ГЛАВА 19

Модель безопасности ASP.NET

Безопасность — важнейшая часть Web-приложений, и она должна приниматься во внимание с первой стадии процесса разработки. По сути, безопасность — это все, что касается защиты вашего имущество от неавторизованных действий. И для ее обеспечения используется несколько механизмов, включая идентификацию пользователей, выдачу или отъем прав доступа к важным ресурсам, а также защиту информации, хранящейся на сервере и передающейся по проводам. Во всех этих случаях вам необходим некий фундаментальный каркас, обеспечивающий базовую функциональность безопасности. ASP.NET удовлетворяет эту потребность благодаря встроенным средствам, которые вы можете использовать для обеспечения защиты ваших приложений.

Система безопасности ASP.NET включает классы для аутентификации и авторизации пользователей, а также для обращения с аутентифицированными пользователями в ваших приложениях. Более того, каркас .NET Framework сам по себе предоставляет набор базовых классов для обеспечения конфиденциальности и целостности путем шифрования и цифровых подписей.

С появлением версии ASP.NET 2.0, которая послужила основой ASP.NET 3.5, инфраструктура безопасности была существенно расширена за счет высокоуровневой модели управления пользователями и ролями, воплощенной как программно, так и встроенными инструментами администрирования. Эта функциональность (доступная через API-интерфейсы `Membership` и `Roles`) строится на базе существующей инфраструктуры безопасности, которая появилась еще в ASP.NET 1.*x*. Но лучше всего то, что эта инфраструктура безопасности является полностью расширяемой через проектный шаблон поставщика, как будет показано в главе 26. Каркас ASP.NET 3.5 пошел еще дальше, расширив эту инфраструктуру функциональностью для интеграции с Ajax, о чем вы узнаете из глав 31 и 32.

В настоящей главе предложен общий путеводитель по средствам безопасности ASP.NET. В последующих главах вы углубите свои познания по каждой теме из числа представленных здесь. А пока мы проведем краткое представление ключевых средств обеспечения безопасности .NET. Вы увидите, как структурированы поставщики аутентификации .NET и модули авторизации, и узнаете о том, как пользовательский контекст безопасности представлен идентичностью и ведущими (principal) объектами. Более важно то, что вы получите общее представление о том, как можно встроить средства безопасности в создаваемую программную архитектуру и дизайн, и увидите, какие факторы наиболее важны при создании безопасного программного обеспечения.

Что означает создание безопасного программного обеспечения

Хотя каркас безопасности, предложенный .NET и ASP.NET, достаточно мощный, все же стоит постоянно иметь в виду базовые принципы и правильно применять эти средства в нужный момент. Слишком во многих проектах забота о безопасности проявляется запоздало; архитекторы и разработчики не думают о ней на ранних стадиях проекта. Но если вы не думаете о безопасности с самого начала, а именно — при разработке дизайна и архитектуры приложения — как вы сможете правильно и вовремя применить средства защиты, предлагаемые .NET Framework?

Таким образом, важно учитывать вопросы обеспечения безопасности с первого момента работы. Это единственный способ принятия правильных решений, касающихся защиты, в процессе разработки архитектуры и дизайна.

Понятие потенциальной угрозы

Создание безопасной архитектуры и дизайна требует глубокого понимания среды вашего приложения. Вы не сможете создавать безопасное программное обеспечение, если не знаете, кто имеет доступ к вашим приложениям и где находятся уязвимые места для атак. Таким образом, наиболее важный фактор для создания безопасной программной архитектуры и дизайна заключается в хорошем понимании таких факторов среды, как пользователи, точки входа и потенциально возможные угрозы с точками для атаки.

Вот почему *моделирование угроз* становится все более важным в современном процессе разработки программного обеспечения. Моделирование угроз — это структурированный способ анализа среды ваших приложений с точки зрения возможных опасностей, их классификации и решения относительно приемов смягчения этих угроз. При таком подходе решения относительно технологий безопасности (таких как аутентификация и SSL-шифрование) всегда имеет действительное основание — потенциальную угрозу.

Однако моделирование угроз важно по еще одной причине. Как вы, возможно, знаете, не все потенциальные угрозы могут быть смягчены применением технологий защиты, такими как аутентификация и авторизация. Другими словами, некоторые из них вообще невозможно разрешить технически. Например, банковское онлайновое решение может использовать SSL для защиты трафика Web-сайта. Но как пользователи могут знать, что они действительно используют банковскую страницу, а не хакерский поддельный Web-сайт? Хорошо, единственный способ убедиться в этом — проверить сертификат, используемый для установки канала SSL. Но пользователи должны быть предупреждены об этом, а потому вы должны каким-то образом их информировать. Поэтому "техника смягчения" угроз — это не только технологии защиты. Это включает требование того, чтобы все ваши пользователи знали, как проверить сертификат. (Конечно, вы не можете заставить их это делать, но если ваша система спроектирована соответствующим образом, все же можно большинство из них стимулировать к этому.) Моделирование угроз — метод анализа, помогающий выявить обстоятельства вроде этих, а не только факторы технического порядка.

Моделирование угроз — обширная тема, которая выходит далеко за пределы настоящей книги. Если интересуетесь — обратитесь к соответствующим источникам. Среди прочих можно порекомендовать несколько дополнительных книг от команды построения шаблонов и приемов в Microsoft (`http://msdn.microsoft.com/patterns`). Эта команда создает строительные блоки приложений на основе реальных проектов, выполняемых Microsoft для реальных заказчиков. Их книга *Building Secure Microsoft ASP.NET Applications by Microsoft Corp.* (Microsoft Press, 2003 г.) раскрывает многие детали и кон-

цепции построения безопасных Web-приложений, во многих случаях не зависящих от используемых версий ASP.NET. Большая часть содержимого шаблонов и приемов доступна бесплатно также в виде электронных книг. Загляните на следующие ресурсы:

- `http://msdn2.microsoft.com/en-us/library/aa302415.aspx`
- `http://msdn2.microsoft.com/en-us/library/ms978378.aspx`
- `http://msdn2.microsoft.com/en-us/library/ms994921.aspx`

Издательство Apress внесло собственный вклад в описание подробностей безопасности ASP.NET, издав книгу *Pro ASP.NET 2.0 Security by Russ Basiura* (Apress, 2007 г.). И, наконец, стоит упомянуть еще один справочник, который, по нашему мнению, чрезвычайно важен для менеджеров проектов и архитекторов — *The Security Development Lifecycle* Майкла Ховарда (Michael Howard) и Стива Липнера (Steve Lipner) (Microsoft Press, 2006 г.). Эта книга сосредоточена на обеспечении безопасности как неотъемлемой части вашего цикла разработки программного обеспечения — начиная с первоначального планирования, включая архитектуру, разработку, тестирование и сопровождение. В ней показано, как управление проектами Microsoft обеспечивает безопасность в качестве неотъемлемой части проекта, причем гладким и прагматичным способом. Поскольку на прочтение этой книги не потребуется много времени, мы рекомендуем прочесть ее даже тем, кто не является менеджером проектов или архитектором.

Правила безопасного кодирования

Разумеется, только безопасная архитектура и дизайн не могут сделать ваше приложение абсолютно защищенным. Это лишь один из наиболее важных факторов. После того, как вы разработали безопасную архитектуру и дизайн, вы должны также написать безопасный код. При написании кода Web-приложений вы всегда должны иметь в виду следующие правила.

- *Никогда не доверяйте пользовательскому вводу.* Предполагайте, что каждый пользователь — злоумышленник, пока он не докажет обратное. Таким образом, всегда строго проверяйте пользовательский ввод. Разрабатывайте свой код проверки достоверности так, чтобы он проверял ввод только правильных значений, а не неправильных (неправильных значений всегда больше, чем вы можете себе представить во время разработки приложения).
- *Никогда не используйте конкатенацию строк для формирования операторов SQL.* Всегда применяйте параметризованные операторы, чтобы ваше приложение не было уязвимо для атак внедрением SQL, как было описано в главе 7.
- *Никогда не выводите данные, введенные пользователем, на Web-страницу перед их проверкой и кодированием.* Пользователь может ввести некоторые фрагменты кода HTML (например, сценарий), которые инициируют межсайтовую сценарную уязвимость. Поэтому всегда используйте `HttpUtility.HtmlEncode()` для защиты специальных символов вроде < или > перед выводом их на страницу или применяйте Web-элемент управления, который выполняет такое кодирование автоматически.
- *Никогда не размещайте важные данные, критичную для бизнеса информацию либо данные, касающиеся внутренних правил безопасности, в скрытых полях вашей Web-страницы.* Скрытые поля могут быть легко изменены простым просмотром исходного кода Web-страницы, модификацией и сохранением в файле. Затем злоумышленник просто может отправить локальную модифицированную копию Web-страницы на сервер. Подключаемые модули браузеров могут сделать такой подход столь же простым, как отправка электронной посты в Microsoft Outlook.

- *Никогда не сохраняйте важные или критичные для бизнеса данные в состоянии представления.* Состояние представления — это просто еще одно скрытое поле на Web-странице, и оно может быть легко декодировано и просмотрено. Если вы используете установку `EnableViewStateMAC=true` для своей страницы, то состояние представления будет подписано кодом аутентификации сообщений, созданном на базе ключа машины, находящегося в серверном файле `machine.config`. Мы рекомендуем использовать `EnableViewStateMAC=true` немедленно после включения данных в ваше состояние представления, которое не должно быть изменено пользователями, просматривающими вашу Web-страницу. Подробнее о защите состояния представления читайте в главе 6.
- *Включайте SSL при использовании Basic-аутентификации или аутентификации форм ASP.NET.* Аутентификация форм описана в главе 20. Об SSL мы поговорим позднее в настоящей главе, в разделе "Что такое SSL".
- *Защищайте свои cookie-наборы.* Всегда защищайте свои cookie-наборы аутентификации при использовании аутентификации форм и устанавливайте таймауты насколько возможно короткими, и не длиннее, чем это действительно необходимо.
- *Применяйте SSL.* В общем случае, если ваше Web-приложение обрабатывает важные данные, защищайте весь Web-сайт с помощью SSL. Не забывайте защищать даже каталоги с графическими изображениями или другими файлами, которые не управляются приложением напрямую через SSL.

Конечно, это лишь несколько общих важных моментов. Чтобы получить общую картину ситуации для вашего конкретного приложения, вам необходимо разработать модель угроз, чтобы составить полный список потенциальных опасностей. В дополнение вкладывайте средства в постоянное обучение разработчиков, потому что хакерские приемы и технологии не стоят на месте, как и все прочие технологии.

Если вы пренебрегаете хотя бы одним из приведенных правил, то все прочие средства защиты становятся в большей или меньшей мере бесполезными. Никогда не забывайте следующий принцип: система защиты ровно настолько надежна, насколько надежна ее самая слабая часть.

Понятие стража

Хороший способ повысить степень безопасности вашего приложения — размещать компоненты в таком месте, которое требует защиты. Концептуальный шаблон *стража* (gatekeeper) применяет модель конвейера к организации инфраструктуры безопасности. Эта модель помогает укрепить безопасность.

Модель стражей предполагает, что безопасное приложение всегда имеет больше механизмов защиты, чем это необходимо. Каждый из этих механизмов реализован как страж, отвечающий за проверку некоторых условий защиты. Если один из таких стражей не сработает, то атакующий столкнется со следующим стражем в конвейере. И чем больше стражей имеется в вашем приложении, тем труднее приходится злоумышленнику. На самом деле эта модель отвечает центральному принципу создания безопасных приложений: обеспечивать насколько возможно высокую степень защиты и создавать максимум проблем нарушителям.

На рис. 19.1 показан конвейер стражей. В конце этого конвейера можно видеть защищенный ресурс (которым может быть что угодно, даже ваш собственный код страницы). Защищенный ресурс будет доступен или выполнен только в том случае, если все стражи откроют к нему доступ. Если хотя бы один из них откажет в доступе, обработка запроса будет прекращена и клиенту отправляется исключение безопасности.

Реализация центрального механизма безопасности в такой манере — вообще хорошая идея. Точно так же вы можете защитить бизнес-уровень своего приложения. Инфраструктура приложений ASP.NET также применяет этот механизм. ASP.NET включает несколько стражей, каждый из которых проверяет несколько условий, таким образом, защищая ваше приложение. В последующих разделах настоящей главы вы познакомитесь со стражами, которые включает в себя каркас ASP.NET, и с зоной ответственности каждого из них.

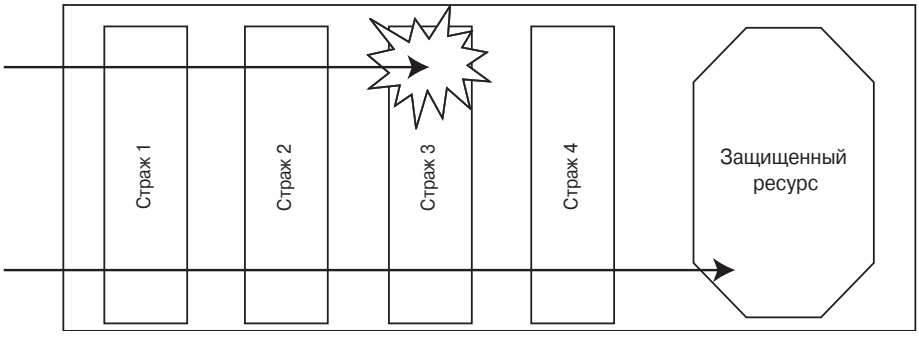

Рис. 19.1. Конвейер стражей

Понятие уровней безопасности

В основном для большей части Web-приложений основные задачи для реализации защиты (помимо тех, что вы идентифицируете во время моделирования угроз) всегда одни и те же.

- *Аутентификация.* Прежде всего, вы должны аутентифицировать пользователей. Аутентификация задает вопрос: кто идет? В конечном итоге она определяет, кто работает с вашим приложением на другой стороне.
- *Авторизация.* Далее, как только вы узнали, кто работает с вашим приложением, ваше приложение должно решить, какие операции данный пользователь может выполнять и к каким ресурсам обращаться. Другими словами, авторизация отвечает на вопрос: каков ваш уровень допуска?
- *Конфиденциальность.* Когда пользователь работает с приложением, вы должны гарантировать, что никто другой не сможет видеть важные данные, которые он обрабатывает. Таким образом, вы должны шифровать канал между браузером клиента и Web-сервером. Более того, возможно, вам придется шифровать данные, сохраняемые в базе данных (или в форме cookie-наборов на стороне клиента), чтобы даже администратор базы данных или другой персонал вашей компании не мог видеть эти данные.
- *Целостность.* И, наконец, вы должны гарантировать, что данные, передаваемые между клиентом и сервером, не изменяются в результате неавторизованного вмешательства. Цифровые подписи позволяют снизить уровень этой угрозы.

ASP.NET включает базовую инфраструктуру для выполнения аутентификации и авторизации. Библиотека базовых классов .NET Framework включает некоторые классы в пространстве имен `System.Security`, предназначенные для шифрования и подписи данных. Более того, SSL — стандартизованный способ обеспечения конфиденциальности и целостности данных, передаваемых между клиентским браузером и Web-сервером. Теперь мы рассмотрим подробнее каждую из этих концепций.

Аутентификация

Аутентификация — процесс определения идентичности пользователя и обеспечения гарантий этой идентичности. Процесс аутентификации аналогичен регистрации участников конференции. Во-первых, вы предъявляете некоторое свидетельство, доказывающее вашу идентичность (вроде паспорта или водительских прав). Во-вторых, как только ваша идентичность проверена по этой информации, вы получаете личный значок участника конференции, или *маркер*, который постоянно носите с собой на протяжении всей конференции. Любой, кто вас встретит на конференции, сможет легко определить вашу идентичность, взглянув на этот значок, который обычно содержит базовую идентифицирующую информацию, такую как ваше имя и фамилия. Весь этот процесс — пример аутентификации. Как только ваша идентичность установлена, ваш маркер подтверждает ее, так что куда бы вы ни пошли в пределах конкретной области, ваша личность известна.

В приложениях ASP.NET аутентификация реализуется одной из четырех возможных аутентифицирующих систем:

- аутентификация Windows;
- аутентификация с помощью форм;
- аутентификация с помощью паспортов;
- специальный процесс аутентификации.

В каждом случае пользователь предъявляет некоторое удостоверение при регистрации в системе. Идентичность пользователя отслеживается разными способами, в зависимости от типа аутентификации. Например, операционная система Windows использует 96-битное число, называемое SID (security identifier — идентификатор безопасности) для идентификации каждого входящего пользователя. В аутентификации форм ASP.NET (которая подробно описывается в главе 20) пользователю выдается аутентифицирующий мандат формы, представляющий собой комбинацию значений, которые шифруются и помещаются в cookie-набор.

Вся аутентификация позволяет приложению идентифицировать, какой пользователь присылает каждый запрос. Это хорошо работает для персонализации и пользовательской настройки, потому что вы можете использовать идентифицирующую информацию для выдачи специфичных для пользователя сообщений на Web-странице, изменять внешний вид Web-сайта, добавлять специальное содержимое на базе предпочтений конкретного пользователя и тому подобное. Однако аутентификации самой по себе недостаточно для ограничения задач, которые разрешено выполнять пользователю на базе его идентичности. Для этого нужна авторизация, о которой мы поговорим ниже. Однако прежде чем перейти к авторизации, следует взглянуть на такую вещь, как заимствование прав, которая тесно связана с аутентификацией.

Заимствование прав

Заимствование прав (impersonation) — процесс выполнения кода в контексте (или от имени) другого пользователя. По умолчанию код ASP.NET исполняется от имени фиксированной, специфичной для конкретной машины, пользовательской учетной записи (обычно ASP.NET на IIS 5.x или Network Service на IIS 6.0 и IIS 7.0). Чтобы выполнить код, применяя другую идентичность, можно воспользоваться встроенными в ASP.NET возможностями заимствования прав. Можно воспользоваться предопределенной пользовательской учетной записью либо предположить пользовательскую идентичность, если пользователь уже был аутентифицирован с применением учетной записи Windows.

Вы можете использовать заимствование прав по двум причинам.

- *Чтобы выдать каждому Web-приложению разные права.* В IIS 5 для выполнения всех Web-приложений на компьютере используется учетная запись по умолчанию, указанная в файле `machine.config`. Если вы захотите предоставить разным Web-приложениям разные права, то можете применить заимствование прав для назначения разных учетных записей Windows каждому приложению. Это особенно важно для сценариев хостинга, когда нужно соответствующим образом изолировать Web-приложения разных заказчиков (так, чтобы, например, Web-приложение заказчика A не могло получить доступ к каталогам или базам данных Web-приложения заказчика B).
- *Чтобы использовать существующие права доступа пользователя Windows.* Например, представим себе приложение, которое извлекает информацию из различных файлов, для которых уже установлены специфичные для пользователей и групп наборы прав доступа. Вместо того чтобы кодировать логику авторизации в приложении ASP.NET, можно использовать заимствование прав для установки идентичности текущего пользователя. Таким образом, Windows выполнит авторизацию для вас, проверив права доступа, как только вы попытаетесь обратиться к файлу.

Эти два сценария принципиально различны. В первом из них заимствование прав определяет одну специфическую учетную запись. В этом случае не важно, какой пользователь обращается к приложению, и не важно, какого рода защита уровня пользователя применяется — код будет запущен от имени той учетной записи, которую вы установили. Во втором сценарии пользователи должны быть аутентифицированы IIS. Код Web-страницы будет выполнен с идентичностью соответствующего пользователя. Подробнее об этих опциях вы прочтете в главе 22.

Авторизация

Авторизация — процесс определения прав и ограничений, назначенных аутентифицированному пользователю. Если взять аналогию с конференцией, то авторизация — это процесс выдачи допуска на определенные мероприятия, например, на главный доклад. На большинстве конференций можно подписаться на разные уровни доступа — на полный доступ, только вступительное заседание или только на посещение выставочного зала. Это значит, что если вы захотите посетить главное заседание "Конференции профессиональных разработчиков Microsoft", чтобы послушать, что скажет Билл Гейтс, то должны для этого иметь соответствующие права доступа. Как только вы войдете в зал заседаний, сотрудник службы безопасности проверит вашу эмблему участника конференции. На основании указанной на ней информации он либо пропустит вас, либо скажет, что вы не можете войти. Это пример авторизации. В зависимости от идентифицирующей вас информации, вам либо открывается, либо закрывается доступ к запрашиваемым ресурсам.

Пример с конференцией представляет случай авторизации на базе ролей — когда авторизация определяется правами группы, к которой принадлежит пользователь, а не на том, кто он такой. Другими словами, вы авторизуетесь для доступа в зал заседаний на основании роли (типа допуска), а не вашей специфичной идентифицирующей информации (имени и фамилии). Во многих случая авторизация на базе ролей предпочтительна, поскольку ее гораздо легче реализовать. Если сотрудник безопасности должен будет сверять имя каждого гостя со списком допущенных, то процесс авторизации существенно замедлится. То же самое верно и для Web-приложений, хотя более вероятно, что роли будут следующими: менеджеры, администраторы, гости, продавцы, клиенты и т.д.

В Web-приложениях разные типы авторизации происходят на разных уровнях. Например, на самом верхнем уровне ваш код может проверять идентичность пользователя и решать, можно ли продолжать данную операцию. На нижнем уровне можно настроить ASP.NET так, чтобы запрещать доступ к определенным Web-страницам или каталогам для определенных пользователей или ролей. На еще более низком уровне, когда ваш код выполняет различные задачи — такие как подключение к базе данных, открытие файла запись в протокол событий и тому подобное — операционная система Windows проверяет права учетной записи Windows, под именем которой выполняется данный код. В большинстве ситуаций вы не захотите полагаться на этот самый нижний уровень, поскольку ваш код всегда будет выполняться от имени фиксированной учетной записи. В IIS 5.x эта учетная запись называется ASPNET. В IIS 6.0 и IIS 7.0 — это по умолчанию фиксированная учетная запись Network Service (в обоих случаях вы можете переопределить учетную запись по умолчанию, как описано в главе 18).

Существует несколько причин применения фиксированной учетной записи для запуска кода ASP.NET. Почти во всех приложениях права, выданные пользователю, не соответствуют правам, которые требуются приложению, работающему от имени пользователя. Как правило, ваш код нуждается в более широком наборе привилегий, чтобы выполнять задачу идентификации, и вы не захотите выдавать такие права каждому пользователю, который может обращаться к вашему приложению. Например, вашему коду может понадобиться создавать журнальные записи о возможных сбоях, даже если данному пользователю не разрешено напрямую писать в журнал событий Windows, в файл или в базу данных. Аналогично приложения ASP.NET всегда должны иметь право доступа в каталог `c:\[WinDir]\Microsoft.NET\[Version]\Temporary ASP.NET Files`, чтобы создавать и кэшировать версии ваших Web-страниц на машинном языке. Обратите внимание, что для приложений ASP.NET 3.5 вы все еще найдете эту папку в подкаталоге версии внутри каталога Microsoft.NET для версии 2.0! Это объясняется тем, что ASP.NET 3.5 — это расширение, построенное на основе ASP.NET 2.0, а не полностью новая версия ASP.NET. И, наконец, вы можете пожелать использовать систему аутентификации, которая вообще никак не взаимодействует с Windows. Например, приложения электронной коммерции могут проверять адреса электронной почты пользователей по базе данных серверной стороны. В этом случае идентичность пользователя никак не соответствует пользовательской учетной записи Windows.

В некоторых редких случаях вы можете предоставить вашему коду временно предполагать идентичность пользователя. Такой подход чаще всего используется при создании приложений ASP.NET для локальных сетей, в которых пользователи уже имеют четко определенные наборы привилегий Windows. В этом случае вам придется пополнить свой арсенал средств безопасности заимствованием прав, как упоминалось в предыдущем разделе и описано в главе 22.

Конфиденциальность и целостность

Конфиденциальность означает обеспечение невидимости данных для неавторизованных пользователей во время передачи их по сети или сохранении в хранилищах, таких как базы данных. *Целостность* — это обеспечение невозможности изменения данных никем во время передачи по сети или сохранения в хранилище. И то, и другое основано на шифровании.

Шифрование — процесс кодирования данных, делающий невозможным их прочтение другими пользователями. Шифрование в ASP.NET — полностью отдельное средство от аутентификации, авторизации и заимствования прав. Вы можете применять его в комбинации с этими средствами либо самостоятельно.

Как уже упоминалось ранее, шифровать Web-приложения может потребоваться по двум причинам.

- *Для защиты коммуникаций (передачи данных по проводам).* Например, вы хотите сделать невозможной кражу номеров кредитных карт, используемых в вашей системе электронной коммерции, по открытым каналам Internet. Стандартный подход к этой проблеме состоит в применении SSL. SSL также реализует цифровые подписи для обеспечения гарантии целостности. SSL не реализуется ASP.NET. Это средство, предоставляемое IIS. Код вашей Web-страницы (или Web-службы) не зависит от того, применяется SSL или нет.
- *Для защиты постоянной информации (в базе данных или в файле).* Например, вы можете пожелать сохранить номер кредитной карточки пользователя в базе данных для будущего использования. Хотя вы можете сохранять эти данные в простом тексте и надеяться, что Web-сервер не будет взломан, однако это плохая идея. Вместо этого вам следует использовать классы шифрования, которые предлагает .NET, чтобы вручную шифровать данные перед их сохранением.

Не важно, что классы .NET, выполняющие шифрование, не связаны напрямую с ASP.NET. Фактически вы можете применять их в любых приложениях .NET. Подробнее о шифровании и цифровых подписях, а также управлении настраиваемым шифрованием будет рассказано в главе 25.

Связываем все вместе

Итак, как же заставить работать вместе аутентификацию, авторизацию и заимствование прав в Web-приложении?

Когда пользователи впервые посещают Web-сайт, они анонимны. Другими словами, ваше приложение не знает (и не заботится о том), кто они такие. Если только вы не аутентифицируете их, все так и останется.

По умолчанию анонимные пользователи могут обращаться к любой странице ASP.NET. Но когда пользователь запрашивает Web-страницу, анонимный доступ к которой закрыт, выполняется несколько шагов (показанных на рис. 19.2).

1. Запрос отправляется Web-серверу. Поскольку идентичность пользователя в этот момент не известна, ему предлагается зарегистрироваться, используя специальную Web-страницу или диалоговое окно регистрации браузера. Специфические детали процесса регистрации зависят от типа используемой аутентификации.

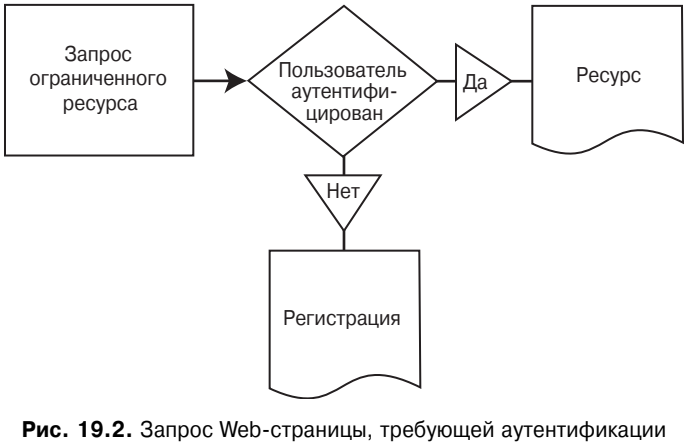

Рис. 19.2. Запрос Web-страницы, требующей аутентификации

2. Пользователь предоставляет свое удостоверение, которое затем верифицируется — либо вашим приложением (в случае аутентификации с помощью форм), либо автоматически средствами IIS (в случае аутентификации Windows).
3. Если удостоверение пользователя подтверждается, ему предоставляется доступ к Web-странице. Если же оно оценивается как нелегитимное, ему предлагается повторить попытку регистрации, либо же выполняется переадресация на страницу с сообщением о закрытии доступа.

Когда пользователь запрашивает защищенную Web-страницу, которая открыта только для определенных пользователей или ролей, процесс аналогичен, но добавляется дополнительный шаг (рис. 19.3).

1. Запрос отправляется Web-серверу. Поскольку идентичность пользователя в этот момент не известна, ему предлагается зарегистрироваться, используя специальную Web-страницу или диалоговое окно регистрации браузера. Специфические детали процесса регистрации зависят от типа используемой аутентификации.
2. Пользователь предъявляет свое удостоверение, которое проверяется приложением. Это стадия аутентификации.
3. Удостоверение или роли аутентифицированного пользователя сравниваются со списком разрешенных пользователей и ролей. Если пользователь присутствует в списке, ему открывается доступ к ресурсу; в противном случае доступ закрыт.
4. Пользователи, которым отказано в доступе, либо приглашаются на повторную регистрацию, либо перенаправляются на Web-страницу с сообщением о закрытии доступа.

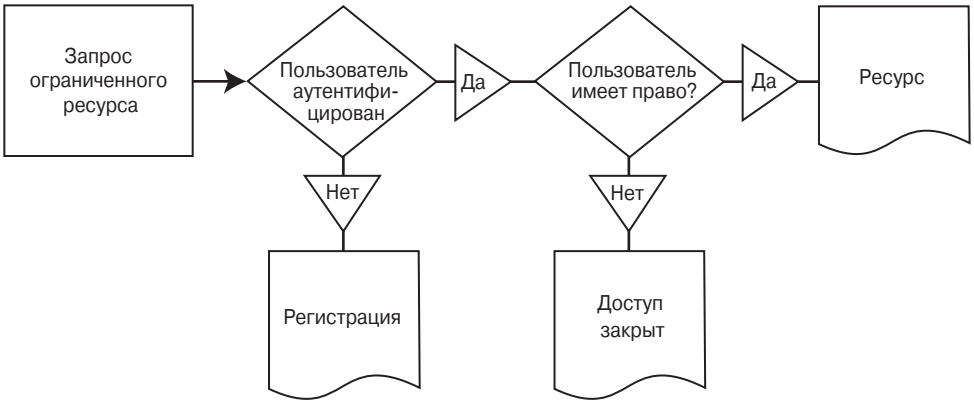

Рис. 19.3. Запрос Web-страницы, требующей аутентификации и авторизации

Средства безопасности Internet Information Services

Прежде чем исполняющей системы ASP.NET даже коснется входящий запрос, IIS проверяет безопасность в соответствии со своей собственной конфигурацией. Таким образом, перед тем, как изучать подробности средств безопасности ASP.NET, вы должны познакомиться с первым стражем в конвейере безопасности вашего Web-приложения — Internet Information Services (IIS).

IIS предоставляет несколько важных механизмов безопасности, которые выполняют функции стражей еще до того, как ASP.NET начинает обработку запросов. В основном, речь идет о перечисленных ниже механизмах.

- *Аутентификация.* IIS поддерживает следующие виды аутентификации: Basic (базовая), Digest (по дайджесту), Passport (по паспорту) и Windows, а также аутентификацию с помощью сертификатов по каналу SSL. Любая аутентификация IIS в конечном итоге сводится к аутентификации пользователя Windows. Таким образом, IIS поддерживает только аутентификацию пользователей Windows.
- *Авторизация.* IIS поддерживает встроенную поддержку ограничений IP-адресов и просмотр списков контроля доступа Windows ACL (Access Control Lists — списки контроля доступа; являются принятым в Windows способом защиты ресурсов, управляемых операционной системой, таких как файлы и папки файловой системы, элементы реестра, именованные каналы и т.д.).
- *Конфиденциальность.* Шифрование может быть обеспечено средствами SSL.

В последующих разделах вы ознакомитесь с деталями настройки безопасности IIS. Вы ознакомитесь с деталями опций конфигурации, касающимися безопасности для IIS 5.x и IIS 6.0, а также для IIS 7.0. Мы сконфигурируем установки безопасности для аутентификации, авторизации и конфиденциальности в точности одинаковым способом в IIS 5.x и IIS 6.0. В IIS 5.x и IIS 6.0 вы всегда должны помнить о безопасности IIS, поскольку она оказывает влияние на поведение ASP.NET при разных настройках защиты, установленных в `web.config`.

Например, если ваше приложение ASP.NET желает использовать аутентификацию Windows, вы должны настроить IIS на применение либо Windows, либо аутентификации Basic (Digest). Если ваше приложение ASP.NET не желает использовать учетные записи Windows (и потому использовать собственную аутентификацию форм), вы должны настроить IIS так, чтобы он разрешал вход анонимным пользователям.

Что же касается IIS 7.0, то здесь вам следует помнить, что IIS 7.0 оснащена гораздо лучше и тесно интегрирована с ASP.NET при работе в "интегрированном" режиме, о котором шла речь в главе 18. Запуск в интегрированном режиме (принятом по умолчанию для каждого пула приложений) IIS 7.0 позволяет вам полагаться на управляемые модули HTTP ASP.NET для аутентификации и авторизации. Поэтому настройка безопасности в Web-приложениях ASP.NET на IIS 7.0 часто осуществляется всего за один шаг (хотя и не во всех случаях, как вы узнаете в последующих главах, и как уже говорилось в главе 18). Сравните этот один шаг с конфигурированием Web-приложений ASP.NET, работающих на IIS 5.x и IIS 6.0, где приходилось настраивать безопасность IIS отдельно. Как должно быть известно из главы 18, IIS 7.0 можно также запустить в "классическом" режиме для обратной совместимости. В этом случае IIS 7.0 ведет себя точно так же, как IIS 5.x и IIS 6.0. За дополнительными деталями, касающимися модулей HTTP и конвейера модулей ASP.NET обращайтесь к главе 5 и разделу "Архитектура безопасности ASP.NET" настоящей главы. В следующем разделе конфигурирование безопасности IIS рассматривается более подробно.

Аутентификация и авторизация в IIS 5.x и IIS 6.0

При работе с Windows XP вы обнаружите, что имеете дело с IIS 5.x, в то время как в среде Windows Server 2003 вам доступен IIS 6.0. Подробную информацию о различиях между этими двумя версиями ищите в главе 18.

Хотя архитектура IIS 6.0 на Windows Server 2003 существенно усовершенствована по сравнению с IIS 5.x, все базовые установки безопасности, такие как аутентификация, авторизация и даже конфиденциальность, остались прежними — в пользовательском

интерфейсе консоли управления ничего не изменилось. В следующих разделах вы узнаете о конфигурации опций аутентификации, авторизации и конфиденциальности в IIS 5.x и IIS 6.0.

Аутентификация в IIS 5.x и IIS 6.0

Как упоминалось ранее, IIS поддерживает несколько механизмов аутентификации. Любые другие конфигурационные настройки безопасности (и, следовательно, аутентификации) устанавливаются для всего Web-сайта и Web-приложения. Вы можете найти эти настройки на вкладке Directory Security (Безопасность каталога) свойств виртуальных каталогов. На рис. 19.4 показаны опции аутентификации IIS 5.x и IIS 6.0.

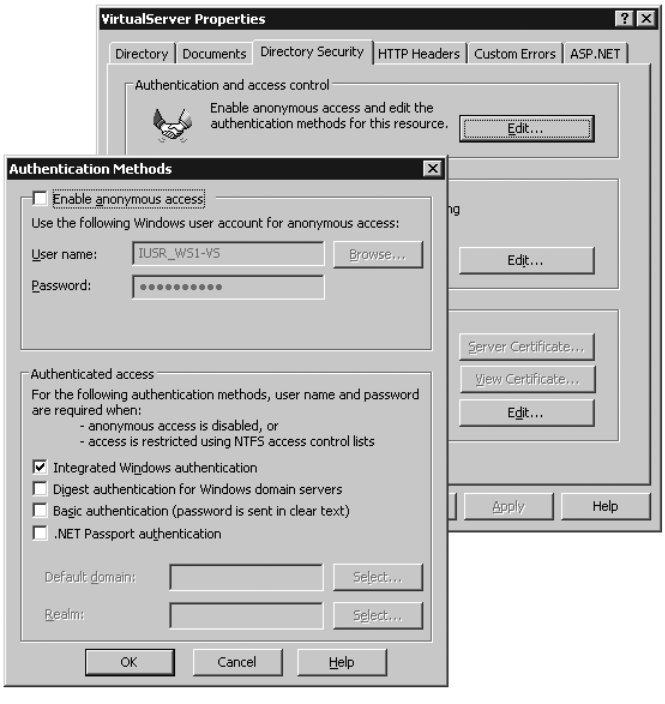

Рис. 19.4. Опции аутентификации IIS 5.x и IIS 6.0

Анонимный доступ открывает Web-страницу для всех. Он переопределяет любые другие установки аутентификации IIS, потому что IIS выполняет аутентификацию только когда это необходимо (и, конечно, если включена анонимная аутентификация, никакие дополнительные шаги не требуются). Далее, если у вас на IIS настроена анонимная аутентификация, вы можете использовать средства безопасности ASP.NET, чтобы аутентифицировать пользователей с помощью интегрированных механизмов ASP.NET, таких как аутентификация посредством форм или специальная аутентификация. Подробнее вы узнаете об этом далее в настоящей главе, а также в нескольких последующих.

Windows-аутентификация настраивает IIS на проверку удостоверений пользователей по зарегистрированным пользовательским учетным записям Windows — либо на локальной машине, либо внутри домена. Работая в домене, пользователи не обязаны вводить свои имена и пароли, если они уже зарегистрированы на клиентской машине в сети, потому что мандат клиентской аутентификации передается для аутентификации на сервер автоматически.

IIS также поддерживает Basic-аутентификацию. Это метод аутентификации, разработанный консорциумом W3C, который определяет дополнительный заголовок HTTP для передачи имен и паролей пользователей по проводам. Информация передается в кодировке Base64. Таким образом, вы должны использовать только Basic-аутентификацию с SSL. Как и в случае Windows-аутентификации, удостоверения, введенные пользователями, оцениваются по учетным записям Windows. Однако способ их передачи по проводам отличается. В то время как Basic-аутентификация передает информацию в заголовке HTTP, Windows-аутентификация использует для передачи информации либо NTLM (аббревиатура расшифровывается как Windows NT LAN Manager и, как вы узнаете ниже, представляет собой ответственный протокол аутентификации), либо Kerberos.

Аутентификация Digest подобна аутентификации Basic. Вместо пересылки удостоверений по кабелю в кодировке Base64, она хеширует пользовательский пароль и передает по сети его хешированную версию. Хотя это выглядит более безопасным, Digest-аутентификация не слишком распространена. В результате она редко используется вне контролируемых сред (таких как корпоративные сети).

Аутентификация Passport использует Microsoft Passport в качестве базовой инфраструктуры. Microsoft Passport реализует централизованное управление идентичностью. В этом случае удостоверениями пользователей управляет отдельный Passport-сервер. Хотя обычно инфраструктура Passport опирается на Microsoft, вы можете организовать собственный хост для инфраструктуры Passport внутри вашей компании, и использовать его вместо него. Как ни странно, он помечен в управляющей консоли заголовком ".NET Passport authentication", но представляет собой общий механизм аутентификации, который не привязан к .NET Framework. Более того, вы более не должны использовать аутентификацию Passport, поскольку ей на смену пришел Windows Live ID — часть платформы Windows Live. Windows Live ID основан на той же концепции централизованно хранящихся идентификаторов, но реализован на гораздо более открытой платформе, нежели Passport. За подробностями обращайтесь к Windows Live SDK по адресу `http://msdn2.microsoft.com/en-us/library/bb264574.aspx`.

И, наконец, IIS поддерживает один дополнительный метод аутентификации — сертификационную аутентификацию, которая не показана на рис. 19.3, поскольку она конфигурируется через SSL.

На заметку! Для отладки приложений ASP.NET должна быть включена Windows-аутентификация, поскольку Windows определяет, разрешено ли вам заниматься отладкой, на основе выданных вам прав внутри Windows.

Авторизация в IIS 5.x и IIS 6.0

На рис. 19.5 показано, как конфигурировать ограничения IP-адресов с IIS. Ограничения IP-адресов обеспечивают возможность ограничить доступ к Web-серверу только с машин, указанных в списке разрешенных. Это имеет смысл, если вы хотите открыть доступ к своему Web-серверу только нескольким известным партнерам по бизнесу.

Конфигурация безопасности IIS 7.0

В среде Windows Vista или в следующей версии Windows Sersver — Windows Sersver 2008 вам придется работать со службами Internet Information Services (IIS) 7.0. Как известно из главы 18, архитектура IIS 7.0 существенно усовершенствована по сравнению с версией IIS 6.0, особенно в отношении намного более тесной интеграции с ASP.NET, упрощения конфигурации и расширяемости.

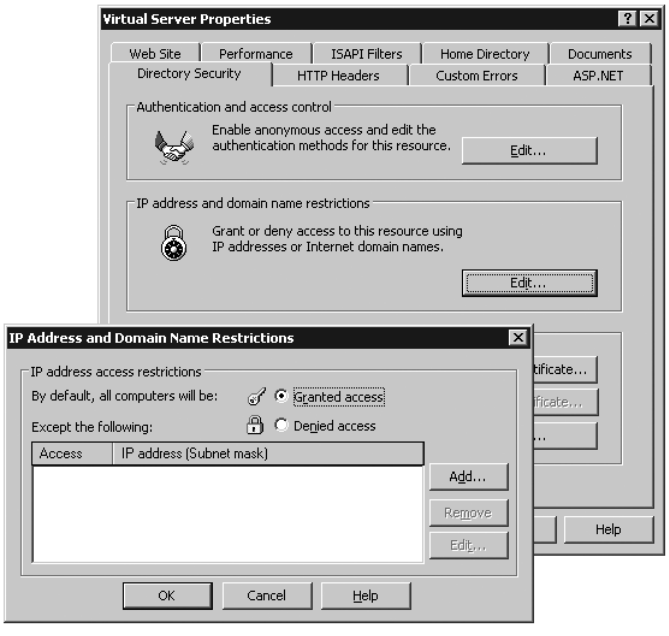

Рис. 19.5. Ограничения IP-адресов на IIS

На самом деле IIS 7.0 по умолчанию запускается в так называемом "интегрированном" режиме ASP.NET, что означает, что Web-сервер интегрирует свои собственные потоки обработки HTTP с потоками модулей HTTP ASP.NET. Это обеспечивает интеграцию "родных" модулей HTTP IIS 7.0 и стандартных модулей HTTP внутри общего потока обработки запросов Web-сервера. Эти модули реализуют функциональность общего назначения, такую как безопасность всех Web-приложений, развернутых на IIS 7.0.

Эта архитектура предоставляет множество преимуществ, относящихся к безопасности, перечисленных в главе 18. Как известно из главы 18, где речь шла о модели конфигурации, вы можете конфигурировать IIS 7.0 таким образом, что он позволит хранить (и перезаписывать) почти все настройки, помещая их в раздел конфигурации `<system.webServer>` файла `web.config` вашего приложения. Что касается развертывания, вы можете реализовать развертывание методом `xcopy`, поскольку нет необходимости сохранять настройки IIS отдельно; разумеется, это важно для конфигурации безопасности. На рис. 19.6 показано различие между классической моделью обработки запросов IIS 5.x и IIS 6.0 и интегрированной в ASP.NET моделью IIS 7.0, обусловленное модулями и обработчиками HTTP, которые представляют собой ядро реализации стражей безопасности.

В IIS 5.x и IIS 6.0 среда ASP.NET не могла повлиять на то, что произойдет до того, как запрос пройдет расширение ASP.NET ISAPI исполняющей системы ASP.NET. Как видите, при работе в интегрированном режиме IIS 7.0 позволяет вам полагаться на существующие модули HTTP в ASP.NET. IIS 7.0 предоставляет множество дополнительных "родных" модулей HTTP для использования функциональности, поставляемой с Web-сервером. Это значит, что исполняющая система ASP.NET может быть задействована с самого начала обработки запросов. Примером "родного" модуля, поставляемого с Web-сервером, может служить `BasicAuthenticationModule`, реализующий квитирование (handshake) Basic-аутентификации, что не включено в ASP.NET изначально. За подробностями о работе Basic-аутентификации обращайтесь к главе 22. Обзор модулей, изначально вклю-

ченных в ASP.NET, представлен ниже в разделе "Архитектура безопасности ASP.NET". Обратите внимание, что определенные типы модулей поставляются как с ASP.NET, так и с IIS 7.0. Например, когда речь идет об авторизации на основе URL, IIS 7.0 старается предоставить это средство, даже если вы не используете ASP.NET на целевой системе Web-сервера. Поэтому он поставляется с собственным независимым модулем авторизации. И, наконец, это означает, что авторизация URL — одна из нескольких конфигураций, в которых нужно учитывать наличие обеих ее реализаций — как на базе IIS, так и на базе ASP.NET. Однако вы можете настроить IIS 7.0 на использование только одного из двух механизмов. Подробнее о том, как это делается, а также прочих деталях авторизации читайте в главе 23.

Такое изменение архитектуры службы Web-сервера оказало замечательное влияние на безопасность ваших Web-приложений. В интегрированном режиме ASP.NET может участвовать в обработке запросов к Web-серверу с самого первого момента (а не только после обработки ISAPI); это обеспечивает великолепную повторную используемость средств ASP.NET между разнотипными Web-приложениями на основе разных платформ разработки, работающих на IIS. Например, теперь вы можете использовать основанную на ASP.NET аутентификацию с помощью форм (описанную подробно в главе 20) для любого другого типа приложений вроде классических приложений ASP или даже приложений PHP. Это происходит потому, что в интегрированном режиме исполняющая система ASP.NET участвует в запросе с самого начала. С другой стороны, интегрированный режим ASP.NET позволяет использовать другие каркасы на базе ASP.NET, такие как API членства и API ролей (описанные подробно в главе 23) с методами аутентификации, отличными от аутентификации форм. Например, вы можете использовать API членства вместе с Basic-аутентификацией, полагаясь на поток модуля HTTP IIS 7.0 с соответствующей настройкой.

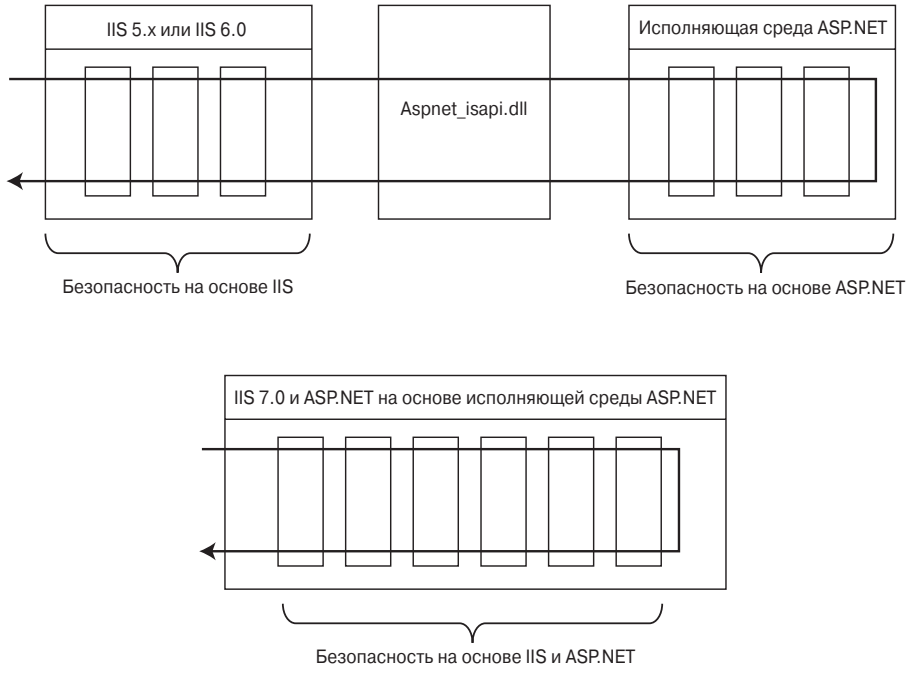

Рис. 19.6. Классическая модель IIS в сравнении с интегрированной моделью ASP.NET в IIS 7.0