Предисловие

Пожалуй, нет ничего более неприятного на свете, чем быть разбуженным рано утром во время отпуска. Мозг еще цепляется за остатки сна, а слух уже ловит «трубный глас» мобильника, играющего мелодию, которая настроена на непосредственного руководителя. Полный неприятных предчувствий, вы подымаете трубку и слышите роковые слова: «Сервер взломан, почта похищена, сеть неработоспособна, сайт в черном списке». И, несмотря на то что вы предсказывали это еще два месяца назад, это произошло именно тогда, когда вам удалось отпроситься в отпуск. И теперь большая часть того, что вам хочется сказать, состоит из слов, воспроизвести которые не возьмется ни один забор...

Всего лишь несколько лет назад таким абзацем можно было бы начинать фантастические произведения на тему «постъядерной зимы», захвата мира кибермозгом и т. п. Сейчас это пойдет разве что на скромную газетную статью об очередном взломе очередного сервера. Тема информационной безопасности, в особенности корпоративной информационной безопасности, сейчас стоит остро, как никогда, – почти каждый день приносит сообщения то о взломе банка и утечке данных кредитных карт, то о взломе крупного игрового сервера и утечке сотен тысяч пользовательских бюджетов. Ну а уж о «сливе» информации действующими и уволенными сотрудниками я даже и не говорю – порой ценные данные от утечки спасают только глупость и некомпетентность. Есть защита от дурака, а есть защита дураком, и зачастую только она одна и выручает – люди просто не знают, что у них лежит под боком и кому это можно сбыть. Но однажды находятся те, кто догадывается...

Почему так происходит? Да потому, что правильно построенная информационная безопасность предъявляет множество требований к организации и людям. Требуется наличие грамотно составленной политики информационной безопасности, которая немыслима без полного анализа бизнес-процессов, производящих и потребляющих данные. В существующей системе их далеко не всегда можно реализовать, как правило, из-за человеческого фактора (Вася – мой друг! Что из того, что он не сотрудник? Настроить сеть так, чтобы он мог прийти к нам в офис и выходить в инет со своего планшета!). Да и финансовые соображения здесь играют далеко не последнюю роль – затраты на безопасность по определению пассивны, и не приведи Господи применить их по назначению! Раза после второго, когда «вдруг» ценные коммерческие данные оказываются у конкурентов просто потому, что были переотправлены (легитимно!) через бесплатный почтовый ящик, который на следующий день был взломан, когда содержимое конфиденциального письма одного руководителя другому становится известно третьему, от которого его нужно было категорически утаить (потому что один из руководителей был в отпуске за рубежом и письмо ему зачитывала в трубку секретарь, которая потом проболталась подруге...), интерес к защите информации «внезапно» просыпается и приходится с криком «Ура!» бросаться на амбразуру (заказывать оборудование, устанавливать софт, планировать, писать, раздавать указания и т. д.).

Зато потом в течение какого-то времени одно слово – «утечка» – будет обладать в течение какого-то времени (пока еще не забылось) поистине магической силой – вопросы будут решаться максимально быстро, счета подписываться без вопросов...

Возможно, эта книга сэкономит вам несколько ценных часов. Или даже дней. Потому что информация, собранная в ней в компактном и готовом к употреблению виде, тоненьким слоем размазана по Интернету, по зиллиону сайтов, форумов, файлов технической документации и т. д.

Пользуйтесь на здоровье.

Введение

Это книга о том, как построить защиту информации в корпоративной сети. О том, как обеспечить основной принцип – надежно доступна кому надо, надежно защищена от того, от кого не надо, и надежно контролируется тем, кем надо. Рассчитана она на администраторов сетей, в которых, кроме офиса, имеются еще и филиалы или розничная сеть, требующие постоянной связи с центром, хотя многие приемы с некоторыми оговорками можно применять в любых случаях. За основу для построения сети берется ОС UNIX (конкретно FreeBSD). Да-да, не переживайте, у вас все в порядке с глазами. На ответственных постах не будет никакого Windows. У Windows будет своя задача – Active Directory, офис, базы данных, пользователи... В качестве же ОС для построения сети – только UNIX.

Это естественным образом подводит нас к тому, что многое здесь будет делаться «самопально» – написанием собственных скриптов, обработчиков, посредников и т. д. Специализированный код всегда работает быстрее и эффективнее универсального. Платой за это является то, что сначала этот специализированный код необходимо написать, поэтому крайне желательно, чтобы вы умели программировать на каком-либо скриптовом языке, хоть на Visual Basic Script (на самом деле для Windows – вполне себе ничего язык). Ну а потому что это UNIX, конфигурироваться все это будет исключительно правкой текстовых файлов. При этом, поскольку я отношусь к тем людям, которые предпочитают вместо навязывания готовых решений выдать максимальное количество информации и оставить с ней наедине, чтобы вы сами создали свой, нужный именно вам конфигурационный файл, поясняться все будет очень подробно, причем говорить будем не только о том, что произойдет, если сделать как сказано, но и что

случится, если сделать не так, как сказано, а наоборот. Хотя, конечно, готовые примеры будут, и в большом количестве.

В главе первой мы составим наши наполеоновские планы по защите от злобных хакеров и запустим наш «паспортный стол» – Удостоверяющий Центр (Certificate Authority), который впоследствии позволит нам беспрепятственно делить пользователей на «своих» и «чужих».

В главе второй расскажем об установке сертификатов в браузеры и системное хранилище Windows, рассмотрим типичные ошибки при использовании сертификатов.

В главе третьей настроим модули для сквозной аутентификации на сервере без использования такой громоздкой и в последнее время весьма «дырявой» вещи, как Samba. Мы люди простые, обойдемся без нее.

В главе четвертой закрываем самое ценное – электронную почту. Защищается среда передачи внутри офиса как от клиента к серверу (отправляемая почта), так и от сервера к клиенту (просматриваемая почта). Защищается также коммуникация между серверами узлов сети, а при наличии правильной настройки внешних серверов – и коммуникация с ними. Здесь же мы расскажем о том, как настроить клиентские почтовые программы – Mozilla Thunderbird и Microsoft Outlook, а также какие тут могут возникать ошибки.

В главе пятой разворачиваем веб-интерфейс для доступа к электронной почте с произвольной точки (чтобы не приходилось секретарше читать письма вслух, когда руководитель в Дубае:-)) и защищаем его от посторонних, предполагая, что с сертификатами в браузере мы уже умеем работать.

Ну и в главе шестой защищаем служебные коммуникации – собственно программы-аутентификаторы, а также разбираемся, как избежать необходимости указания паролей в командных файлах Windows, запускаемых по расписанию или событию (например, при обработке сигнала от UPS). Ну а про то, что рутовый пароль вообще не должен **НИГДЕ** использоваться, кроме как для входа с консоли в самом крайнем случае, я думаю, вы и так знаете. Ну а не знали – так знаете теперь.

Понятное дело, что нам придется разбирать примеры. Чтобы каждый раз не объяснять, что «из пункта А до пункта Б шел пешеход, он шел и думал», заранее опишем конфигурацию, применительно к которой будут строиться любые примеры. Эта конфигурация показана на рис. В.1.

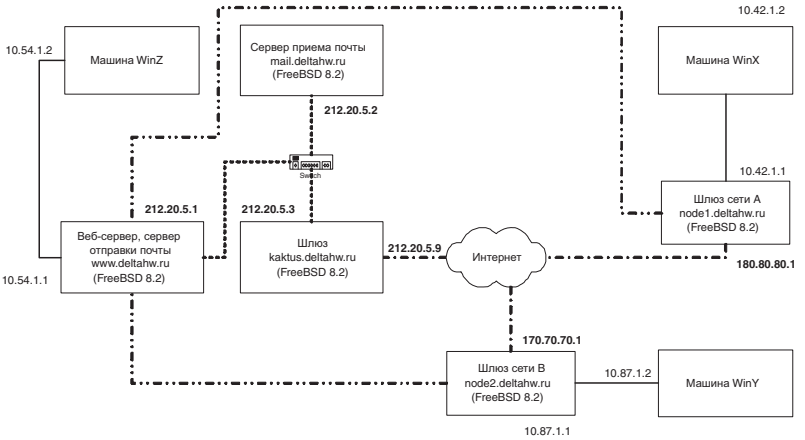

Рис. В.1. Конфигурация сети, на которой будут разбираться примеры

Перед тем как начать приводить примеры, я хочу сделать одно замечание. Ниже приводится описание некой компании, применительно к которой будут рассматриваться все примеры. Также мне придется приводить фрагменты логов и конфигурационных файлов. В этих примерах, логах, файлах будут указаны почтовые адреса, адреса серверов и прочая информация, даже «персональные данные»! Так вот – все эти адреса и прочее – ненастоящие! Несмотря на то что приводимая конфигурация вся протестирована, все фрагменты будут намеренно искажены так, что все почтовые адреса, IP-адреса и имена серверов будут заменены придуманными, никогда не существовавшими адресами. Они только похожи на настоящие. Имейте это в виду.

Кроме того, поскольку нам придется выдавать и проверять персональные сертификаты, мы «назначим» в компании как минимум директора и системного администратора. Все совпадения с реальными именами, фамилиями и должностями случайны, все персонажи вымышлены. Как говорил в свое время один из моих любимых писателей Томас Майн Рид: «Читатель! Перед тобой роман и ничего более. Не считай автора книги ее героем».

Предположим, существует фирма «DeltaHardware Ltd.», которая занимается производством и продажей средств промышленной автоматизации. У нее есть центральный офис с IP-адресом 212.20.5.1 и некоторое количество филиалов – на схеме показаны два: с IP-адресами 180.80.80.1 и 170.70.70.1. Все подразделения объединены в

один VPN, в котором сеть головного офиса имеет адрес 10.54.1.0/24, сеть первого филиала – 10.42.1.0/24, а сеть второго – 10.87.1.0/24. В каждой сети UNIX-шлюз имеет адрес .1 (например, 10.54.1.1), контроллер домена Windows – .2 (например, 10.42.1.2). Фирма имеет собственный сайт по адресу http://www.deltahw.ru, который размещен также на 212.20.5.1, почтовый сервер mail.deltahw.ru с адресом 212.20.5.2 и файрволл по адресу 212.20.5.3. Внешний адрес файрволла 212.20.5.9. На всех UNIX-шлюзах установлена FreeBSD 8.2-STABLE, на контроллерах домена Windows – Windows 2003 R2 Server, на рабочих станциях – Windows XP. Да, я знаю, что уже есть Windows 7 и Windows 2008 Server. Но корпоративная среда отличается большой инертностью и склонностью не менять программы без крайней на то необходимости – ведь это все будет стоить немалых денег. Вследствие того, что у директора «экономить мое любимое», на рабочих станциях стоит минимум платного программного обеспечения и вместо привычной многим связки Microsoft Office плюс Microsoft Exchange будет стоять Mozilla Thunderbird + Mozilla Sunbird + хранение почты на сервере и доступ по протоколу IMAP. Имя домена Windows – deltahw.int, «короткое» имя – DELTAHW (потом эти имена нам понадобятся).

Директором конторы является Головань Павел Серафионович, почтовый адрес golovan-ps@deltahw.ru, системным администратором – Северцев Руслан Валентинович, почтовый адрес – severtsev-rv@deltahw.ru.

Вперед, викинги!

ГЛАВА 1 Корпоративный «паспортный стол»

Большая работа всегда начинается с большого... нет, не перекура, а с большого плана. Итак:

- филиалы при подключении к VPN должны быть уверены в том, что «та» сторона – это именно тот компьютер, за который он себя выдает;
- почтовые серверы при пересылке писем между узлами VPN должны быть уверены в том, что получатель – именно тот, за которого он себя выдает;
- при подключении к веб-интерфейсу почты сервер должен быть уверен, что клиент имеет право к нему подключаться, а клиент – в том, что сервер – именно тот сервер, за которого он себя выдает.

Да и вообще при установлении защищенного соединения первоначально обе стороны должны убедиться в том, что противоположная сторона – это именно та, которая нужна нам, а не подстава.

Каким образом стороны могут убедиться в том, что с той стороны – нужный тебе абонент, а не злобный хакер? Либо стороны предварительно обмениваются какой-либо информацией по отдельному, независимому и заведомо неподконтрольному хакеру каналу (например, по сотовому созвониться), либо они доверяют некоторому третьему лицу, которое заранее выдало нечто, что бы подтверждало то, что ты – это ты, причем выдало обеим сторонам. В России такой третьей стороной является УФМС, иначе говоря – паспортная служба. Предполагается, что доверие сторон к третьему лицу абсолютно – если, допустим, на паспорте стоит печать УФМС и фото,

значит, он подлинный. Вот поначалу мы и займемся тем, что создадим собственный «паспортный стол», а потом объявим, что в масштабах компании наличие сертификата, подписанного сертификатом нашего УЦ, означает его подлинность.

УЦ будет размещаться по адресу 212.20.5.1, на компьютере www.deltahw.ru (это чтобы не путаться при создании сертификатов).

И точно так же, как наличие паспорта у человека означает предоставление ему некоторых привилегий, так и наличие сертификата будет означать предоставление привилегий с точки зрения программы. Правда, привилегия будет всего одна – предоставить доступ к данным.

1.1. Создание удостоверяющего центра

Прежде чем выдавать паспорта, надо построить паспортный стол. Прежде чем выдавать сертификаты, необходимо будет создать удостоверяющий центр (УЦ). Правда, мы будем пользоваться его англоязычным наименованием – CA, Center of Authority – так будет понятнее при чтении документации и конфигурационных файлов. В UNIX для создания не нужно ничего запускать: собственно весь CA – это несколько файлов и каталогов, ключ самого CA, сертификат CA и список отозванных сертификатов. Ну и конечно же конфигурационный файл openssl.cnf, настройки которого будут играть чрезвычайно важную роль. Собственно говоря, openssl.cnf и ключ корневого сертификата CA плюс его индексный файл и есть CA, потому что все остальное – просто обвязка.

1.1.1. Файловая структура CA

Сначала определяемся с путями к файлам. Нам понадобится отдельный каталог, доступный только пользователю root, в котором будут находиться:

- принятые нами запросы на сертификацию (CSR);
- подписанные нами сертификаты (CRT);
- список отозванных сертификатов (CRL);
- закрытый ключ нашего собственного сертификата.

Также здесь будут находиться одна из копий сертификата нашего CA и рабочие файлы CA. Имена файлов можно использовать произвольные, какого-либо стандарта на это нет. Я использую структуру каталогов, которой когда-то придерживался модуль mod_ssl веб-сервера Apache 1.x, хотя конечно же можно использовать любые другие.

Структура каталогов следующая:

- /usr/local/share/ca-dhw. Как видно из листинга ниже – корневой каталог CA;
- подкаталоги:
  - ssl.crl – для хранения CRL;
  - ssl.crt – для хранения подписанных сертификатов;
  - ssl.csr – для хранения запросов на сертификацию;
  - ssl.key – для хранения ключей. Как правило, в этом каталоге лежит самый главный ключ CA.

Создаем каталоги:

```
# mkdir /usr/local/share/ca-dhw
# cd /usr/local/share/ca-dhw
# mkdir ssl.crl
# mkdir ssl.crt
# mkdir ssl.csr
# mkdir ssl.key
# cd ..
# chmod -R 0700 ca-dhw
```

Создаем служебные файлы:

```
# cd ca-dhw
# touch index.txt
# echo '01' > serial
```

Файл index.txt – это файл списка подписанных нами сертификатов от самого начала. Он будет расти в размере с каждым выданным сертификатом. Несмотря на то, что это просто текст, лучше его текстовыми редакторами не править – ну разве что удалить строчку о заведомо неверном сертификате, разумеется, после удаления самого сертификата.

Формат этого файла очень простой. На каждый сертификат отводится по одной строке. В первом столбце указывается «V» для действующего сертификата (видимо, от *vita* – жизнь (лат.)) или «R» – для

отозванного (а это, видимо, от слова *Revoked* – отозванный). Во втором столбце указывается дата окончания срока действия сертификата. Формат даты текстовый, в виде строки ГГММДДЧЧММССZ (последний символ – не обозначение, это именно буква Z – это строка типа `ASN.1 UTCTime`). В третьем столбце, если он есть, указываются дата отзыва сертификата и причина отзыва через запятую. Причины отзыва мы рассмотрим, когда дойдем до рассмотрения CRL. В четвертом столбце указывается серийный номер сертификата в шестнадцатеричной форме без префикса 0x. Пятый столбец почему-то всегда содержит слово «unknown». Ну и в шестом столбце приводится полное значение поля `Subject` сертификата. Только этот файл да еще файл serial – вот и все, что нужно поправить в том случае, если вдруг выдали ошибочный сертификат.

Файл serial содержит серийный номер следующего сертификата, который будет выдан. В этом же каталоге обычно образуется файл index.txt.attr, но этот файл служебный, его содержимое неинтересно.

1.1.2. Конфигурационный файл openssl.cnf

Теперь займемся правкой конфигурационного файла openssl.cnf. Это очень важный этап, поэтому файл будет разбираться по секциям. Не все секции для нас важны, но тем не менее из песни слов не выкинешь. О тех секциях, где ничего нет необходимости менять, мы просто упомянем вскользь. Чтобы получить из этого готовый конфигурационный файл, нужно просто объединить все секции в один файл. Впрочем, особой необходимости это делать нет – OpenSSL поставляется всегда с готовым файлом openssl.cnf, который уже можно править.

Начало, то есть задание `HOME`, `oid_section` и секция `[new_oids]`. В секции `[new_oids]` должны размещаться новые OID'ы, но у нас их нет, поэтому секция пустая.

```
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd
oid_section             = new_oids
[new_oids]
```

Секция с параметрами CA. Ее мы рассмотрим весьма подробно – в ней множество важных вещей. В секции `[ca]` указывается единственный параметр – имя секции с настройками. Зачем так сделано, непонятно.

```
[ ca ]
default_ca              = CA_default

[ CA_default ]
Dir                     = /usr/local/share/ca-dhw
Certs                   = $dir/ssl.crt
crl_dir                 = $dir/ssl.crl
database                = $dir/index.txt
new_certs_dir           = $dir/ssl.crt
certificate             = $dir/caserv.crt
serial                  = $dir/serial
crl                     = $dir/cacrl.crl
private_key             = $dir/ssl.key/caserv.key
RANDFILE                = $dir/ssl.key/.rand
x509_extensions         = usr_cert
name_op                 = ca_default
cert_opt                = ca_default
copy_extensions         = copy
crl_extensions          = crl_ext
default_days            = 365
default_crl_days        = 30
default_md              = md5
preserve                = no
policy                  = policy_match
```

Пожалуй, самая важная секция файла. Впрочем, в этом файле все секции важные.

dir – указывает корень всего CA.

certs – каталог для хранения выданных сертификатов.

crl_dir – каталог для хранения созданных CRL.

database – индексный файл для хранения списка сертификатов. Должен существовать к началу процесса выдачи сертификатов.

new_certs_dir – куда будут помещаться новые сертификаты по умолчанию.

certificate – имя файла сертификата CA.

serial – имя служебного файла, содержащего следующий номер сертификата. Должен существовать к началу процесса выдачи сертификатов и содержать правильный номер.

crl – имя файла CRL.

private_key – имя файла ключа сертификата CA.

RANDFILE – файл рэндома для работы генераторов случайных чисел.

x509_extensions – имя секции с описанием расширений сертификата.

name_opt – имя секции с параметрами отображения имен.

cert_opt – имя секции с параметрами полей сертификатов.

copy_extensions – если задано `copy`, то в сертификат копируются все отсутствующие там расширения из CSR. Если задано `copyall`, то в случае присутствия расширения в сертификате оно сначала удаляется, а потом копируется из CSR.

crl_extensions – имя секции с описанием расширений CRL.

default_days – срок действия сертификата по умолчанию.

default_crl_days – срок действия CRL по умолчанию. Обратите внимание, он достаточно короткий!

default_md – message digest для сертификата по умолчанию. Другие варианты: `sha1`, `mdc2`.

preserve – не сохранять порядок полей, заданный в поле `Subject` при создании сертификата.

policy – имя секции, описывающей элементы DN. Если в секции элемент DN не описан, он просто молча удаляется.

Секции политик для создания сертификатов CA называются `[policy_match]`, а для создания обычных сертификатов – `[policy_anything]`. В них ничего не меняется. Различные варианты параметра – `match`, `supplied`, `optional` – указывают на то, является данный атрибут обязательным или нет. Не думаю, что вам когда-либо придется править эти секции, разве только возникнет необходимость добавить какой-либо атрибут к запросу атрибутов при создании CSR.

```
[ policy_match ]
countryName                     = match
stateOrProvinceName             = match
organizationName                = match
organizationalUnitName          = optional
commonName                      = supplied
emailAddress                    = optional
[ policy_anything ]
countryName                     = optional
stateOrProvinceName             = optional
localityName                    = optional
organizationName                = optional
organizationalUnitName          = optional
commonName                      = supplied
emailAddress                    = optional
```

Дальше идет секция `[req]`, которая задействуется как раз тогда, когда создается CSR. Секция небольшая, но важная:

```
[ req ]
default_bits                    = 2048
default_keyfile                 = privkey.pem
distinguished_name              = req_distinguished_name
attributes                      = req_attributes
x509_extensions                 = v3_ca
string_mask                     = nombstr
req_extensions                  = v3_req
```

default_bits – размер ключа по умолчанию в битах.

default_keyfile – имя файла ключа по умолчанию.

distinguished_name – имя секции, которая описывает правила запроса DN у пользователя при формировании CSR.

attributes – имя секции, которая описывает дополнительные атрибуты CSR.

x509_extensions – имя секции, которая описывает расширения, добавляемые в сертификат CA или самоподписанный сертификат.

input_password, output_password – здесь может быть задан пароль для открытия ключа или для формирования ключа. Если он не задан здесь и не отключен другим ключом, то запрашивается у пользователя.

string_mask – маска, накладываемая на строковые значения. Строки данных в сертификатах могут быть разного типа и содержать различные наборы символов. Всего типов строк четыре: `PrintableString`, `T16String (TelelexString)`, `BMPString` и `UTF8String`. Не все браузеры поддерживают работу с `BMPString` и `UTF8String`, которые позволяют закодировать буквы национальных алфавитов. Значением параметра может быть одна из перечисленных ниже строк:

```
default: PrintableString, T61String, BMPString
pkix: PrintableString, BMPString
utf8only: только UTF8Strings
nombstr : PrintableString, T61String (но не BMPStrings или UTF8Strings).
```

Обычно используется *nombstr* для гарантии совместимости, потому что некоторые программы могут аварийно завершаться, обнаружив строку типа `BMPString` или `UTF8String`. И, хотя эти типы позволяют использовать в сертификатах символы национальных алфавитов, я бы не рекомендовал ими пользоваться – сертификат выдается надолго, и пользоваться им будут в разных местах разными программами.

req_extensions – имя секции, в которой описываются включаемые в CSR расширения.

Далее идет секция `[req_distinguished_name]`, которую обычно правят на предмет умолчаний в значениях, запрашиваемых при формировании DN. В ней описываются запрашиваемые поля, их значения по умолчанию, минимальные и максимальные длины.

```
[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = RU
countryName_min                 = 2
countryName_max                 = 2
stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Novosibirsk region
localityName                    = Locality Name (eg, city)
localityName_default            = Novosibirsk
0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = DeltaHardware Ltd
organizationalUnitName          = Organizational Unit Name (eg, section)
commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64
emailAddress                    = Email Address
emailAddress_max                = 64
```

Конечно же, вы уже обратили внимание на то, что запрос `organizationName` записан в каком-то странном виде с нулем впереди. Это означает, что теоретически можно было бы запросить еще одно `organizationName` и для его запроса использовать форму **1.organizationName**. Данной возможностью – иметь в сертификате несколько атрибутов одного типа, перечисляемых через `0.<имя_атрибута>`, `1.<имя_атрибута>`, – редко кто пользуется, но тем не менее она есть.

При создании запроса на сертификат пользователю на консоль будет выдано сообщение **Country Name (2 letter code) [RU]:**, и в ответ будет ожидаться ввод двух символов. Исправите параметр `countryName` – и будет выведен другой текст, исправите `countryName_default` – изменится значение по умолчанию в квадратных скобках, исправите `countryName_min` или `countryName_max` – изменятся минимальная и максимальная длины ввода. Разумеется, не обязательно описывать все параметры: если что-то не описано, то принимается значение по умолчанию. Исправляют обычно значения по умолчанию и иногда длины, хотя, с моей точки зрения, длины достаточные. Максимальные длины полей для данных, вводимых при создании запроса на сертификат, приведены в табл. 1.1

Таблица 1.1. Максимальные длины полей элементов атрибута Subject

Имя поля CSR	Максимальная длина поля в байтах
Country	2
CommonName	64
Locality	128
StateOrProvince	128
OrganizationalName	64
OrganizationalNameUnit	64
Email	128

В конфигурационном файле приведены описания не всех возможных полей DN, описание всех возможных полей заняло бы слишком много места, приведены только наиболее нужные. Если вам для каких-либо целей нужно расширить количество запрашиваемых атрибутов, это можно сделать, добавив нужные поля к секции. Какие есть еще атрибуты, можно посмотреть в [1]. Например, если добавить в секцию приведенные ниже строки, то при генерации CSR будет запрашиваться **Given Name**.

```
givenName               = Given Name
givenName_max           = 128
```

За ней идет секция `[req_attributes]`, которая описывает необязательные атрибуты CSR. Обычно ее никто не правит.

```
[ req_attributes ]
challengePassword       = A challenge password
challengePassword_min   = 4
challengePassword_max   = 20
unstructuredName        = An optional company name
```

За `[req_attributes]` идет секция `[usr_cert]` с расширениями, добавляемыми в сертификат по умолчанию. Как правило, она не используется, потому что требуется обычно подписывать сертификаты SSL-серверов и сертификаты SSL-клиентов. Поэтому секция чаще всего полупустая и ненастроенная. Хотя конечно же никто не мешает вам настроить ее и пользоваться.

```
[ usr_cert ]
basicConstraints        = CA:FALSE
keyUsage                = digitalSignature, keyEncipherment
nsComment               = "OpenSSL Default Generated Certificate"
```

```
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer:always
```

За `[usr_cert]` идут собственно секции, описывающие атрибуты для создаваемого сертификата сервера, – `[ssl_server]` и клиента – `[ssl_client]`. Ниже разбирается секция сервера, секция клиента точно такая же, только значение атрибута `nsCertType` будет другим.

```
[ ssl_server ]
basicConstraints        = CA:FALSE
nsCertType              = server, email
keyUsage                = digitalSignature, keyEncipherment
extendedKeyUsage        = serverAuth, emailProtection
nsComment               = "OpenSSL Certificate for SSL Web Server"
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer:always
authorityInfoAccess     = caIssuers;URI:http://certs.deltahw.ru
issuerAltName           = issuer:copy, URI:http://certs.deltahw.ru/certs/caserv.crt
crlDistributionPoints   = URI:http://certs.deltahw.ru/certs/cacrl.crl
nsCaRevocationUrl       = http://certs.deltahw.ru/certs/cacrl.crl
```

basicConstraints указывает, что это не сертификат CA. Больше он ни для чего не нужен – он только указывает, это сертификат CA или нет.

nsCertType – устаревший атрибут, указывавший тип создаваемого сертификата. Этот атрибут указывать было не обязательно. Возможные значения этого атрибута: client, server, email, objsign, reserved, sslCA, emailCA, objCA. Все эти значения задают различные типы сертификата, наиболее употребимые – это client, server и email, которые задают соответственно сертификат SSL-клиента, SSL-сервера и сертификат для защиты электронной почты. Фактически этот атрибут дублирует атрибут `extendedkeyUsage`.

keyUsage – указывает, для чего будет использоваться сертификат, сгенерированный по данному запросу. Возможные значения атрибута: digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly and decipherOnly. Практически все значения имеют достаточно понятные названия – шифрование ключей и данных, подпись сертификатов и CRL. Атрибут nonRepudiation устанавливается, когда ключ используется для проверки сигнатур, используемых для предоставления «безотказного» сервиса, который не позволит второй стороне отказаться от факта участия в обмене данными. Такое вот юридически-полицей-

ское толкование. Ну а KeyAgreement используется для согласования сертификатов.

extendedKeyUsage – фактически настоящее применение сертификата описывается здесь. Возможные значения атрибута:

- serverAuth – аутентификация сервера;
- clientAuth – аутентифкация клиента;
- codeSigning – подпись программного кода;
- emailProtection – защита электронной почты;
- ipsecEndSystem – сервер IPSec;
- ipsecTunnel – туннель IPSec;
- ipsecUser – оконечный пользователь IPSec;
- timeStamping – простановка меток времени.

Видимое расхождение с атрибутом `KeyUsage` объясняется просто – этот атрибут относится к расширенным, а атрибут `KeyUsage` – к основным.

nsComment – просто комментарий.

subjectKeyIdentifier – идентификатор ключа субъекта. Обязательный атрибут. Может быть задан либо словом `hash` – в таком случае в качестве идентификатора будет взят 160-битный SHA-1 хэш от общего ключа субъекта или же задан вручную. Задавать вручную категорически не рекомендуется.

authorityKeyIdentifier – идентификатор ключа CA. Обязательный атрибут. Может быть задан двумя параметрами – `keyid` и `issuer`, каждый может сопровождаться опцией `always`. При указании `keyid` в сертификат копируется идентификатор ключа субъекта родительского сертификата. При указании `issuer` в сертификат копируются атрибут `Subject` и серийный номер ключа CA. Если указана опция `always`, то при невозможности скопировать атрибут процесс создания сертификата завершается аварийно.

authorityInfoAccess – дополнительная информация о CA, выдавшем сертификат. Указывается в формате `OID;<имя или ссылка>`, где OID должен иметь значение `caIssuers` или `OSCP`, а формат поля `<имя или ссылка>`, который на самом деле то же самое, что SAN, будет рассмотрен после полного примера конфигурационного файла openssl.cnf.

subjectAlternativeName (SAN) – дополнительное имя субъекта. Несмотря на то что в примере этой строки нет, мы рассмотрим ее здесь. Дело в том, что изначально в конфигурационном файле она была, но впоследствии была удалена, потому что, оказывается, нельзя объеди-

нять атрибуты, заданные в конфигурационном файле, и атрибуты, задаваемые в CSR. Можно либо ничего не делать с атрибутами из CSR (и так и делается по умолчанию), либо скопировать то, что отсутствует в сертификате (copy), либо скопировать все, удалив предварительно дублирующиеся расширения (copyall). Поэтому атрибут был удален, и задаваться он будет в CSR. Задаваться SAN может большим количеством разнообразных способов, чему будет посвящен отдельный раздел.

issuerAlternativeName – дополнительное имя CA. Может задаваться параметром `issuer` с опцией `copy`, что будет означать скопировать в сертификат SAN из сертификата CA, а также всеми возможными способами задания SAN. В примере задана ссылка на сертификат CA через опцию HTTP.

crlDistributionPoints, а также *nsCARevocationUrl* – задают точку распространения CRL. Причем первый атрибут стандартный, задает ее в виде URI, а второй, устаревший, но еще повально много где используемый, задает ее просто в виде ссылки. Оба атрибута указывают непосредственно на файл CRL.

Секция [ssl_client], на основании которой генерируется сертификат клиента, ничем практически не отличается от секции сервера, отличаются только ключевые параметры `nsCertType`, `extendedKeyUsage`:

```
[ ssl_client ]
basicConstraints            = CA:FALSE
nsCertType                  = client, email
keyUsage                    = digitalSignature, keyEncipherment
extendedKeyUsage            = clientAuth, emailProtection
nsComment                   = "OpenSSL Certificate for SSL Web Server"
subjectKeyIdentifier        = hash
authorityKeyIdentifier      = keyid,issuer:always
authorityInfoAccess         = caIssuers;URI:http://certs.deltahw.ru
issuerAltName               = issuer:copy, URI:http://certs.deltahw.ru/certs/caserv.crt
crlDistributionPoints       = URI:http://certs.deltahw.ru/certs/cacrl.crl
nsCaRevocationUrl           = http://certs.deltahw.ru/certs/cacrl.crl
```

Далее следует небольшая секция `[v3_req]`, которая как раз и описывает атрибуты CSR:

```
[ v3_req ]
basicConstraints            = CA:FALSE
keyUsage                    = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName              = ${ENV::SAN}
```