



Kaspar Brand

Eine für alle

Mehrere SSL-Zertifikate pro IP-Adresse unter Apache

Namensbasierte virtuelle Webserver funktionieren seit langem klaglos. Eine einzige IP-Adresse reicht aus, um beliebig viele Servernamen zu bedienen – aber leider nur bei unverschlüsselten Verbindungen. TLS Server Name Indication ermöglicht dies auch für SSL-geschützte Websites.

Jeder SSL-Webserver benötigt eine eigene IP-Adresse, sonst hagelt es Zertifikatsfehlermeldungen im Browser – routinierte Systemadministratoren haben diese Aussage bereits als Dogma verinnerlicht. Mit der Server-Name-Indication-Erweiterung (SNI) unter TLS lässt sich diese Restriktion aber aufbrechen. Insbesondere weil Apache seit Version 2.2.12 diese Erweiterung unterstützt, können Admins künftig sparsamer mit ihren IP-Adressen umgehen.

Dass SNI notwendig ist, liegt an einer Lücke im ursprünglich spezifizierten Protokoll. Sowohl die SSL-Versionen 2 und 3 als auch TLS 1.0 sehen für den SSL-Client keine

Möglichkeit vor, schon zu Beginn des Verbindungsaufbaus den gewünschten Servernamen anzugeben. Im Browser gibt der Anwender zwar konkrete URLs wie `https://www.example.com/login` und `https://www.example.org` ein, beide Adressen werden aber zunächst gegen IP-Adressen aufgelöst. Den eigentlich gewünschten Hostnamen bekommt der Server bislang erst nach dem Herstellen einer verschlüsselten Verbindung in Form des HTTP-Host-Headers zu Gesicht. Ohne diese Information weiß der Server aber nicht, ob er nun das Zertifikat für `example.com` oder `example.org` präsentieren soll, weshalb man pro IP-Adresse und

TCP-Port auf ein einziges SSL-Zertifikat beschränkt ist.

SNI schafft dieses Problem aus der Welt, und zwar durch eine Erweiterung der Client-Hello-Nachricht. RFC 3546, veröffentlicht im Juni 2003, definiert ein neues „Extended ClientHello“, das einen generischen Mechanismus für die Übermittlung zusätzlicher Parameter vorsieht: die sogenannten TLS-Erweiterungen. Eine dieser Erweiterungen ist die Server Name Indication, inzwischen findet sich in der entsprechenden IANA-Registry bereits ein gutes Dutzend weiterer davon, etwa `max_fragment_length`, `user_mapping` oder `SessionTicket`. Wenn der Client den gewünschten Servernamen in einem erweiterten ClientHello mitschickt, kann der Server ohne Weiteres das passende Zertifikat herausuchen und dem Client präsentieren.

Ein RFC mit einer Protokollerweiterung allein hilft aber wenig, wenn niemand sie implementiert. Auch im Fall von SNI mahnten die Mühlen langsam. Während viele aktuelle Browser die SNI-Erweiterung beim Verbindungsaufbau schon seit einiger Zeit mitschicken, wächst das Angebot an Webserver-Software, die mit dieser Funktion aufwarten kann, erst nach und nach. Die Chancen stehen jedoch gut, dass sich diese Technik etabliert und das „Verschwenden“ dedizierter IP-Adressen für SSL-Server der Vergangenheit angehört. Einen Stolperstein hat die Sache aller-

dings: Der Internet Explorer unterstützt unter Windows XP kein SNI. Doch dazu später mehr.

Indianergeburtstag

Jüngster Vertreter bei den SNI-fähigen Webservern ist der Apache-HTTP-Server – und zwar seit Version 2.2.12, die Ende Juli erschienen ist. Das mitgelieferte `mod_ssl`-Modul bietet nun SNI-Unterstützung und steht daher im Zentrum dieses Artikels, andere Lösungen auf Serverseite beleuchtet der Kasten „Alternativen zu Apache und `mod_ssl`“. IIS-Administratoren haben zurzeit leider das Nachsehen: Auch die in Windows Server 2008 R2 enthaltene Version 7.5 kann nicht mit SNI aufwarten.

Wer bisher versuchte, mit `mod_ssl` mehrere `VirtualHost`-Direktiven für dieselbe IP-Adresse und denselben TCP-Port zu konfigurieren, bekam in der `ErrorLog`-Datei die ziemlich gestrenge Warnung „You should not use name-based virtual hosts in conjunction with SSL!“ präsentiert. Spätestens beim Testen der einzelnen virtuellen Server musste er dann feststellen, dass Apache in jedem Fall nur das im ersten `VirtualHost`-Block definierte Zertifikat an den Client übermittelte. In bestimmten Fällen lässt sich zwar mit dieser Einschränkung leben, den Segen der Apache-Entwickler hatten solche Lösungen jedoch nie.

Seit Version 2.2.12 hat sich die Situation nun entscheidend gebessert: Nach längeren Anstrengungen ließ sich das Apache-`httpd`-Team überzeugen, einen unter Beteiligung des Autors entwickelten Patch in die offizielle Distribution aufzunehmen und `mod_ssl` so um SNI zu erweitern. Die Konfiguration des neuen Features ist dabei transparent, es sind also keine zusätzlichen Konfigurationsdirektiven erforderlich. Ganz von alleine wird `mod_ssl` allerdings nicht SNI-fähig: zwingende Voraussetzung ist eine Version der OpenSSL-Programm-Bibliothek, die mit TLS-Erweiterungen umgehen kann. Seit der im Oktober 2007 erschienenen Version 0.9.8f ist diese Funktion durch eine Option beim Kompilieren verfügbar, ab Version 0.9.8j sind TLS-Erweiterungen standardmäßig aktiviert.

Linux-Distributionen wie Fedora, Ubuntu, Suse und so weiter werden in den nächsten Monaten ihre Apache-Pakete wohl auf Version 2.2.12 oder höher aktualisieren. Dies bedeutet aber nicht in jedem Fall, dass sie auch gleich über SNI verfügen – entscheidend ist nach wie vor die eingesetzte OpenSSL-Version. Am schnellsten beantwortet einer der folgenden Shell-Befehle die Frage, ob eine im System bereits vorhandene `mod_ssl`-Version die gewünschte Option enthält:

```
grep -cU TLS_SNI /usr/lib/httpd/modules/mod_ssl.so
```

Die genauen Dateinamen lauten je nach verwendetem Betriebssystem leicht anders (bei Ubuntu etwa `/usr/lib/apache2/modules/mod_ssl.so`). Handelt es sich um einen monolithisch kompilierten `httpd`-Daemon, so ist das folgende Kommando auszuführen:

```
grep -cU TLS_SNI /usr/sbin/httpd
```

Mindestens einer dieser Befehle sollte dann als Resultat einen Wert größer 0 produzieren – andernfalls steht vor ersten Tests mit SNI zunächst etwas Handarbeit auf dem Programm. Das Kompilieren einer geeigneten Apache-Version ist jedoch keine Hexerei, wie die folgenden, auf eine Unix-Umgebung bezogenen Hinweise zeigen.

Zubereiten

Als erstes prüft man, ob die im System vorhandene OpenSSL-Version bereits die notwendige Unterstützung für TLS-Erweiterungen bietet:

```
grep -cU SSL_get_servername /usr/lib/libssl.so
```

Ergibt das einen Wert größer als 0, so kann es direkt mit dem Kompilieren von Apache weitergehen. Im anderen Fall ist zunächst die Herstellung geeigneter OpenSSL-Bibliotheken erforderlich, was sich nach dem Herunterladen der aktuellen Sourcen von www.openssl.org mit den folgenden Befehlen bewerkstelligen lässt:

```
tar xzf openssl-0.9.8k.tar.gz
cd openssl-0.9.8k
./config no-shared no-dso --openssldir=/etc/pki/tls
make build_libs
ln -s . lib
```

Die Angabe von `--openssldir` bezieht sich auf das standardmäßig verwendete Konfigurationsverzeichnis und ist nicht zwingend erforderlich; jedoch bietet es sich an, dasselbe Verzeichnis wie eine bereits installierte Version zu verwenden (`openssl version -d` gibt darüber Auskunft). Die Optionen `no-shared` und `no-dso` sorgen dafür, dass der Compiler nur statische Bibliotheken ohne zusätzliche Abhän-

gigkeiten erzeugt – auf diese Weise spart man sich spätere Unverträglichkeiten mit einer bereits vorhandenen `libssl.so`.

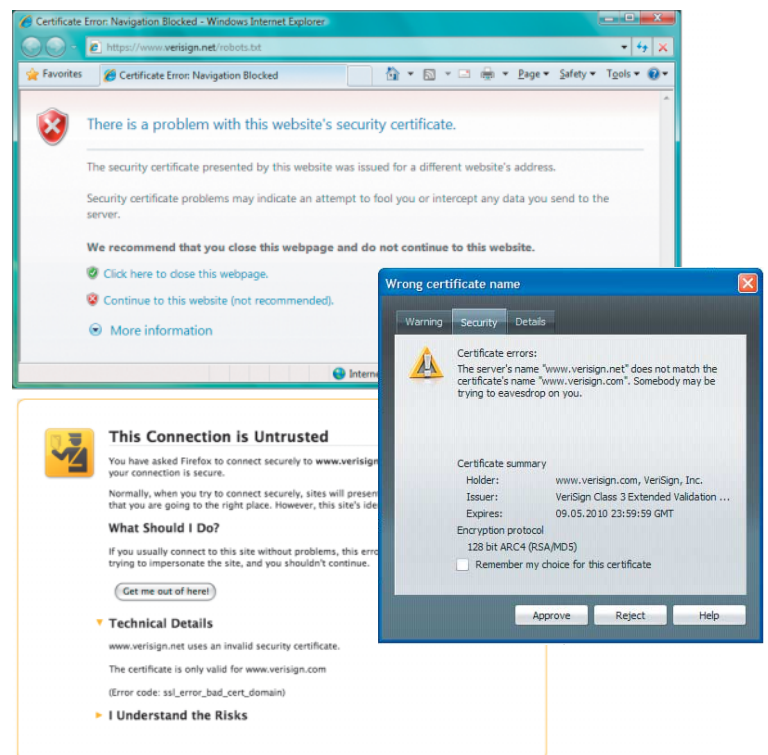
Liegen passende OpenSSL-Bibliotheken vor, so geht es im zweiten Schritt um Apache selber (Download über www.apache.org/dist/httpd). Hier ist das Angebot an verfügbaren Konfigurationsoptionen zwar sehr umfangreich, grundsätzlich reicht aber die folgende Form:

```
tar xzf httpd-2.2.14.tar.gz
cd httpd-2.2.14
./configure --enable-mods-shared=all \
--enable-ssl --with-ssl=../openssl-0.9.8k
make
```

Die Option `--enable-ssl` ist in jedem Fall notwendig, während `--with-ssl` dazu dient, die gegebenenfalls zuvor kompilierte OpenSSL-Version einzubinden (unter der Annahme, dass OpenSSL- und Apache-Sourcen im gleichen Verzeichnis liegen). Das abschließende `make` erzeugt eine ausführbare `httpd`-Datei mit rund 50 Modulen, inklusive des gewünschten `mod_ssl.so`.

Anrichten

Der mit Root-Rechten ausgeführte Befehl `make install` befördert alle Dateien nach `/usr/local/apache2`; wer ein anderes Verzeichnis vorzieht, kann dies beim `configure`-Befehl durch die Option `--prefix` steuern. Von einem Austausch der vom Betriebssystem-Hersteller oder Distributor gelieferten Version durch eine händisch kompilierte Variante ist wegen möglicher Unverträglichkeiten (Konfigurationsoptionen, Module) grundsätzlich abzuraten – nicht zuletzt deshalb, weil ein späteres System-Update die selber kompilierte Ver-



Wählt der Server das falsche Zertifikat aus, so sind die berüchtigten Warnhinweise im Browser die Folge.

sion wieder überschreiben würde. Zusätzlich muss man im Hinterkopf behalten, dass diese Varianten vom automatischen Update einer Linux-Distribution abgekoppelt sind. Das bedeutet, veröffentlichte Sicherheits-Updates jedesmal selbst einpflegen und den Code neu übersetzen zu müssen.

Als nächstes geht es an die Konfiguration von mod_ssl. Die Standardinstallation von Apache erzeugt dafür zwar eine separate Datei (/usr/local/apache2/conf/extra/httpd-ssl.conf), die dazugehörige Include-Direktive bleibt allerdings in der httpd.conf auskommentiert. Für erste Tests mit SNI erfüllen daher auch die folgenden, am Schluss von /usr/local/apache2/conf/httpd.conf eingefügten Direktiven ihren Zweck:

```
SSLRandomSeed startup file:/dev/urandom 512
SSLRandomSeed connect file:/dev/urandom 512
SSLMutex default
SSLSessionCache shmcb:logs/ssl_scache
SSLProtocol all -SSLv2
SSLCipherSuite TLSv1:!LOW:!EXP:!NULL
Listen 443
NameVirtualHost *:443
<VirtualHost *:443>
    ServerName alice.example.com:443
    SSLCertificateFile conf/alice.crt.pem
    SSLCertificateKeyFile conf/alice.key
    SSLEngine on
    DocumentRoot htdocs/alice.example.com
</VirtualHost>
<VirtualHost *:443>
    ServerName bob.example.org:443
    SSLCertificateFile conf/bob.crt.pem
    SSLCertificateKeyFile conf/bob.key
    SSLEngine on
    DocumentRoot htdocs/bob.example.org
</VirtualHost>
```

Wer bereits namensbasierte virtuelle Hosts mit Apache konfiguriert hat, dürfte bei einem Blick auf diese Konfiguration erfreut feststellen, dass keine neuen Optionen oder sonstigen Kniffe notwendig sind, um SNI in Betrieb zu nehmen. Die bisher benutzten Direktiven behalten ihre Gültigkeit auch für namensbasierte VirtualHosts mit SSL. Die abge-

bildete Konfiguration lässt sich zwar auch an eine frühere Apache-Version verfüttern, doch der entscheidende Unterschied besteht bei SNI darin, dass die SSLCertificateFile- und SSLCertificateKeyFile-Direktiven nun innerhalb jedes VirtualHost-Blocks ihre Wirkung entfalten.

Neu eingeführt hat Version 2.2.12 die Option SSLStrictSNIVHostCheck, die aber standardmäßig abgeschaltet ist. Ihre Anpassung dürfte nur in sehr speziellen Situationen notwendig sein: Ist sie aktiviert, so sperrt mod_ssl alle Clients, die beim Handshake keine SNI-Erweiterung mitsenden, vom Zugriff auf die namensbasierten VirtualHosts aus – entweder auf einzelne oder auch auf alle.

Die Werte für ServerName, SSLCertificateFile, SSLCertificateKeyFile und DocumentRoot sind dem eigenen Setup entsprechend anzupassen, allenfalls ist SSLCertificateChainFile zu ergänzen (wenn Intermediate-CA-Zertifikate im Spiel sind). Wer gerade keine geeigneten Zertifikate zur Hand hat und mit den vorgeschlagenen Namen experimentieren möchte, kann mit den folgenden OpenSSL-Befehlen ein selbstsigniertes CA-Zertifikat sowie zwei jeweils für 30 Tage gültige Testzertifikate erzeugen:

```
: cd /usr/local/apache2/conf
openssl req -new -x509 -nodes -sha1 \
    -subj "/CN=SNI Demo CA" -keyout ca.key -out ca.crt.pem \
openssl req -new -nodes -subj /CN=alice.example.com \
    -keyout alice.key | openssl x509 -req -CA ca.crt.pem \
    -CAkey ca.key \
    -CAcreateserial -sha1 -out alice.crt.pem
openssl req -new -nodes -subj /CN=bob.example.org \
    -keyout bob.key | openssl x509 -req -CA ca.crt.pem \
    -CAkey ca.key \
    -sha1 -out bob.crt.pem
```



Servieren

Nach dem Ausführen von /usr/local/apache2/bin/httpd empfiehlt sich zuerst ein kurzer Blick in die ErrorLog-Datei, wo sich nun eine Zeile der Art „[warn] Init: Name-based SSL virtual hosts only work for clients with TLS server

name indication support (RFC 4366)“ finden sollte. Ist dies nicht der Fall, so fehlt dem mod_ssl-Modul entweder SNI, es sind mindestens zwei namensbasierte SSL-Hosts (mit identischer VirtualHost-Direktive) definiert oder aber der globale LogLevel steht auf error oder niedriger.

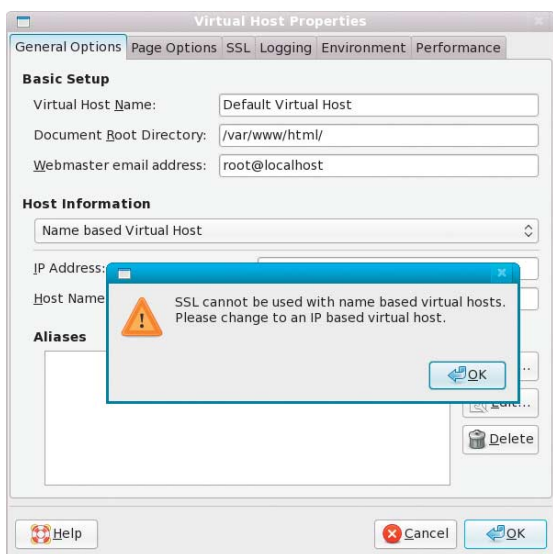
Sieht in der ErrorLog-Datei alles wie erwartet aus, so kann es ans Testen mit einem Client gehen. Hier ist die verfügbare Auswahl in den letzten Jahren erfreulich breit geworden: Im April 2005 übernahm Opera 8.0 die Vorreiterrolle, rund eineinhalb Jahre später gefolgt von Firefox 2.0, und im Januar 2007 gesellte sich Internet Explorer 7 unter Windows Vista hinzu. Bei Internet Explorer ist die Hervorhebung des Betriebssystems nicht unwesentlich: die SNI-Unterstützung ist in diesem Fall nicht im Browser, sondern im Betriebssystem implementiert (konkret: in der Bibliothek schannel.dll). Leider hat Microsoft diese Funktion in Windows XP und älteren Windows-Versionen nicht nachgerüstet, so dass Benutzer dieser Betriebssysteme bedauerlicherweise auch dann nicht von SNI profitieren können, wenn sie Internet Explorer Version 7 oder höher einsetzen. Dies stellt für Webseiten-Betreiber einen ziemlich dicken Pferdefuß dar. Zwar nutzen auch unter XP viele Anwender etwa den Firefox, dennoch verbleiben viele IE-Nutzer. Heise online besuchten beispielsweise im September 2009 71 Prozent der Anwender mit nativ SNI-fähigen Browsern und nur 19 Prozent mit MS-Browsern. Dafür lag aber der Anteil von Windows XP bei 53 Prozent.

Damit der Test im Browser auf Anhieb klappt, sind vor der Eingabe der URL zwei Dinge sicherzustellen: eine korrekte Namensauflösung (via DNS oder über Einträge in der hosts-Datei) und – bei Verwendung von Testzertifikaten nach dem oben dargestellten Verfahren – die Installation des selber erzeugten Root-Zertifikats (ca.crt.pem) im entsprechenden Zertifikatspeicher. Und dann kann es losgehen: Ein Aufruf von https://alice.example.com, https://bob.example.org oder den selber gewählten Servernamen sollte nun erstens keine Warnmeldungen produzieren und zweitens im Browser jeweils ein individuelles Zertifikat anzeigen, auch wenn beide Namen auf dieselbe IP-Adresse auflösen.

Googles Chrome und Apples Safari gehören inzwischen auch zu den Browsern mit SNI-Unterstützung. Bei ihnen präsentiert sich die Lage jedoch ähnlich wie beim Internet Explorer: Sie stützen sich für TLS auf die vom Betriebssystem zur Verfügung gestellte Funktion. Das heißt, um sie nutzen zu können, sind Windows Vista oder höher beziehungsweise OS X Leopard (ab Version 10.5.6) Voraussetzung. Außen vor bleiben zurzeit Benutzer von Konqueror, weil das Qt-Framework SNI noch nicht zur Verfügung stellt.

Rücksicht auf Verluste

Ob ein Client beim TLS-Handshake tatsächlich eine SNI-Erweiterung mitgeschickt hat,



Noch verhindert Red Hats Konfigurations-tool das Aufsetzen von namensbasierten virtuellen Hosts.

lässt sich serverseitig, etwa in einem CGI-Skript, durch die neue Umgebungsvariable `SSL_TLS_SNI` feststellen: Ist sie vorhanden, so enthält sie den via SNI-Erweiterung übermittelten Servernamen. Die SNI-Unterstützung des Browsers lässt sich auch auf folgender Seite testen: <https://sni.velox.ch>.

Die Umgebungsvariable ist jedoch nicht nur für diagnostische Zwecke nützlich, sondern kann beispielsweise auch in Kombination mit `mod_rewrite` verwendet werden, um Clients ohne SNI-Fähigkeit auf eine Seite mit Informationen über empfohlene Browser-Alternativen umzuleiten. In der globalen Apache-Konfiguration definiert man zu diesem Zweck etwa Regeln der folgenden Art:

```
RewriteCond %{SSL:SSL_TLS_SNI} =""
RewriteCond %{HTTP:Host} !=alice.example.com
RewriteRule .* https://alice.example.com/sni-info.html [L]
```

und aktiviert sie innerhalb der jeweiligen VirtualHost-Blocks (denjenigen ab Position zwei und höher) mit

```
RewriteEngine on
RewriteOptions inherit
```

Zwar führt dies weiterhin zu einer Zertifikats-Warnmeldung bei Browsern ohne SNI, aber immerhin erhält der Website-Besucher auf diese Weise Informationen über die Ursache der Warnung und mögliche Lösungen. Um vorerst nur aufzuzeichnen, wie viele der aktuellen Besucher bereits Browser mit SNI-Unterstützung verwenden, lässt sich der Inhalt der Erweiterung zwecks späterer Auswertung auch einfach durch die Verwendung von `%{SSL_TLS_SNI}` in einer LogFormat-Direktive protokollieren.

Wer befürchtet, durch die Umstellung auf SNI im Moment zu viele potenzielle Besucher wegen fehlenden Browser-Supports auszuschließen, für den gibt es zwei Optionen, um trotzdem IP-Adressen einzusparen – beide verdienen allerdings eher das Prädikat „behefsmäßig“. Gehören alle virtuellen Server

Ob ein Client beim TLS-Handshake tatsächlich eine SNI-Erweiterung mitgeschickt hat, lässt sich serverseitig durch die neue Umgebungsvariable `SSL_TLS_SNI` feststellen.



zur selben DNS-Domain, kann ein Wildcard-Zertifikat für Ruhe vor SSL-Warnhinweisen sorgen. Handelt es sich dagegen um Hostnamen aus verschiedenen Domains, so hilft möglicherweise ein Zertifikat mit mehreren `subjectAltName`-Einträgen. CA-Betreiber bieten solche Zertifikate oft auch unter der Bezeichnung UCC (Unified Communications Certificate) an, weil sie vor allem für den Einsatz bei Exchange 2007 und dem Office Communications Server 2007 vorgesehen sind.

Beide Lösungen haben jedoch einen Pferdefuß. Wildcard-Zertifikate unterlaufen ein wesentliches Ziel von SSL, nämlich die eindeutige Identifikation des Kommunikationspartners: für den Besucher sind die zwei Websites `gut.example.net` und `boese.example.net` bei Verwendung eines Wildcard-Zertifikats für `*.example.net` nicht mehr unterscheidbar, der Browser zeigt in beiden Fällen jeweils dasselbe Zertifikat. Gelangt ein Angreifer zudem unbemerkt an den privaten Schlüssel eines Wildcard-Zertifikats, so kann er vergleichsweise einfach weitere Websites mit neuen Hostnamen in der betreffenden Domain aufsetzen (gut geeignet etwa für Phishing-Seiten, inklusive perfekter SSL-Tarnung).

Bei Zertifikaten mit einer `subjectAltName`-Extension, die jeden Servernamen explizit aufführt, besteht die Wildcard-Problematik zwar nicht. Allerdings sind solche Zertifikate

nur bei einer kleineren, relativ stabilen Zahl von Hostnamen noch vernünftig handhabbar. Wenn es um SSL-Zertifikate mit überprüftem Organisationsnamen geht, dann kommt außerdem die Einschränkung hinzu, dass sämtliche Domains demselben Inhaber zugeordnet sein müssen.

Ausblick

Gut Ding will Weile haben – bei der Implementation der SNI-Erweiterung für TLS ist das nicht anders. Vor sechs Jahren erstmals in einem RFC spezifiziert, wächst die Unterstützung für diese Technik aber inzwischen doch kontinuierlich. Was die Browser-Seite betrifft, so hemmt leider Windows XP den Fortschritt zurzeit noch am empfindlichsten, da Internet Explorer, Safari oder Chrome auf SNI-Unterstützung durch das Betriebssystem angewiesen sind und diese erst ab Vista zur Verfügung steht. Serverseitig sorgt jedoch Apache mit `mod_ssl` seit Version 2.2.12 nun hoffentlich für weiteren Auftrieb. Und selbst wenn IPv6 dereinst das Problem der Adressknappheit entschärfen wird: jedem SSL-Server nur wegen Einschränkungen einer Protokollspezifikation aus den neunziger Jahren eine eigene Adresse spendieren zu müssen, ist auf Dauer auch kein befriedigender Zustand – SNI löst dieses Problem weit eleganter. (dab)

Alternativen zu Apache und `mod_ssl`

Neben `mod_ssl`, das seit Apache 2.0 offiziell zur `httpd`-Distribution gehört, stehen auch andere Lösungen zur Verfügung, um mit Hilfe von SNI mehrere SSL-Websites unter derselben IP-Adresse zu betreiben. Die wohl erste serverseitige Implementation stammt von Paul Querna: Er entwickelte gegen Ende 2004 `mod_gnutls` – weil er es nach eigenen Aussagen leid war, Fehler in `mod_ssl` zu beheben. Mit Version 0.2.0, veröffentlicht im April 2005, bot `mod_gnutls` erstmals SNI. Für längere Zeit galt das Modul als experimentell, vor knapp zwei Jahren stieß aber mit Nikos Mavrogiannopoulos, Mitinitiator und ursprünglicher Entwickler der GnuTLS-Bibliothek, ein weiterer Autor hinzu. Seit Version 0.4.0 sind daher die entsprechenden Warnhinweise auf der Website verschwunden, und mittlerweile stellt auch der

eine oder andere Linux-Distributor fertige Pakete von `mod_gnutls` bereit. Gegenüber `mod_ssl` streichen die Entwickler vor allem den deutlich geringeren Codeumfang heraus, allerdings hat sich die Zeilenzahl gegenüber der Urversion seither auch bereits verdoppelt.

Nächster Server im Bunde war chronologisch gesehen `nginx` („engine x“), die Igor Sysoev ursprünglich für das russische Suchportal Rambler entwickelte. Seit der im Juni 2007 veröffentlichten Version 0.5.23 bietet `nginx` SNI-Funktion, sofern eine geeignete OpenSSL-Version installiert ist. Wer eine schlanke und leistungsfähige Alternative zu Apache sucht und auf `mod_ssl`-Spezialitäten wie etwa die `SSLRequire`-Direktive verzichten kann, für den ist

`nginx` zweifellos eine ernstzunehmende Option.

Darüber hinaus bietet sich Cherokee an, ein von Alvaro Lopez Ortega entwickelter Webserver, der mit Version 0.9.0 im September 2008 erstmals SNI-Unterstützung mitbringt. Cherokee lässt sich entweder mit OpenSSL oder GnuTLS kompilieren und fällt vor allem durch das mitgelieferte GUI-Konfigurations-tool auf, das die einzige offiziell unterstützte Methode für die Serverkonfiguration bildet (das Format der Konfigurationsdatei ist zwar dokumentiert, ein manuelles Editieren aber nicht vorgesehen). Für den von Jan Kneschke entwickelten `lighttpd` schließlich existieren zwar Patches im Bug-Tracking-System, jedoch haben sie bisher keinen Eingang in ein offizielles Release gefunden.

