один из идеальных способов решений.

Принцип работы chroot простой. Для начала создается директория (в Linux для этого существует команда chroot), которая является для программы корневой. Выше это директории программа, помещенное в закрытое окружение попасть не может. Чтобы проще было конфигурировать chroot, я всегда рекомендовал использовать утилиту jail. Она реально упрощает конфигурирование.

Посмотрите на рисунок "Окружение chroot". Здесь показана часть файловой системы Linux. Во главе всего стоит корневая директория /. В ней находятся /bin, /etc, /home, /usr и т.д. В /home расположены каталоги пользователей системы. Мы создаем здесь новую директорию, для примера назовем ее chroot и она будет является корнем для службы. В ней будут свои каталоги /bin, /usr и т.д. и служба будет работать с ними, а все, что выше /home/chroot будет недоступно. Просто служба будет считать, что /home/chroot - это и есть корень файловой системы.

На рисунке в рамку обведены папки, которые будут видны службе. Именно в этом пространстве будет работать служба и будет считать, что это и есть реальная файловая

система сервера.

JMCResearch
Knowledge for our questions
fun for everyone!

Main Weblog Journal Projects Bio Contact Login Misc

Search

Table of contents

- Introduction
- How Jail works
- Jail internals
- Configuring Jail
 - Setup Entries
 - Create directories
 - Adding users
 - Adding software
- Install HOWTO
- Download
- Top Sections
 - Libros (10)
 - Escalada (22)
 - Load Balancer (15)
 - LibTool (17)
 - Jail (1)
- Top project downloads
 - Jail (16119)
 - Load Balancer (3471)
 - eventmonitor (2261)
 - phptemplater (1536)
 - Libxml (127)
- Top active projects
 - libnetcomm (662)
 - Jail (6197)
 - Libxml (635)
 - eventmonitor (4636)
 - phptemplater (3661)
- Exits
 - Jail at Sourceforge
 - Jail Mailing list
 - Jail Slashdotted
 - Jail "Barrapunteo"
 - Jail at Freshmeat
 - Jail in the WWW
 - Go back to projects
- Poll (79 votes)
- What kind of templates you use?
 - None
 - String replacements
 - Custom

Introduction to Jail

Basic concepts and supported platforms

Introduction to Jail

Basic concepts and supported platforms

Jail Chroot Project is an attempt of write a tool that builds a chrooted environment. The main goal of Jail is to be as simple as possible, and highly portable. The most difficult step when building a chrooted environment is to set up the right libraries and files. Here, Jail comes to the rescue with a tool to automagically configures & builds all the required files, directories and libraries. Jail is licensed under the GNU General Public License.

Jail program has been written using C, and the setup script has been written using a bash script and perl. Jail has been tested under Linux (Debian 2.1 & 2.2, RedHat 6.1, 6.2 and 7.0 and Caldera Openlinux 7.0), Solaris (2.6), IRIX (6.5) and FreeBSD 4.3. Some people has contributed to jail with patches and ideas. Thanks to all of them.

Jail supports lots of interesting features:

- Runs on Linux, Solaris, IRIX and FreeBSD (tested) and should run in any of the flavours of these operating systems.
- Modular design, so you can port Jail in an easy way.
- Support for multiple users in a single chrooted environment.
- Fully customizable user shell.
- Support for multiple servers: telnetd, sshd, ftpd...
- Easy to install thanks to the environment creation script.
- Should work in any UNIX.
- Ease of porting.
- Allows run any kind of program as a shell.

An html version of the mailing list has been added to the web site. Now you can read all the user contributions, ideas and patches here.

SOURCEFORGE
net

Jail now uses SourceForge. You can reach it at <http://jail.sourceforge.net>. We have a mailing list with archive and more!

Published at 26/09/2003 17:19:59
Last Updated 09/12/2003 20:25:07
By Juan M. Casillas

Download

Latest version is 1.9a
(32.9 Kb, tar.gz file)
» from JMCResearch

Quick links

- Jail Mailing list
- Jail install HOWTO
- Bug Report
- Feature Request
- CVS Repository
- Local CVS Repository
- Ask your questions

December's articles

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

More articles

Search

Search

Other versions

- *XML version
- *Printable version

Если хакер проникнет в систему через защищенную службу и захочет просмотреть каталог /etc, то он увидит каталог /home/chroot/etc, но никак не системный /etc. Чтобы взломщик ничего не заподозрил, в каталоге /home/chroot/etc можно расположить все необходимые файлы, но содержащие некорректную информацию. Запросив файл /etc/passwd через уязвимую службу, хакер получит доступ к /home/chroot/etc/passwd, потому что служба видит его системным.

Файл /home/chroot/etc/passwd может содержать неверные пароли. На работу системы в целом это не повлияет, потому что система будет брать пароли из файла /etc/passwd, а службе реальные пароли системы не нужны, поэтому в файл /home/chroot/etc/passwd можно засунуть что угодно.

Далеко не все любят chroot из-за его не простой настройки, но при правильной настройке он реально помогает.