PHP глазами ХАКЕРА

2-е издание

Михаил Фленов

+cd

Безопасное программирование на PHP
Защита от SQL-инъекции в PHP
Описание реальных атак и защиты
Оптимизация Web-приложений
Работа с сетью

bhv

Михаил Фленов

PHP глазами ХАКЕРА

2-е издание

Санкт-Петербург

«БХВ-Петербург»

2010

УДК 681.3.06
ББК 32.973.26-018.2
Ф69

Фленов М. Е.

Ф69 PHP глазами хакера: 2-е изд., доп. и перераб. — СПб.: БХВ-Петербург, 2010. — 336 с.: ил. + CD-ROM

ISBN 978-5-9775-0546-8

Рассмотрены вопросы безопасности и оптимизации сценариев на языке PHP. Описаны типичные ошибки программистов, благодаря которым хакеры проникают на сервер, а также представлены методы и приведены практические рекомендации противостояния внешним атакам. Показаны реальные примеры взлома Web-серверов. Во 2-м издании книги большое внимание уделено защите баз данных, шифрованию, рассказано о тесте CAPTCHA, а также о распространенных приемах "борьбы за пользователя" (переадресации, всплывающих окнах и др.). Компакт-диск содержит исходные тексты примеров, рассмотренных в книге, а также полезные программы и утилиты.

Для Web-программистов, администраторов и специалистов по безопасности

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Ирина Иноземцева*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.12.09.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 27,09.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0546-8

© Фленов М. Е., 2010
© Оформление, издательство "БХВ-Петербург", 2010

Оглавление

Предисловие

Данная книга посвящена одному из популярнейших языков программирования Web-сайтов — PHP. С ее помощью вы научитесь создавать собственные сайты и делать их более эффективными и защищенными.

Чем она отличается от других? Большинство авторов ставит перед собой цель научить читателя программировать и только в конце книги обращает внимание на оптимизацию и безопасность. Но это неправильно, потому что если человека научить программировать неэффективно, то потом с помощью пары глав переучить его будет сложно. Привычки тяжело искоренить.

О безопасности нужно думать всегда, потому что это комплексная проблема. Нет такого решения, которое может гарантировать, что после его реализации ваш код станет абсолютно безопасным. Проблемы бывают разные, и решать их приходится по-разному.

Ошибки программы могут принести много пользы, потому что, выясняя природу ошибки, вы сумеете лучше разобраться с поставленным вопросом и сможете найти правильное решение. Чтобы ошибка не оказалась фатальной для вашего сайта и вашей карьеры программиста, желательно постоянно отслеживать тенденции в технологиях компьютерной безопасности и проверять свой сайт на предмет возможных уязвимостей.

Данная книга освещает язык программирования PHP, начиная с самых основ, и одновременно затрагивает аспекты безопасности и оптимизации работы сценариев. Таким образом, вы с самого начала будете учиться создавать быстрые и защищенные приложения. О безопасности нужно думать всегда, а не в конце работы. Я даже скажу больше — о ней нужно думать еще до того, как вы начнете писать код.

Почему книга называется "Программирование на PHP глазами хакера"? Что это за глаза? Это простые человеческие глаза, потому что хакеры тоже люди. Главное их отличие состоит в том, как они создают программы. Если вы создали код, который выполняет поставленные задачи, то вы программист.

Но если ваш код работает лучше, быстрее и надежнее, а главное, безопаснее для сервера, то вы хакер.

Хотя мы будем рассматривать безопасность и оптимизацию в течение всего периода изучения языка PHP, я не могу сказать, что вы получите исчерпывающие сведения. Очень многое может оказаться за пределами книги, потому что нельзя увидеть все и нельзя выработать абсолютно эффективные и универсальные алгоритмы на все случаи жизни. Универсальность очень часто несовместима с понятиями эффективности и безопасности. Мы же постараемся научиться мыслить так, чтобы создаваемые нами программы выполнялись наиболее эффективно.

Взломщики очень часто используют на взломанных системах собственные или сторонние сценарии на языке Perl для повышения привилегий или выполнения определенных действий. В связи с этим некоторые компании отказываются от использования Perl на своих серверах, и администраторы все реже устанавливают его, что позволяет в какой-то степени повысить безопасность. Но это только маленький шажок на пути к полноценной безопасности.

В данной книге мы рассмотрим, как взломщики могут писать собственные сценарии для взлома серверов. Нет, я не буду писать что-то из серии автоматических взломщиков, дабы меня не обвинили в том, что теперь количество хакеров увеличится в разы. Проблема в том, что средства, применяемые администраторами для защиты, могут использоваться взломщиком для вскрытия защиты. Аналогично, средства взлома могут быть использованы для тестирования безопасности и для защиты.

Я позиционирую книгу как средство повышения безопасности и надеюсь, что именно с этой целью вы будете применять свои знания. Даже простой нож может служить средством нарезки продуктов, а может использоваться как оружие. Хочется надеяться, что вы используете все по назначению, в том числе и знания.

В наше время знания становятся самым опасным оружием. Именно поэтому я буду делать больший упор на безопасности, и особо важные моменты буду замалчивать, дабы не подтолкнуть любопытных молодых людей к нарушению закона. Конечно же, одной этой книги недостаточно для взлома сервера, потому что нужны намного большие знания, в том числе и знания ОС.

Рассматривая примеры того, как хакер может взломать сценарии, написанные на языке PHP, мы будем, как правило, подразумевать, что сервер работает под управлением ОС UNIX или одной из UNIX-подобных систем, например, Linux. Дело в том, что основная часть Web-сайтов с PHP-сценариями работает под управлением именно этих операционных систем, а ОС Windows больше

используется совместно с технологиями Microsoft, например, ASP.NET. Использование PHP на сервере Microsoft — достаточно дорогое удовольствие. Поэтому для лучшего понимания материала вам необходимо иметь хотя бы поверхностное представление о какой-нибудь ОС семейства UNIX. А если вы знаете основы безопасности этой системы, то наш разговор будет более продуктивным. Для этого я рекомендую вам прочитать книгу "Linux глазами хакера" [3].

В данной книге будет рассматриваться самая популярная в Интернете связка — LAMP (Linux, Apache, MySQL и PHP). Именно на этих четырех китах стоит большинство сайтов в Интернете и 10 моих собственных сайтов. За свою практику я создал достаточно много сайтов на этой платформе и считаю ее очень удачной для интернет-решений.

Ни одна книга не сможет сделать из человека хакера. Точно так же никакой труд не сделает из обезьяны человека. Нужна долгая и кропотливая работа, чтобы стать хакером. В этой книге я не собирался научить кого-то хакерскому искусству. Если вы купили книгу только из-за этого слова, то можете даже разочароваться. Но если вы купили ее для получения знаний о безопасности, то можете найти для себя много нового и интересного. Я, по крайней мере, надеюсь на это и ставил перед собой такую задачу.

Благодарности

Очень хочется поблагодарить всех тех, кто помогал мне в создании этой книги. Я не буду благодарить в порядке значимости, потому что каждая помощь очень значима для меня и для такой книги. Поэтому порядок не несет в себе никакого смысла, а выбран так, чтобы постараться никого не забыть.

Хочется поблагодарить издательство "БХВ-Петербург", с которым у меня сложилось уже достаточно долгое и продуктивное сотрудничество. Надеюсь, что это сотрудничество не прервется никогда. Спасибо редакторам и корректорам за то, что указывают на мои недочеты и помогают сделать книгу лучше и интереснее.

Хочется поблагодарить свою семью, которая терпит мои исчезновения за компьютером. Я прекрасно понимаю, что тяжело видеть мужа и отца семейства, который вроде бы дома и в то же время отсутствует. Это напоминает загадку: "Висит груша, нельзя скушать". Я, правда, не груша и нигде не подвешен, но вот пользы от того, что я нахожусь дома, для детей и жены мало ☺.

Хочу поблагодарить тех, кто разрешил тестировать их серверы и сценарии в целях выявления ошибок, а также позволил просмотреть свои сценарии

и настройки безопасности. Сотрудничество оказалось взаимовыгодным: все эти компании и владельцы сайтов обеспечили себе более спокойный сон, потому что до этого момента их сайты никто не тестировал, и ошибок было очень много.

Единственная благодарность, которую я хотел бы подчеркнуть больше всех и придать ей большую значимость, — это вам, за то, что купили книгу, и моим постоянным читателям, которые также участвуют в создании книг.

Все мои последние работы основываются на вопросах и предложениях читателей, с которыми я регулярно общаюсь через свой сайт **www.flenov.info**. Я постараюсь помочь по мере возможности и жду любых комментариев по поводу этой книги. Ваши замечания помогут мне сделать эту работу лучше.

Как устроена эта книга

Мы будем рассматривать затронутые в книге темы достаточно подробно, чтобы читатели с разным уровнем подготовки и разным опытом могли понять описываемый материал.

Книга состоит из пяти глав, которые последовательно погружают вас в мир Web-программирования с помощью PHP. Давайте кратко посмотрим, что вас ожидает в каждой из пяти глав.

Глава 1. Введение. В этой главе мы узнаем, кто такие хакеры и как стать хакером. Мы разберемся, в чем отличие хакеров от крэкеров, и вы должны четко это уяснить, потому что врага нужно знать в лицо или хотя бы понимать его психологию. Помимо этого мы познакомимся с основами интерпретируемого языка PHP и узнаем, для чего он нужен и как и где может применяться.

Глава 2. Основы PHP. В этой главе мы приступим к написанию сценариев, начиная с самых основ — оформление кода, переменные, управление выполнением сценария и т. д. Хотя будут обсуждаться азы, уже здесь мы познакомимся с интересными алгоритмами и основами безопасности рассматриваемых технологий. Безопасность — очень сложная тема, которая затрагивает абсолютно каждую строчку кода, и одной главы для освещения этой темы недостаточно. Например, при обсуждении темы передачи параметров от пользователя к серверу мы узнаем, что для этого существует несколько методов, и ни один из них нельзя считать безопасным, поэтому мы затронем методы проверки параметров.

Глава 3. Безопасность. В этой главе будет рассматриваться безопасность создаваемого кода, но только общие принципы и основы. Каждая глава книги

затрагивает вопросы безопасности в той или иной степени. Здесь же мы сделаем упор на теорию, а в *главах 2* и *5* при рассмотрении примеров мы будем больше уделять внимания практике.

Вы также узнаете, что обеспечение безопасности — это комплексная задача, и, написав идеальный сценарий, но неправильно настроив сервер, вы не сможете спать спокойно. Максимально безопасным должно быть все.

Глава 4. Оптимизация. Безопасность, скорость работы и удобство — чаще всего противоречивые понятия, поэтому постоянно приходится искать золотую середину, чтобы сценарий работал быстро и безопасно, а код был легко читаемым и удобным в сопровождении.

Глава 5. Примеры работы в PHP. Эта глава содержит практические решения типичных задач совместно с теоретическими знаниями. Мы рассмотрим сетевые функции и тут же напишем несколько интересных примеров. Мы будем говорить об аутентификации и авторизации и узнаем, как решается эта задача в Web-приложениях.

Каждая технология будет рассматриваться с точки зрения хакера и безопасности. Мы достаточно подробно обсудим некоторые вопросы того, как хакеры взламывают сценарии (например, атаку SQL Injection), чтобы вы знали, какими могут быть меры противодействия атакам.

На этом я заканчиваю вступление, и мы можем переходить к самому интересному в этой книге — к PHP, программированию и безопасности.

Глава 1

Введение

Программирование интернет-приложений, в том числе и работающих в браузере, влечет за собой большую ответственность за безопасность и защиту данных. Если вы напишете календарь, калькулятор или текстовый редактор, то такие программы, скорее всего, не привлекут внимания хакеров, и никто не захочет их взламывать. Программы, не работающие с Интернетом, неинтересны взломщикам.

Но если вы разрабатываете даже самый простой Web-сайт, ваш труд моментально привлечет внимание злоумышленников. Один начинающий программист как-то сказал: "Кому нужна моя личная домашняя страничка? Кто будет ее взламывать ради дефейса?" Внимание может привлечь не сам сайт или информация на нем, и дефейс (смена главной страницы) — далеко не единственная цель хакеров. Целью может быть не сам сайт, а сервер, на котором все работает.

На одном сервере может работать множество сайтов одновременно, а при неправильной настройке прав доступа, взломав только один из них, злоумышленник может получить доступ ко всем сайтам.

Взломанные и подконтрольные злоумышленникам серверы в Интернете являются очень ценным ресурсом, ведь с их помощью можно проводить атаки на другие ресурсы Интернета, причем делать это можно анонимно.

Я считаю, что безопасность при разработке Web-сайтов очень важна, и мы будем думать о ней постоянно на протяжении всей книги. Да, безопасность — это комплексный вопрос, ответ на который кроется не только в особенностях языка программирования, но и сервера баз данных, и ОС, под управлением которой все это работает. Безопасность невозможно измерить, поэтому очень сложно сказать, когда безопасность становится достаточной.

1.1. Кто такие хакеры?

Это довольно спорный вопрос, и я достаточно много писал о том, кто такие хакеры и как ими стать. Если вы уже читали мои книги, то этот раздел покажется вам знакомым, но отличия есть, потому что в данном случае мы делаем упор на Web-программирование и Web-взлом. Давайте разберем понятие "хакер" с позиции, с которой я буду рассматривать его в данной книге. Но для начала надо немного углубиться в историю.

Понятие "хакер" зародилось, когда только начинала распространяться первая сеть ARPANET. Тогда этот термин обозначал человека, хорошо разбирающегося в компьютерах. Некоторые даже подразумевали под хакером человека, "помешанного" на компьютерах. Понятие ассоциировали со свободным компьютерщиком, человеком, стремящимся к свободе во всем, что касалось его любимой "игрушки". Благодаря этому стремлению и тяге к свободному обмену информацией и началось такое бурное развитие Всемирной сети. Именно хакеры помогли развитию Интернета и создали FIDO. Благодаря им появились UNIX-подобные системы с открытым исходным кодом, под управлением которых сейчас работает большое количество серверов, в том числе и в Интернете.

В те далекие времена еще не было вирусов и не внедрилась практика взломов сетей или отдельных компьютеров. Образ хакера-взломщика появился немного позже. Но это только образ. Настоящие хакеры никогда не имели никакого отношения к взломам, а если хакер направлял свои действия на разрушение, то это резко осуждалось виртуальным сообществом. Даже самые яркие представители борцов за свободу не любят, когда кто-либо вмешивается в их личную жизнь.

Настоящий хакер — это творец, а не разрушитель. Так как творцов оказалось больше, чем разрушителей, то истинные хакеры выделили тех, кто занимается взломом, в отдельную группу и назвали их крэкерами (взломщиками) или просто вандалами. И хакеры, и взломщики являются гениями виртуального мира. И те, и другие борются за свободу доступа к информации. Но только крэкеры взламывают сайты, закрытые базы данных и другие источники информации с целью собственной наживы, ради денег или минутной славы, такого человека можно назвать только преступником (кем он по закону и является!).

Если вы взломали программу, чтобы увидеть, как она работает, то вы — хакер, а при намерении ее продать или просто выложить в Интернете crack (крэк) становитесь преступником. Ежели вы взломали сервер и сообщили администрации об уязвимости, то вы, несомненно, — хакер, но коли уничтожили информацию и скрылись, то это уже преступление.

Жаль, что многие специалисты не видят этой разницы и путают хакерские исследования с правонарушениями. И наибольшую роль тут сыграли журналисты, не понимающие разницы. Для журналистов главное — сенсация, а правильность применения терминов не имеет особого значения. Если читатели хотят услышать страшилку, то журналист напишет эту страшилку о том, что хакер что-то взломал. Да, взломщика можно назвать хакером, но слишком частое использование термина в отношении злоумышленника сделало термин негативным.

Хакеры интересуются системой безопасности систем и серверов для определения ее надежности (или в образовательных целях), а крэкеры — с целью воровства или уничтожения данных.

Перечислю категории крэкеров.

- *Вирусописатели* применяют свои знания для того, чтобы написать программу разрушительной направленности.
- *Вандалы* стремятся уничтожить систему, удалить все файлы или нарушить работу сервера.
- *Взломщики компьютеров/серверов* совершают "кражу со взломом" с целью наживы, выполняя, зачастую, чьи-либо заказы на получение информации, они очень редко используют свои знания в разрушительных целях.
- *Взломщики программ* снимают защиту с программного обеспечения и предоставляют его для всеобщего использования. Такие люди наносят ущерб фирмам-создателям программного продукта и государству. Программисты должны получать зарплату за свой труд.

Чтобы еще раз подчеркнуть разницу между хакером и крэкером, можно сравнить их со взломщиками программ. Все прекрасно понимают, что многие фирмы-разработчики завышают цены на свои программные продукты. Крэкер будет бороться с ценами с помощью снятия защиты, а хакер создаст свою программу с аналогичными функциями, но меньшей стоимости или вообще бесплатную. Так, представителей движения Open Source можно причислить к хакерам, а тех, кто пишет крэки, нужно относить к взломщикам, т. е. крэкерам.

Мне кажется, что путаница в понятиях отчасти возникла из-за некомпетентности в этом вопросе средств массовой информации. Журналисты популярных СМИ, не вполне разбираясь в проблеме, приписывают хакерам взломы, делая из них преступников. Солдат далеко не всегда является убийцей, несмотря на то, что он профессионал и носит оружие.

На самом же деле, хакер — это просто профессиональный компьютерщик и даже далеко не всегда программист или специалист по безопасности. Истинные хакеры никогда не используют свои знания во вред другим. Именно

к этому я призываю в данной книге, и никакого конкретного взлома или вирусов она не содержит. Здесь будет представлена только полезная и познавательная информация, которую вы сможете использовать для умножения своих знаний. Если вы надеялись прочитать что-нибудь про взлом NASA или правительственных серверов, то вы можете разочароваться.

Если ваш сайт взломал хакер, то он покажет вам уязвимость и, возможно, даже посоветует, как исправить ошибку. Но если ошибку найдет крэкер, то можно потерять данные или главную страницу.

Так как путаница все равно уже возникла и хакерами называют как специалистов, так и взломщиков, очень часто в Интернете и в СМИ можно увидеть другую градацию — цветовую. Те, кто взламывает сайты в корыстных или вандалистских целях, стали называться Black Hat Hackers (черные хакеры). В то время как специалисты по безопасности, которые исследуют безопасность и взламывают сайты для построения еще более защищенных систем, стали называться White Hat Hacker (белые хакеры). Слово Hat (шапка) в обоих терминах часто может опускаться. Самое главное в этих терминах — это цвета.

Черный и белый цвета являются противоположностями. Точно так же и хакеры White и Black занимают противоположные стороны баррикад в мире безопасности. Первые исследуют сайты для защиты, а вторые для взлома.

В большинстве случаев к хакерам (белым) относятся опытные или молодые люди, которыми движет стремление к изучению чего-то нового. Они тестируют сайты для того, чтобы узнать, как те работают, и если в процессе изучения будет найдена ошибка, то программист сайта или администратор узнают об этом первыми.

Крэкеры (или черные хакеры) — это, в основном, молодые люди. В большинстве случаев ими движет стремление показать свое превосходство. Такие крэкеры могут только изменить главную страницу или сыграть с пользователями или администраторами безобидную шутку. Я чаще встречал далеко не опытных людей этого типа, знания которых не уходят дальше запуска готовой программы, которая и производит непосредственно взлом. Но если крэкером движет идеологическое соображение, то это уже опасно. Такой человек может уничтожить информацию, потому что здесь на первый план выходит стремление максимально нарушить работу сервера и сайта. Чаще всего такие люди очень опытны и обладают глубокими знаниями в сфере безопасности.

Хакеры должны знать не только компьютер, но и популярные ОС и желательно программирование. Когда мы будем рассматривать атаки, которые используют хакеры, то вы увидите, что без навыков программирования реа-

лизовать большинство из этих приемов будет невозможно. Если вы решили познакомиться с другими направлениями безопасности, то могу посоветовать прочитать мои книги "Программирование в Delphi глазами хакера" [1] и "Программирование на C++ глазами хакера" [2]. Надеюсь, это поможет вам научиться создавать собственные шуточные программы и хакерский программный продукт.

Итак, хакером может быть как взломщик, так и специалист по защите информации. Мы будем рассматривать программирование, в основном, с точки зрения взломщика. Чтобы писать защищенные приложения, нужно четко представлять себе, как злоумышленники смогут попытаться взломать приложение или нарушить его работу.

1.2. Как стать хакером?

Этот вопрос задают себе многие, но точного и быстрого рецепта вам не даст никто. Я постараюсь выделить некоторые общие аспекты, но все зависит от конкретной области, в которой вы хотите стать лучшим.

Сравним компьютерного специалиста со строителем. В каждой профессии существует некая специализация. Хорошим строителем может быть отличный каменщик или штукатур. Точно также и хакером может быть специалист по операционным системам (например, UNIX) или программист (приложений или Web-сайтов). Все зависит от ваших интересов и потребностей.

Приведу некоторые рекомендации, которые помогут вам стать настоящим хакером и добиться признания со стороны друзей и коллег.

1. Вы должны знать свой компьютер и научиться эффективно им управлять. Если вы будете еще и знать в нем каждую "железку", то это только добавит к вашей оценке по "хакерству" большой и жирный плюс.

   Что я подразумеваю под умением эффективно управлять своим компьютером? Знание всех возможных способов выполнения каждого действия и умение в каждой ситуации использовать оптимальный. В частности, вы должны научиться пользоваться "горячими" клавишами и не дергать мышь по любому пустяку. Нажатие клавиши выполняется быстрее, чем любое, даже маленькое, перемещение мыши. Просто приучите себя к этому, и вы увидите все прелести работы с клавиатурой. Лично я использую мышь очень редко и стараюсь всегда применять клавиатуру.

   Маленький пример на эту тему. Мой бывший начальник всегда копировал и вставлял данные из буфера обмена с помощью кнопок на панели инструментов или команд контекстного меню, которое появляется при щелчке правой кнопкой мыши. Но если вы делаете так же, то, наверное, знаете,

что не везде есть кнопки **Копировать**, **Вставить** или соответствующие пункты в контекстном меню. В таких случаях мой начальник набирал текст вручную. А ведь можно было бы воспользоваться копированием/вставкой с помощью "горячих клавиш" <Ctrl>+<C>/<Ctrl>+<V> или <Ctrl>+<Ins>/<Shift>+<Ins>, которые достаточно универсальны и реализованы практически во всех современных приложениях (даже там, где не предусмотрены кнопки и меню). Просто эти клавиши чаще работают на уровне ОС и их даже не нужно реализовывать. Единственные поля ввода, в которых горячие клавиши явно отключаются, — поля для ввода паролей и регистрационных кодов (последнее не всегда справедливо).

За копирование и вставку в стандартных компонентах Windows (строки ввода, текстовые поля) отвечает сама операционная система. Если программист не предусмотрел кнопку, то это не значит, что данного действия нет. Оно есть, но доступно через "горячую клавишу".

Еще один пример. Я работал программистом на крупном предприятии (более 20 000 работников). Моей задачей было создать программу ведения базы данных для автоматизированного формирования отчетности. Большое количество параметров вводилось вручную. Первый вариант программы работал без "горячих клавиш", и для ввода данных требовалось 25 человек. После внедрения "горячих клавиш" производительность возросла, и с программой работало уже менее 20 человек. Экономия заметна даже без увеличительного стекла.

2. Вы должны досконально изучать все, что вам хочется знать о компьютерах. Если вас интересует графика, то вы должны освоить лучшие графические пакеты, научиться рисовать в них любые сцены и создавать самые сложные миры. Если вас интересуют сети, то старайтесь узнать о них все. Если вы считаете, что познали уже все, то купите книгу по данной теме потолще, и вы поймете, что сильно ошибались. Компьютеры — это такая сфера, в которой невозможно знать все!!! Уже через год ваши знания могут устареть, и придется снова браться за учебу.

   Хакеры — это, прежде всего, профессионалы в каком-нибудь деле. И это не обязательно должен быть компьютер или какой-то определенный язык программирования. Хакером можно стать в любой области, но мы в данной книге будем рассматривать только компьютерных хакеров.

3. Желательно уметь программировать. Любой хакер должен знать, как минимум, один язык программирования, а лучше даже несколько языков. Лично я рекомендую всем изучить для начала Delphi, C++ или C#. Delphi достаточно прост, быстр, эффективен, а главное, — это очень мощный язык. C++ — признанный стандарт во всем мире, но немного сложнее

в изучении. C# — современный язык, который может стать будущим стандартом. Однако это не означает, что не надо владеть другими языками.

Вы можете научиться программировать на чем угодно, даже на языке классический Basic (использовать его я не советую, но знать этот язык не помешает). Хотя я не очень люблю Visual Basic за его ограниченность, неудобство и другие недостатки, я видел несколько великолепных программ, которые были написаны именно на этом языке. Глядя на них, сразу хочется назвать их автора хакером, потому что это действительно виртуозная и безупречная работа. Создание из ничего чего-то великолепного как раз и есть искусство хакерства.

Современный Basic.NET стал намного лучше, но если уж и писать что-то для .NET, то, может быть, сразу же на языке, который создавался специально под эту платформу, — C#? Этот язык прост как Delphi, удобен как Java и мощен почти как C++. Он сочетает в себе лучшее из разных языков, которые были когда-либо созданы.

По ходу изучения книги вы увидите, что без навыков программирования некоторые приемы были бы невозможны. Используя готовые программы, написанные другими хакерами, вы можете стать только взломщиком, а для того, чтобы стать хакером, желательно все же научиться создавать свой код.

Хакер — это творец, человек, который что-то создает. В большинстве случаев это касается кода программы, но можно создавать и графику, и музыку, что тоже относится к искусству хакера. Но даже если вы занимаетесь компьютерной музыкой, умение программировать повысит ваш уровень. Сейчас писать свои программы стало гораздо легче. С помощью таких языков, как Delphi или C#, можно создавать простые утилиты за очень короткое время, и при этом вы не будете ни в чем ограничены. Так что не поленитесь и изучите программирование.

4. Не тормозите прогресс. Хакеры всегда боролись за свободу информации. Если вы хотите быть хакером, то тоже должны помогать другим. Хакеры обязаны способствовать прогрессу. Кто-то с этой целью пишет программы с открытым кодом, а кто-то просто делится своими знаниями.

   Открытость информации не означает, что вы не можете зарабатывать деньги. Это никогда не возбранялось, потому что хакеры тоже люди, и тоже хотят кушать, и должны содержать свою семью. Самое главное — это созидание. Вот тут проявляется еще одно отличие хакеров от крэкеров: хакеры "создают", а крэкеры "уничтожают" информацию. Если вы написали какую-нибудь уникальную шуточную программу, то это вас де-

лает хакером. Но если вы изобрели вирус, который с улыбкой на экране уничтожает диск, то вы — крэкер-преступник.

В борьбе за свободу информации может применяться даже взлом, но только не в разрушительных целях. Вы можете взломать какую-либо программу, чтобы посмотреть, как она работает, но не убирать с нее систем защиты. Нужно уважать труд других программистов, не нарушать их авторские права, потому что защита программ — это их хлеб.

Представьте себе ситуацию: вы украли телевизор. Это было бы воровство и преследовалось бы по закону. Многие люди это понимают и не идут на преступление из-за боязни наказания. Почему же тогда крэкеры спокойно ломают программы, не боясь закона? Ведь это тоже воровство. Лично я приравниваю взлом программы к воровству телевизора с полки магазина и считаю это таким же правонарушением.

При этом вы должны иметь право посмотреть на код купленной программы. Ведь вы же можете вскрыть свой честно купленный телевизор, заглянуть под крышку, и никто не будет вас преследовать по лицензионным соглашениям. Кроме того, вас же не заставляют регистрироваться, когда вы честно приобретаете товар, а процедура активации купленной программы это предусматривает.

Я понимаю разработчиков программ, которые пытаются защитить свой труд. Я сам программист и продаю свои программы. Но я никогда не делаю сложных систем защиты, потому что любые попытки "предохранения" портят жизнь законопослушным пользователям, а крэкеры все равно взломают. Какие только "замки" не придумывали крупные корпорации, чтобы защитить свою собственность, но Crack существует на любые программы, большинство из которых взламывалось еще до официального выхода на рынок. С нелегальным распространением программ нужно бороться другими методами, а системы активации или ключей бесполезны.

В цивилизованном мире программа должна иметь только простое поле для ввода некоего кода, подтверждающего оплату, и ничего больше. Производителям программного продукта следует отказаться от активации и сложной регистрации. Но и пользователи должны быть честными, потому что любой труд должен оплачиваться. А то, что какой-то товар (программный продукт) можно получить бесплатно, еще не означает, что вы должны это делать.

5. Знайте меру. Честно сказать, я уважаю Билла Гейтса за то, что он создал Windows и благодаря ей сделал компьютер доступным для каждого в этом мире. Если раньше пользоваться компьютерами могли только люди

с высшим образованием и математическими способностями, то теперь он доступен каждому ребенку.

Единственное, что я не приветствую, — это методы, которыми продвигается Windows на компьютеры пользователей. Мне кажется, что уже давно пора ослабить давление, тогда Windows, наоборот, станет более популярной, и у многих пропадет ненависть к корпорации и ее руководству.

Нельзя просто так лишать денег другие фирмы только из-за того, что ты проиграл в конкурентной борьбе, как это произошло в случае с Netscape Navigator. Тогда Microsoft не удалось победить фирму Netscape честным образом, и Microsoft сделала свой браузер бесплатным, потому что у корпорации достаточно денег, и она может себе это позволить. Но почему нельзя было просто уйти от борьбы и достойно принять проигрыш? Ведь доходы фирмы от перевода браузера на бесплатную основу увеличились незначительно, а интеграция Internet Explorer в ОС — чистый фарс.

Для продвижения своих продуктов Microsoft могла найти и другие способы, а конкуренция должна быть честной. Нельзя требовать от пользователя добросовестного отношения к своим продуктам, когда сам играешь нечестно.

6. Не изобретайте велосипед. Тут опять действует созидательная функция хакеров. Они не должны стоять на месте и обязаны делиться своими знаниями. Например, если вы написали какой-то уникальный код, то поделитесь им с ближними, чтобы людям не пришлось создавать то же самое. Вы можете не выдавать все секреты, но должны помогать другим.

   Ну, а если к вам в руки попал чужой код, то не стесняйтесь его использовать (с согласия хозяина!). Не выдумывайте то, что уже сделано и обкатано другими пользователями. Если каждый будет изобретать колесо, то никто и никогда не создаст повозку, и тем более автомобиль.

7. Хакеры — не просто отдельные личности, а целая культура. Но это не значит, что все хакеры одеваются одинаково и выглядят на одно лицо. Каждый из них — отдельный индивидуум и не похож на других. Не надо копировать другого человека. Удачное копирование не сделает вас "продвинутым" хакером. Только ваша индивидуальность поможет создать вам имя.

   Если вы известны в каких-либо кругах, то это считается очень почетным. Хакеры — это люди, добывающие себе славу своими познаниями и добрыми делами.

   Как вам определить, являетесь ли вы хакером? Очень просто: если о вас говорят как о хакере, то вы один из них. Жаль, что такого добиться очень сложно, потому что большинство людей считает хакерами взломщиков,

а не созидателей. Поэтому, чтобы о вас заговорили как о хакере, нужно что-то вскрыть. Но это неправильно, и не надо поддаваться этому соблазну. Старайтесь держать себя в рамках дозволенного и добиться славы только хорошими делами (если вы стремитесь к ней). Это намного сложнее, но что поделаешь… Никто и не обещал, что будет легко.

8. Чем отличаются друг от друга программист, пользователь и хакер? Программист, когда пишет программу, видит, какой она должна быть, и все делает на свое усмотрение. Пользователь не всегда знает, что задумал программист, и работает с программой так, как понимает.

   Программист не всегда может предугадать действия своих клиентов, да и приложения не всегда тщательно протестированы. Пользователи имеют возможность ввести параметры, которые приводят к неустойчивой работе программ.

   Хакеры намеренно ищут в программе лазейки, чтобы заставить ее работать неправильно, нестабильно или необычно. Для этого требуется воображение и нестандартное мышление. Вы должны чувствовать исполняемый код и видеть то, чего не видят другие.

   Для тестирования программ приглашают тестеров, которые специализируются в поисках проблемных мест и знают, как пользователи могут повести себя и как хакеры могут искать уязвимости.

   Если вы нашли какую-то уязвимость, то необязательно ее использовать. Об ошибках лучше сообщать владельцу системы (например, администрации сайта). Это весьма благородно, а главное, — создаст вам имя, и при этом можно не опасаться оказаться в зале суда. Хотя те, кто попадает под суд, быстрее получают популярность, потому что о таких людях пишут в газетах. Но кому в тюрьме нужно признание общественности? Я думаю, что никому. Тем более что после отбывания срока наказания очень часто тяжело найти себе работу. Мало кто захочет держать в штате бывшего преступника, да и после пребывания в местах не столь отдаленных может еще долго действовать запрет на работу, связанную с любимыми компьютерами. Лучше быть здоровым и богатым на свободе, чем знаменитым, но в тюрьме.

Некоторые считают, что правильно надо произносить "хэкер", а не "хакер". Если подойти к этому вопросу строго, то оба варианта неверны, потому что второй звук в слове hAcker произносится как нечто-то среднее между русскими А и Э. В русском алфавите просто нет буквы для такого звука. У нас в стране это слово обрусело и стало "хакером".

Хакеры Web-сайтов должны отлично знать следующее:

- ОС, которая будет использоваться в качестве платформы. Наибольшей популярностью в Web пользуются серверы на платформе UNIX, поэтому желательно знать их основные команды, особенно касающиеся работы с файловой системой;
- Web-сервер, на котором выполняются сценарии. Наиболее популярным сервером в настоящее время является Apache. Именно он работает на большинстве серверов в Интернете;
- базу данных. Использовать в этих целях файловую систему не очень эффективно, особенно при большом количестве данных. Базы данных дают нам большие возможности. Как и любая технология, они не гарантируют абсолютную безопасность, но к ней нужно стремиться. Вы должны понимать работу сервера (в Web чаще всего это MySQL) и знать язык SQL (Structured Query Language, язык структурированных запросов), который используется для доступа к данным. Именно тщательному изучению SQL удалось изобрести атаку SQL Injection (внедрение SQL), которая является очень опасной;
- язык Web-программирования, особенно его слабые и сильные стороны. В данном случае мы рассматриваем язык программирования PHP, который становится одним из самых популярных и не без оснований.

У нас получился список из четырех пунктов. Выпишем названия конкретных продуктов, которые могут быть использованы в качестве средств реализации:

- Linux — самая популярная ОС для Web-серверов;
- Apache — самый популярный Web-сервер;
- MySQL — бесплатная и очень мощная база данных;
- PHP — популярный язык программирования.

Если взять первые буквы этих названий, то получится сокращение LAMP. Это слово уже вошло в лексикон большинства Web-программистов и обозначает четыре программных продукта, которые используются для построения Web-сайтов. Даже в объявлениях о приеме на работу можно часто встретить выражение типа "знание LAMP", т. е. кандидат на работу должен знать все четыре программы.

1.3. Что такое PHP?

Язык PHP (Personal Home Page Tools, инструменты персональных домашних страниц) — это язык сценариев с открытым исходным кодом, встраиваемых в HTML-код и выполняемых на Web-сервере. Этот язык написан Web-разработчиками и для Web-разработчиков. Язык PHP является конкурентом таких продуктов, как Microsoft Active Server Pages (ASP), Macromedia ColdFusion

и Sun Java Server Pages. Некоторые специалисты называют PHP "открытым языком ASP" или "ASP с открытым исходным кодом". Это неверно, потому что PHP разрабатывался на несколько лет раньше, примерно в одно и то же время с Java Server Pages, поэтому можно сказать, что ASP является закрытой альтернативой для PHP. А если уж сравнивать PHP и современную версию ASP (ASP.NET), то в них совершенно ничего похожего нет.

Сам по себе Web-сервер не умеет выполнять сценарии PHP, для этого необходима программа-интерпретатор. Такие интерпретаторы существуют для всех популярных Web-серверов (IIS, Apache) на всех основных платформах (Windows, Linux и т. д.).

Язык PHP является официальным модулем Apache Web Server. Это бесплатный Web-сервер, который является лидером и используется более чем на половине серверов в Интернете (точную цифру назвать сложно, но любые данные указывают на превосходство данного сервера). Что значит официальный модуль? Это значит, что движок обработки PHP-сценариев может быть встроен в Web-сервер, что позволяет ускорить выполнение и улучшить управляемость памятью. Сервер Apache существует для всех основных платформ — Windows, Mac OS X и основные разновидности UNIX-систем — и на любой платформе эффективно работает с PHP.

Язык PHP позволяет встраивать фрагменты кода непосредственно в HTML-страницы, а интерпретированный код вашей страницы отображается пользователю. Код на языке PHP можно воспринимать как расширенные теги HTML, которые выполняются на сервере, или как маленькие программы, которые выполняются внутри страниц, прежде чем будут отправлены клиенту. Все, что делает код программы, незаметно для пользователя.

Язык PHP позволяет соединяться с популярными базами данных, расположенными на сервере, и обрабатывать информацию из таблиц (изменять, добавлять, удалять данные). Несмотря на то, что работать можно с разными базами, наиболее удобно и эффективно реализован интерфейс с сервером. Это делает язык PHP очень мощным при создании корпоративного сайта, содержащего множество данных. Да и любая домашняя страница уже немыслима без централизованного хранилища данных.

Практически ни один более-менее крупный Web-сайт не может работать без хранилища данных. Для решения этой задачи можно использовать текстовые файлы на сервере или базы данных (второе намного удобнее при обработке). В данной книге мы будем рассматривать работу именно баз данных. В качестве основной будет использоваться самая распространенная разновидность — MySQL. Это реляционная база данных с открытым исходным кодом, которая проста в использовании и поддерживается большинством хостинговых компаний.

1.4. Как работает PHP?

Что значит "встраиваемый" язык? Рассмотрим простой пример кода Web-страницы, в которой используются PHP-инструкции (листинг 1.1).

Листинг 1.1. Код страницы с PHP-инструкциями

```
<html>
<head>
<title> test page </title>
</head>

<body>
<?php
$title='We are glad to see you again';
?>
<p>hello. <?php echo $title ?>
<p>current time <?php echo date('y-m-d h:i:s') ?>
</body>
<html>
```

ПРИМЕЧАНИЕ

Исходный код из листинга 1.1 вы можете найти в файле \Chapter1\embadded.php на компакт-диске, прилагаемом к книге.

Инструкции PHP по умолчанию заключаются в тег `<?php … ?>`. Мы пока не будем вникать в код, показанный здесь, потому что сейчас главная задача — уяснить принцип работы. Уже после чтения следующей главы вы поймете написанный здесь код. Если теперь загрузить эту страничку с Web-сервера, то вы должны увидеть примерно то, что изображено на рис. 1.1.

Давайте теперь посмотрим исходный код страницы в браузере. Для этого выберите меню **Вид | В виде HTML** (**View | Source**). Перед вами откроется окно Блокнота, в котором будет содержаться примерно следующий код:

```
<HTML>
<HEAD>
<TITLE> Test page </TITLE>
</HEAD>

<BODY>
<p>Hello. We are glad to see you again
<p>Current time 2004-09-30 13:16:42
</BODY>
<HTML>
```

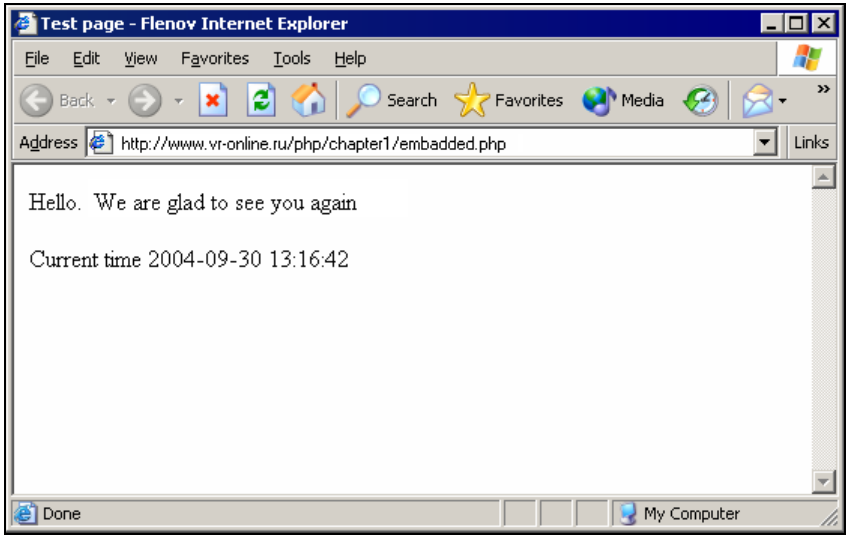

Рис. 1.1. Просмотр страницы, в которой работает код PHP

Как видите, никаких PHP-инструкций больше нет. Все они исчезли. Это связано с тем, что сервер обработал наши команды и передал пользователю чистый HTML-код, или можно сказать, что пользователь видит только результат работы сценария. Когда пользователь запрашивает страницу, сервер обрабатывает все PHP-инструкции на этой странице и возвращает только результат обработки на чистом HTML. Это дает следующие преимущества:

- любой браузер или устройство может правильно отобразить результат, если результирующий HTML-код соответствует его правилам;
- запросы выполняются быстро и используют минимальное количество ресурсов;
- код страницы состоит из HTML-тегов, а значит, вы можете легко создавать макеты с помощью простого в использовании языка разметки HTML;
- можно создавать Web-страницы в любом редакторе (даже визуальном), а потом только добавлять в них PHP-код.

Таким образом, мы создаем простую HTML-страницу, в которой присутствует код PHP. В некоторых языках, схожих по назначению с PHP (например, Perl), происходит обратное. Там вы пишете код программы, а для добавления HTML-тегов нужно писать специальные инструкции. В данном случае все наоборот (хотя можно выводить код HTML и с помощью функций, как в Perl). Часто это бывает очень удобно, а встречаются страницы, полностью написанные на PHP.

Язык PHP является интерпретируемым. Это значит, что его не надо компилировать в программу (хотя можно и откомпилировать), а выполнение происходит "на лету". Это очень удобно, но иногда может приводить к издержкам, например, если какой-то фрагмент кода выполняется в программе 100 раз. Каждый раз будет происходить интерпретация, что отнимает лишнее процессорное время. Это действительно проблема интерпретируемых языков, но в PHP решили эту проблему компиляцией "на лету". Такой фрагмент кода будет интерпретироваться только один раз.

Еще один недостаток интерпретируемости — такие программы поставляются в исходных кодах, и любой программист сможет увидеть ваш труд и использовать в своих целях. Но это только если вы будете распространять свои сценарии. Пользователи вашего сайта не смогут увидеть исходный код, потому что в браузер попадает только результат или HTML-документ. Если вы хотите защитить свою собственность, то можете воспользоваться принудительной компиляцией.

1.5. Серверные и клиентские технологии

В настоящее время существует множество клиентских и серверных технологий для построения Web-страниц. Клиентские технологии выполняются в браузере (JavaScript, VBScript, Java-апплеты, DHTML и т. д.), а серверные обрабатывает сервер и возвращает клиенту только HTML-код (Perl, ASP, PHP). Язык PHP не ограничивает вас и позволяет с легкостью использовать клиентские технологии совместно с инструкциями PHP. Но стоит ли их использовать без особой надобности? Я думаю, что нет, и это мы сейчас увидим.

Рассмотрим клиентскую технологию на примере JavaScript. Если вы будете использовать этот код в своих проектах, то нет гарантии, что страница будет отображена в любом Web-браузере. Некоторые не поддерживают эту технологию, а там, где есть поддержка, пользователи иногда отключают JavaScript в целях безопасности. Таким образом, ваша страница может отображаться некорректно, и это вызовет лишние проблемы у посетителей.

Не стоит использовать JavaScript, если он не принесет реальной выгоды. Намного лучше будет возложить выполнение этих операций на сервер, и тогда ваш сайт будет правильно отображаться в любом браузере.

Клиентские технологии не могут соединяться с базами данных и формировать HTML-код для удобного отображения и восприятия информации. Они, скорее, предназначены для придания сайту привлекательности. Серверные технологии используются для динамического создания страниц и отображения их пользователю. Как мы уже знаем, эта работа невидима для пользователя.

1.6. Установка PHP

Прежде чем приступить к программированию на PHP, нужно обзавестись необходимыми средствами для разработки и тестирования. Для написания PHP-кода можно использовать даже простой Блокнот, хотя специализированная среда разработки будет намного удобнее.

Для тестирования описываемых примеров необходим Web-сервер, который будет обрабатывать наши запросы, и программа PHP, которая будет обрабатывать сценарии. Если у вас выделенная линия в Интернете и неограниченный трафик, то можно воспользоваться услугами компании, предоставляющей услуги Web-хостинга. Таких компаний сейчас достаточно много и большинство из них поддерживает PHP, но чаще всего эта услуга платная, и цена зависит от списка предоставляемых услуг и размера выделяемого дискового пространства.

Помимо PHP нам еще понадобится база данных MySQL, потому что в некоторых будущих примерах мы будем использовать ее для своих проектов. Эта база данных является наиболее распространенной в Интернете, и ее также поддерживает большинство компаний, предоставляющих Web-хостинг.

Но если у вас есть проблемы с интернет-соединением или трафиком, то намного выгоднее будет использовать во время разработки локальные версии Web-сервера, PHP и MySQL. Тогда после завершения разработки вам достаточно будет закачать все изменения на сервер и отлаживать окончательный вариант.

В случае с разработкой на локальном компьютере тестирование может усложниться. Вы должны будете отладить и проверить на ошибки все сценарии на локальном компьютере, а потом уже на сервере. При переносе готовых файлов на сервер вы должны проверить сценарии и на безопасность. Когда вы работаете с локальной копией, то ее поведение может отличаться от серверного варианта, потому что могут быть отличия в настройках интерпретатора PHP.

Итак, прежде чем приступить к установке, вам понадобится посетить следующие три сайта:

- **http://www.php.net/downloads.php** — здесь можно найти последнюю версию PHP;
- **http://www.mysql.com/** — здесь можно найти последнюю версию базы данных MySQL;
- **http://www.apache.com/** — здесь можно найти последнюю версию бесплатного Web-сервера Apache.

Я не буду тратить много времени на описание процесса установки, потому что оба продукта легко устанавливаются на любой платформе и поставляются с достаточно подробными инструкциями по установке и настройке. Если вы работаете под Linux, то, скорее всего, все эти продукты уже встроены в дистрибутив, и надо только убедиться, что они установлены и запущены. Если нет, то можно установить PHP и MySQL с диска дистрибутива Linux или скачать последнюю версию с вышеуказанного сайта.

Если вы работаете под Windows, то в качестве Web-сервера можно использовать встроенный в Windows (идет не со всеми версиями, в версиях для предприятий, таких как Professional, Ultimate) сервер MS Internet Information Server. Он вполне прост и приемлем для тестирования ваших приложений. Если во время установки Windows вы отказались от использования IIS, то войдите в настройках системы в панель **Установка и удаление программ**, выберите **Компоненты Windows** и установите компонент **Internet Information Server**.

Под Windows установка PHP происходит при помощи простой программы. После этого он готов к использованию. Достаточно поместить файлы сценариев в папку, которая используется Web-сервером для хранения опубликованных файлов, и загрузить их в браузере. По умолчанию IIS использует папку Inetpub\wwwroot на системном диске. Если поместить файл в эту папку, то достаточно набрать в браузере адрес **http://127.0.0.1/filename.php** (filename.php — имя файла сценария), и вы увидите результат работы файла сценария.

Я рекомендую также программу Denwer (**http://denwer.ru**). Это небольшой бесплатный установочный пакет, который уже включает в себя три необходимых для локальной разработки компонента: PHP, MySQL и Apache. Установка проста и выполняется в два щелчка мышью. К тому же, сайт и программный интерфейс русскоязычные.

Лично я использую сразу три метода:

- Linux-сервер на виртуальной машине. В качестве виртуальной машины у меня работает VirtualBox, а в качестве ОС установлена Ubuntu;
- Denwer, который я запускаю для небольших тестов и небольших экспериментов. Для разработки сайтов я не люблю его использовать;
- рабочий хостинг в Интернете. Этот метод я использую чаще всего для внесения небольших изменений. Да, это не очень удачное решение, потому что любая опечатка или ошибка приводит к тому, что сайт становится недоступным, но в этом есть своя соль. Приходится быть очень аккуратным и проверять каждую строчку кода перед закачиванием на сервер.

А когда нажимаешь кнопку **Upload**, чувствуешь нервную дрожь и надеешься, что не совершил никакой ошибки. К тому же, этот метод заставляет по десять раз проверять и безопасность перед отправкой изменений на сервер.

Какой способ выберете вы, зависит от личных предпочтений. При выборе третьего варианта будьте очень осторожны.

На этом мы заканчиваем вступительную теоретическую часть и переходим непосредственно к программированию. Надеюсь, что я не сильно перегрузил ваш мозг вводными речами. Я сделал это из добрых побуждений, чтобы книга не выглядела слишком академичной и читалась легко.