HowTo

''The solution of this problem is trivial and is left as an exercise for the reader.''
Standard textbook cookie

Table Of Contents

How to solve particular security constraints for an SSL-aware webserver is not always obvious because of the coherences between SSL, HTTP and Apache's way of processing requests. This chapter gives instructions on how to solve such typical situations. Treat is as a first step to find out the final solution, but always try to understand the stuff before you use it. Nothing is worse than using a security solution without knowing it's restrictions and coherences.

Cipher Suites and Enforced Strong Security

- **How can I create a real SSLv2-only server?** [L]

  The following creates an SSL server which speaks only the SSLv2 protocol and its ciphers.

  httpd.conf

```
SSLProtocol -all +SSLv2
SSLCipherSuite SSLv2:+HIGH:+MEDIUM:+LOW:+EXP
```

- **How can I create an SSL server which accepts strong encryption only?** [**L**]

  The following enables only the seven strongest ciphers:

  httpd.conf

```
SSLProtocol all
SSLCipherSuite HIGH:MEDIUM
```

- **How can I create an SSL server which accepts strong encryption only, but allows export browsers to upgrade to stronger encryption?** [**L**]

  This facility is called Server Gated Cryptography (SGC) and details you can find in the `README.GlobalID` document in the mod_ssl distribution. In short: The server has a Global ID server certificate, signed by a special CA certificate from Verisign which enables strong encryption in export browsers. This works as following: The browser connects with an export cipher, the server sends it's Global ID certificate, the browser verifies it and subsequently upgrades the cipher suite before any HTTP communication takes place. The question now is: How can we allow this upgrade, but enforce strong encryption. Or in other words: Browser either have to initially connect with strong encryption or have to upgrade to strong encryption, but are not allowed to keep the export ciphers. The following does the trick:

  httpd.conf

```
#   allow all ciphers for the inital handshake,
#   so export browsers can upgrade via SGC facility
SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
<Directory /usr/local/apache/htdocs>
#   but finally deny all browsers which haven't upgraded
SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
</Directory>
```

- **How can I create an SSL server which accepts all types of ciphers in general, but requires a strong ciphers for access to a particular URL?** [**L**]

  Obviously you cannot just use a server-wide `SSLCipherSuite` which restricts the ciphers to the strong variants. But mod_ssl allows you to reconfigure the cipher suite in per-directory context and automatically forces a renegotiation of the SSL parameters to meet the new configuration. So, the solution is:

  httpd.conf

```
#   be liberal in general
SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
<Location /strong/area>
#   but https://hostname/strong/area/ and below requires strong ciphers
```

```
SSLCipherSuite HIGH:MEDIUM
</Location>
```

Client Authentication and Access Control

- **How can I authenticate clients based on certificates when I know all my clients? [L]**

  When you know your user community (i.e. a closed user group situation), as it's the case for instance in an Intranet, you can use plain certificate authentication. All you have to do is to create client certificates signed by your own CA certificate `ca.crt` and then verifiy the clients against this certificate.

  httpd.conf

  ```
  #   require a client certificate which has to be directly
  #   signed by our CA certificate in ca.crt
  SSLVerifyClient require
  SSLVerifyDepth 1
  SSLCACertificateFile conf/ssl.crt/ca.crt
  ```

- **How can I authenticate my clients for a particular URL based on certificates but still allow arbitrary clients to access the remaining parts of the server? [L]**

  For this we again use the per-directory reconfiguration feature of mod_ssl:

  httpd.conf

  ```
  SSLVerifyClient none
  SSLCACertificateFile conf/ssl.crt/ca.crt
  <Location /secure/area>
  SSLVerifyClient require
  SSLVerifyDepth 1
  </Location>
  ```

- **How can I authenticate only particular clients for a some URLs based on certificates but still allow arbitrary clients to access the remaining parts of the server? [L]**

  The key is to check for various ingredients of the client certficate. Usually this means to check the whole or part of the Distinguished Name (DN) of the Subject. For this two methods exists: The `mod_auth` based variant and the `SSLRequire` variant. The first method is good when the clients are of totally different type, i.e. when their DNs have no common fields (usually the organisation, etc.). In this case you've to establish a password database containing *all* clients. The second method is better when your clients are all part of a common hierarchy which is encoded into the DN. Then you can match them more easily.

The first method:

/usr/local/apache/conf/httpd.conf

```
SSLVerifyClient      none
<Directory /usr/local/apache/htdocs/secure/area>
SSLVerifyClient      require
SSLVerifyDepth       5
SSLCACertificateFile conf/ssl.crt/ca.crt
SSLCACertificatePath conf/ssl.crt
SSLOptions           +FakeBasicAuth
SSLRequireSSL
AuthName             "Snake Oil Authentication"
AuthType             Basic
AuthUserFile         /usr/local/apache/conf/httpd.passwd
require              valid-user
</Directory>
```

/usr/local/apache/conf/httpd.passwd

```
/C=DE/L=Munich/O=Snake Oil, Ltd./OU=Staff/CN=Foo:xxj31ZMTZzkVA
/C=US/L=S.F./O=Snake Oil, Ltd./OU=CA/CN=Bar:xxj31ZMTZzkVA
/C=US/L=L.A./O=Snake Oil, Ltd./OU=Dev/CN=Quux:xxj31ZMTZzkVA
```

The second method:

httpd.conf

```
SSLVerifyClient      none
<Directory /usr/local/apache/htdocs/secure/area>
SSLVerifyClient      require
SSLVerifyDepth       5
SSLCACertificateFile conf/ssl.crt/ca.crt
SSLCACertificatePath conf/ssl.crt
SSLOptions           +FakeBasicAuth
SSLRequireSSL
SSLRequire           %{SSL_CLIENT_S_DN_O}  eq "Snake Oil, Ltd." and \
                     %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"}
</Directory>
```

- **How can I require HTTPS with strong ciphers and either basic authentication or client certificates for access to a subarea on the Intranet website for clients coming from the Internet but still allow plain HTTP access for clients on the Intranet?** [L]

  Let us assume the Intranet can be distinguished through the IP network 192.160.1.0/24 and the subarea on the Intranet website has the URL `/subarea`. Then configure the following outside your HTTPS virtual host (so it applies to both HTTPS and HTTP):

httpd.conf

```
SSLCACertificateFile conf/ssl.crt/company-ca.crt

<Directory /usr/local/apache/htdocs>
#   Outside the subarea only Intranet access is granted
Order                deny,allow
Deny                 from all
Allow                from 192.168.1.0/24
</Directory>

<Directory /usr/local/apache/htdocs/subarea>
#   Inside the subarea any Intranet access is allowed
#   but from the Internet only HTTPS + Strong-Cipher + Password
#   or the alternative HTTPS + Strong-Cipher + Client-Certificate

#   If HTTPS is used, make sure a strong cipher is used.
#   Additionally allow client certs as alternative to basic auth.
SSLVerifyClient      optional
SSLVerifyDepth       1
SSLOptions           +FakeBasicAuth +StrictRequire
SSLRequire           %{SSL_CIPHER_USEKEYSIZE} >= 128

#   Force clients from the Internet to use HTTPS
RewriteEngine        on
RewriteCond          %{REMOTE_ADDR} !^192\.168\.1\.[0-9]+$
RewriteCond          %{HTTPS} !=on
RewriteRule          .* - [F]

#   Allow Network Access and/or Basic Auth
Satisfy              any

#   Network Access Control
Order                deny,allow
Deny                 from all
Allow                192.168.1.0/24

#   HTTP Basic Authentication
AuthType             basic
AuthName             "Protected Intranet Area"
AuthUserFile         conf/protected.passwd
Require              valid-user
</Directory>
```