http://www.therealgang.de/

Titel :	APACHE Zusammenfassung
Author :	Michaela Uhl
Kategorie :	Webserver-Allgemein

Zusammenfassung Apache Script

2. http – Protokoll

Webserver und Clients kommunizieren dank des Hypertext Transfer Protokolls (http). Übermittlung der Daten erfolgt nach dem Request- / Respons- Verfahren. Im Gegensatz zum FTP-Protokoll keine mehrstufige Hand-Shake-Phase beim Verbindungsaufbau. http ist grundsätzlich abwärtskompatibel (aktuelle Version http/1.1).

2. 2. Ablauf einer http – Verbindung

Kommunikation zwischen Client und Webserver erfolgt durch Austausch von http-Nachrichten. Im Standardfall stellt der Client-Rechner eine TCP –Verbindung auf Port 80 um Webserver her (z.B.: über Browser). Anschließend schickt der Client eine Anfrage (so genannter Request) an den Server. Der Request enthält unter anderem die Art der Anfrage (Methode) den URL (Universal Ressource Locator) und die Protokollversion. Art der Anfrage kann abhängig von der Protokollversion (http/1.0. oder http/1.1) GET, POST, HEAD, PUT, DEL, TRACE sein. Nachdem der Request vom entfernten Webserver registriert wurde, durchsucht der Server sein Dateisystem nach der angeforderten Datei und lädt sie. Falls die Aktion erfolgreich war, schickt der Server eine Antwort (so genannter Response), die die gewünschten Informationen sowie den MIME Type der Datei enthält. Andernfalls wird eine Fehlermeldung übergeben. Aus dem MIME Type schließt der Client, was er mit den empfangenen Daten anfangen soll (z.B.: Anzeigen im Browser).
Um die Leistungskapazitäten zu schonen, wird die Verbindung direkt nach dem Senden des Response beendet. Dieses Vorgehen ist allerdings nicht durch das Protokoll definiert.
Client und Server müssen in der Lage sein, mit Verbindungsabbrüche während des Datenaustauschs umzugehen. Damit ist der Request, egal in welcher Phase er sich befindet, ebenfalls beendet (→ zustandsloses Protokoll)

2.2.1. http-Requests

Ein Request besteht aus

- Methode (z.B.: GET, PUT oder DELETE) Server antwortet auf jeden Request mit Informationen egal ob Methode zulässig ist, oder nicht.
- URL: Client muss einen absoluten URL angeben um über Proxy-Server laufen zu können. Nach Zugriff auf eine Quelle reicht die Angabe von relativen URLs.
- Request – Header – Felder welche die Uhrzeit, die Browserversion und MIME Type enthalten

Erste Zeile eines Requests allgemein:
METHOD URL http/Version

2.2.2. http-Methoden

Methoden bestimmen die Aktionen der Anforderungen. Die aktuelle http-Spezifikation sieht acht Methoden vor: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE und CONNECT

- *GET - Methode:* Dient zur Anforderung eines Dokuments oder einer anderen Quelle. Unterscheidung zwischen *Conditional GET* und *Partial GET*.
  Beim *Contitional GET* ist die Anforderung von Daten an Bedingungen geknüpft. Mit Hilfe dieser Bedingungen kann die Netzbelastung deutlich verringert werden, da nur

die wirklich benötigten Daten übertragen werden. Dies nutzen z.B. Proxy-Server um die mehrfache Übertragung von Daten im Cache zu verhindern.
Die Partielle GET – Methode verfolgt das gleiche Ziel. Verwendung des Range-Header-Feldes, das nur Teile der Daten überträgt, die der Client noch verarbeiten kann (Verwendung zur Wiederaufnahme eines unterbrochenen Datentransfers).

2.2.3. http-Response

Aufbau eines http-Response ist ähnlich wie der des Requests (Struktur aus Header und Nachrichten-Body gleich). Der Server übermittelt zunächst die http-Version der Nachricht, der zweite Eintrag ist die Statusmeldung („200 ok“ bedeutet keine Fehler aufgetreten). Wichtig für die weitere Bearbeitung durch den Client ist:

- Content Type: beschreibt den MIME Type der im Datenbereich übermittelten Dateien
- Content Length: Server gibt die Länge der Daten in Byte an (Einsatz des Feldes nicht zwingend vorgeschrieben)

Erste Zeile eines Responses allgemein:
http/Version STATUS-CODE REASON-Zeile

Kategorien von Response-Codes:
1xx → Informelle Meldungen: Request erhalten, Bearbeitung wird durchgeführt
2xx → Erfolg: Request wurde erfolgreich erhalten, verstanden, angenommen
3xx → Weiterleiten: weitere Aktionen müssen eingeleitet werden, damit ein Request vollständig bearbeitet werden kann
4xx → Clientfehler: Der Request enthält ungültige Syntax oder kann nicht bearbeitet werden
5xx → Serverfehler: Der Server kann einen gültigen Request nicht bearbeiten

5.Konfiguratonsdateien

In der Regel finden sich die Server-Konfigurationen des Apache im Verzeichnis **/ect/httpd/conf**, wobei alle Konfigurationsanweisungen in der Datei **httpd.conf** gespeichert sind.

6. Modulkonfiguration

6.1. Dynamic Shared Objects →DSO

Der Webserver Apache ist mittlerweile so groß und komplex geworden, dass er nicht mehr monolithisch in einem Block aufgebaut ist, sondern modular. Der eigentliche Kernserver (CoreServer) kennt nur die grundlegenden Eigenschaften eines Webservers. Zusätzliche Fähigkeiten wie z.B. Authentifizierung, PHP-Scripts, Session-Management werden mit Hilfe von Modulen eingebaut. Durch DSO- Funktionalität können Module dynamisch in den Apache eingebunden und wieder entfernt werden. Hierfür muss der Apache nicht neu übersetzt werden, aber das DSO-Modul muss statisch im Apache eingebunden sein. Interessant ist, dass DSO-Module zur Laufzeit des Apaches verändert werden können ohne evtl. aktive Verbindungen unterbrechen zu müssen.

7 Konfigurationsanweisungen

Konfigurationsdatei http.conf ist in Teilbereiche aufgeteilt:

- globale Einstellungen

- Einstellungen des Hauptservers
- Einstellungen für die virtuellen Server

7.1. Globale Einstellungen

Unter „globale Einstellungen“ werden die Bedingungen verstanden, unter denen der Apache-Webserver arbeiten soll. Zu dieses Arbeitsumgebung gehören u.a.:

- *Angaben zum Servertyp*: Der httpd-daemon kann durch den inetd gestartet werden oder als stand-alone laufen. Eher üblich ist die stand-alone Version, da bei inetd bei jeder anfrage der Apache neu gestartet werden muss.
- *Definition des Hauptverzeichnisses*: Ist das Root-Verzeichnis für den Server, in dem sich die Verzeichnisse für die Log-Files, Konfigurations- und Protokolldateien befinden.
- Ein paar Festlegungen für Zugriffszeiten und –zahlen
- Die Liste der zur Laufzeit einzubindenden Module

7.2. Einstellungen des Hauptservers

- *Festelegungen des, vom Server beanspruchten, Ports*: (für des http-Protokoll in der Regel Port 80) Es kann sinnvoll sein einen anderen Port für ein lokales LAN zu wählen, um Konflikten auszuweichen. Dann sollte es einer, der nicht für Standards vorgesehenen Ports unter 1024 sein (privilegierte Ports). Falls Apache als stand-alone laufen soll, wird die Portnummer, auf der der daemon lauschen soll, unbedingt benötigt.
- *Administratoradresse*: Mit Hilfe der Anweisung Serveradmin wird die Adresse angegeben, an die eventuelle Probleme mit dem Server gemeldet werden können.
- *Servername*: erlaubt es einen Hostnamen festzulegen über den der Webserver angesprochen werden soll. Ein Name, der hier angegeben wird, muss ein gültiger DNS-Name sein (alternativ IP – Adresse). 127.0.0.1 ist die lokale Loop-Back-Adresse für das TCP/IP Protokoll (localhost). In Apache 2.0 wurde die Port und Servername-Anweisung kombiniert. Es gibt in dieser Version nur noch die Servername-Anweisung, welche die optionale Angabe einer Portnummer erlaubt. Defaultmässig wird entweder Port 80 (http) oder 443 (https) verwendet, je nach dem ob für den jeweiligen Server SSL aktiviert wurde, oder nicht.
- *Identität des Webusers:* Der Apache – Server unter UNIX muss zunächst unter der Benutzerkennung ROOT gestartet werden, um an den Port 80 gebunden werden zu können. **(ERKLÄREN LASSEN!!!!)**
- *Pfadangaben zu Dokument-, Benutzer- und Script-Verzeichnissen:*
    - *Dokumentverzeichnis*: Wurzelverzeichnis für alle veröffentlichten Dokumente
    - *Benutzerverzeichnis*: Webserver enthalten häufig URLs bei denen der URL-Pfad mit einer Tilde beginnt. Wie der Apache verwenden die meisten Webserver diese Art der URLs um Benutzerverzeichnisse zu kennzeichnen. Der Name nach der Tilde ist der Accountname des jeweiligen Benutzers (~benutzername). Dadurch kann man den Benutzern eines UNIX-Systems ermöglichen selbst Seiten im Netz anzubieten, in dem man innerhalb des HOME-Verzeichnisses des Users ein Unterverzeichnis anlegt, welches dann vom Webserver verwaltet wird. (in der Regel heißen diese Verzeichnisse public_html).
- *Verzeichnisindex:* d.h. der Name (die Namen) der Datei, die in einem Verzeichnis aufgerufen werden soll, wenn nur der Verzeichnissname angegeben wurde.
- Formatierung der Protokolldateien

7.3. Virtuelle Server

Unterstützung von virtuellen Servern eines Webservers meint, dass er gleichzeitig unter mehreren Hostnamen agieren und dabei jeweils ein eigenes Dokumentverzeichnis, Log-Files usw. verwalten kann. Um mehrere Websites gleichzeitig verwalten zu können, braucht man

also weder mehrere Rechner noch mehrere Instanzen des Apache. Es gibt zwei Arten von virtuellen Servern:

- IP-basierte virtuelle Server verlangen die Vergabe einer IP-Adresse pro Server
- Namensbasierte virtuelle Server (aus verschiedenen Gründen sinnvoll→ nicht genug IP-Adressen, Zeitersparnis des Administrators ect.)
- Port-basierte virtuelle Server (prinzipiell auch IP oder Namensbasierte Server nur mit unterschiedlichen Portnummern)

7.3.1. IP-basierte virtuelle Hosts

Ursprüchliche Form von virtuellen Servern setzt voraus, dass jeder virtuelle Server eine eigene IP-Adresse bekommt. Da der Client (bei http/1.0) nicht kenntlich machen kann, auf welche Website (Hostname) er zugreifen will, verwendet man für jeden virtuellen Server eine eigene IP-Adresse. Der Web-Server erkennt anhand der IP-Adresse, welches Dokument er zurückliefern soll.

7.3.2 Namensbasierte virtuelle Hosts

Virtuelle Server bei denen keine eigene IP-Adresse vergeben werden muss. Im Nameserver muss jediglich ein Hostname –Alias (CNAME) eingetragen werden. Die mehrfache Verwendung von IP-Adressen ist aus verschiedenen Gründen sinnvoll. (s.o)
Dies funktioniert allerdings nur, wenn der Web-Client diesen Teil von http/1.1 unterstützt. Realisierung dieser namensbasierten virtuellen Hosts dadurch, dass ein Client bei jedem Zugriff den http-Header Host mitschickt, indem er den Hostnamen der Website, auf die er zzugreifen will, übermittelt. Wird der Header nicht mitgeschickt, wird der für diese IP-Adresse konfigurierte Default-Server verwendet.

7.3.2.1 <Virtuelle Hosts>

Zentrale Anweisung bei der Konfiguration von virtuellen Servern. Wird zum Einrichten und Aktivieren von virtuellen Servern benötigt. Es muss mindestens eine IP-Adresse, ein Hostname (optional mit Portnummer) angegeben werden. Die <Virtual Host> -Anweisung kann beliebig oft in der Serverkonfiguration auftauchen. Defintion des Virtuellen Hosts mit Servername, Serveradmin, DokumentRoot etc. innerhalb des Virtuellen Hosts-Abschnitt. Bei der Konfiguration eines virtuellen Servers sollte man bevorzugt IP-Adressen angeben damit beim Start der Apache keinen dnslookup machen muss, um die Adressen herauszufinden. Bei namensbasierten Servern ist zwingend erforderlich, in der servername-Anweisung einen Hostnamen anzugeben.

7.3.2.2 NameVirtualHost

NameVirtualHost - Anweisungen wird benötigt um namensbasierte Server verwenden zu können. Sollen ein oder mehrere namensbasierten Server verwendet werden, müssen die NameVirtualHost - Anweisung zur Kennzeichnung der jeweiligen IP-Adresse und ggfs. Die Portnummer angegeben werden. Bei Weglassen der Portangabe wird die Standartportnummer(meist Port 80) verwendet.
Bei einer Anfrage an den gewünschten Hostnamen werden die virtuellen Hostblöcke nach den Servernamen durchsucht, um zu wissen an welchen Virtuellen Host die Anforderung gerichtet ist. Die Position der NameVirtualHost innerhalb der Serverkonfiguration ist nicht relevant. Aber dir Reihenfolge der zugehörigen <VirtualHost> - Abschnitte ist wichtig.

8. .htaccess – Server-Reaktionen kontrollieren

Konfigurationsdatei zur Verzeichniskonfiguration -> typischerweise .htaccess als Dateiname genutzt. Wird verwendet um z.B. nur bestimmten Benutzern Zugang zu bestimmten Daten zu erlauben. .htaccess bietet also einen „richtigen“ Passwortschutz. Es ist möglich ganze Benutzerkreise automatisch (ohne Passwortzugang auszusperren oder alle bis auf bestimmte auszusperren). Automatische Weiterleitung oder eigene Regelung für den Fall von http-Fehlermeldungen können geschaffen werden.

Weitere Möglichkeiten:

- Alternative Inhalte von Webseiten wenn bestimmte Bedingungen erfüllt sind.
- Komprimierte Datenübertragung auf den Browser

.htaccess – Dateien sind deshalb nicht in der zentralen Webserverkonfiguration, damit auch Benutzer von Web-Projekten Zugriff auf diese Konfigurationsmöglichkeiten haben. → .htaccess – Dateien werden da abgelegt, wo die HTML – Dateien abgelegt werden und gelten auch immer nur für dieses Verzeichnis und seine Unterverzeichnisse. (Soll in einem Unterverzeichnis eine andere .htaccess – Datei Gültigkeit haben, so muss dort wiederum eine neue Datei eingefügt werden.
Verwaltung der Dateien entweder per Telnet oder SSH – Zugang direkt auf den Webserver oder bei FTP-Zugang auf den lokalen Rechner.

8.2. Verzeichnisse und Dateien Passwort schützen

.htaccess – Dateien sind verzeichnisspezifisch, d.h. die Datei wird in dem Verzeichnis, dessen Daten geschützt werden sollen, abgespeichert. Geschützt werden können:

- gesamte Verzeichnisse mit allen Unterverzeichnissen
- nur bestimmte Dateien
- nur bestimmte Dateitypen

Dies kann wahlweise für einzelne Benutzer oder ganze Benutzergruppen eingerichtet werden. Um Dateien mit einem Passwort zu schützen, wird eine Datei mit den Benutzern und deren Passwörtern benötigt. Möchte man mit Benutzergruppen arbeiten, benötigt man eine zusätzliche Datei mit der Festlegung der Benutzergruppen.

8.3. Schutz von Dateien , Dateitypen oder Zugriffsmethoden

Schutzmechanismen, die mit .htaccess erstellt werden, sind auf http – Ebene wesentlich sicherer als solche, die mit CGI – Scripts oder JavaScript erstellt wurden.
Der Schutz gilt allerdings nur, wenn Web-Browser oder mehrere Web-Clients über den Web-Server auf Dateien zugreifen. Der Schutz ist nicht gewährleistet, wenn andere Protokollen (wie FTP) benutzt werden.

8.4. IPs, IP- Bereiche oder Namensadressen zulassen/ausschließen

Zentrale Voreinstellung: Zugriff für alle Benutzer erlaubt
Daher ist es sinnvoll eingeschränkte Verbote zu definieren. Es müssen nicht komplette IP-Adressen oder Namensadressen angegeben werden (z.B.: 192.168.) Es gibt die gleichen Möglichkeiten der Einschränkung wie beim Passwortschutz.

8.5. Verzeichnis-Optionen einstellen

Ändern der zentralen Verzeichnis Verzeichnisoptionen, die in der Konfigurationsdatei des Webservers eingestellt sind, für ein Verzeichnis und seine Unterverzeichnisse.
Voraussetzung: *AllowOverride* muss in der zentralen Webserver-Konfiguration auf *ALL* gesetzt sein.
Einige Optionen sind:

- ***+ExecCGI*** erlaubt das Ausführen von CGI – Scripts
- ***+Includes*** erlaubt das ausführen von Server Side Includes im Verzeichnis, falls es zentral verboten ist, ***-Includes*** verbietet es, falls es zentral erlaubt ist
- ***+IncludesNOEXEC*** erlaubt das Ausführen von Server Side Includes, die kein CGI – Script ausführen, ***-IncludesNOEXEC*** verbietet sie
- ***+Indexes*** erlaubt das Verzeichnis – Browsing, falls es zentral verboten ist, ***-Indexes*** erbietet es, falls es zentral erlaubt ist
- ***+MultiViews*** erlaubt das definieren alternativer Inhalte, ***-MultiViews*** verbietet es

Normaler Weise darf man im Webserver „stöbern“ falls keine Index-Datei vorhanden ist. Mit DirectoryIndex kann man andere Dateien bestimmen, die als Index-Datei fungieren sollen.
Mit ErrorDocument ändert man die Fehlerseiten, die angezeigt werden, wenn Zugriffsfehler auftreten.

8.6. Verzeichnis – Browsing einstellen

Wenn das Verzeichnis – Browsing erlaubt ist, so wird beim dem Aufruf einer Adresse wie http://www.domain.de/bilder/ die Verzeichnisstruktur angezeigt wird.
Man kann Einfluss auf die Optik der Ausgabe nehmen, da bei einer solchen Anfrage dynamisch vom Webserver HTML-Seiten erstellt werden.
Man kann sowohl die Darstellung der Verzeichnisliste (FancyIndexing), einen Beschreibungstext für jede Datei (AddDescription) oder eine eigene Symbolgrafik einsetzen (AddIcon)

9. Secure Webserver

Sichere Kommunikation ist für kommerzielle Nutzung des Internets wichtig. Gesicherte Übertragung von Daten durch Verschlüsselung und Authentifikation.
Das Problem: Zwei Kommunikationspartner (A und B) kennen sich nicht. Sie besitzen daher kein gemeinsames Wissen, das nicht auch ein Dritter (C) kennt. A und B sitzen jeder für sich in einer sicheren Umgebung, so dass niemand weiß, was sie tun. Das einzig unsichere ist der Kommunikationsweg. Er ist öffentlich und kann von Jedermann abgehört werden. C hat die Möglichkeit das „Gespräch“ von Anfang an mitzuverfolgen oder Teile davon zu verändern, ohne dass der Empfänger dies merkt.
C wird „Man in the middle“. Beim Verändern von Daten spricht man von aktivem beim Mithören ect. Von passiven Angriffen.
Ist C aktiver Angreifer, so ist es für A und B unmöglich eine sichere Verbindung herzustellen, da C immer dazwischen hängt.
Die Lösung: Es muss eine Authentifikationsmöglichkeit für A und B geben, die C nicht fälschen kann. B benötigt somit einen eindeutigen Ausweis, der von einer Dritten Person (D) ausgegeben werden muss. Diese Authetifizierungsstelle prüft die Identität des Antragsstellers bevor sie einen Ausweis ausstellt. Dies sollte nicht auf elektronischem Weg geschehen.

9.2. Verschlüsselungsverfahren

Verwenden von bestimmten Wörtern oder Zeichenkombinationen als Schlüssel. Verschlüsselung berechnet aus dem unverschlüsselten Text und dem Schlüssel den verschlüsselten Text. Entschlüsselt wird umgekehrt. Sicherheit beruht darauf, dass Schlüssel nur schwer oder gar nicht gefunden werden können.
Grundsätzlich gibt es zwei Verschlüsselungsverfahren für eine gesicherte Kommunikation im Internet.
Symmetrische Verschlüsselung: Der selbe Schlüssel wird zum Ver- und Entschlüsseln benutzt. Dieser Schlüssel muss allen Beteiligten bekannt sein, wozu der Schlüssel den Kommunikationspartnern zuerst bekannt gemacht werden muss. Daher ist dieses Verfahren ungeeignet, wenn sich die Kommunikationspartner nicht persönlich begegnen, da der Schlüssel über eine ungesicherte Leitung übertragen werden muss. Vorteil der symmetrischen Verschlüsselung ist, dass sie Daten mit hoher Geschwindigkeit ver- und entschlüsselt.
Zur Verschlüsselung von großen Datenmengen in sehr kurzer Zeit. Symmetrische Algorithmen können nur da eingesetzt werden, wo der Schlüssel nicht übermittelt werden muss, oder der Schlüssel per asymmetrische Verschlüsselung übertragen wird.
Asymmetrische Verschlüsselung: Unterschiedliche Schlüssel zur Ver- und Entschlüsslung. Bei den Schlüsseln handelt es sich um so genannte Schlüsselpaare, welche immer aus einem *private key* und einem *public key* , der allgemein zugänglich ist, bestehen. Daten, die mit dem privaten Schlüssel verschlüsselt werden, können nur mit dem öffentlichen Schlüssel entschlüsselt werden. Der Empfänger der Daten erzeugt einen öffentlichen und einen privaten Schlüssel. Den öffentlichen Schlüssel übergibt er dem Absender. Der verwendet den öffentlichen Schlüssel um die Daten zu verschlüsseln. Nur der Empfänger kann mit seinem privaten Schlüssel die Daten entschlüsseln. Nachteil ist der hohe Rechenaufwand beim Ver- und Entschlüsseln. Vorteil ist die einfache Methode des Schlüsseltauschs.

9. 3. Secure Socket Layer → SSL (sichere Verbindungsschicht)

Verwendung eines Webservers, der SSL unterstützt. SSL Protokoll verwendet sowohl symmetrische als auch asymmetrische Verschlüsselung. Asymmetrische Verbindung nur zur Übermittlung des Schlüssels verwendet.
SSL = Protokoll für sichere, verschlüsselte Übertragung von Nachrichten im Internet.
Verschlüsselung von beliebigen Diensten u.a. SMTP, POP3, FTP

9.3.1. sslmod und Open SSL

sslmod = Modul, das dem Apache SSL Funktionalitäten zur Verfügung stellt
Schnittstelle zwischen Apache und er entsprechenden SSL – Bibliothek
Vorraussetzung für sslmod ist ein korrekt installiertes OpenSSL – Paket, welches dem System ein Bibliothek der grundlegendsten kypographischen Funktionen bereit stellt.

9.4. Verbindung über SSL

Aufbau einer gesicherten Verbindung über Angabe des Schemas https statt http in der URL.
https – Verbindungen laufen über TCP, der Standardport ist 443.
SSL führt vor dem Aufbau einer Verbindung eine Initialisierung durch das so genannte Handshake – Protokoll durch. Dieses legt die Sicherheitsstufe fest und übernimmt die notwendigen Echtheitsbestätigungen für die Verbindung und handelt einen Sitzungsschlüssel für die Verschlüsselung aus.

- der Client stellt anfrage an Server und schickt von ihm unterstützte Verschlüsselungsverfahren

- Server wählt ein Verfahren aus und übermittelt dem Client sein Zertifikat mit dem öffentlichen Schlüssel
- Client generiert den Sitzungsschlüssel für ein symmetrisches Verschlüsselungsverfahren
- Verschlüsselung des Sitzungsschlüssels mit dem öffentlichen Schlüssels des Servers
- Übertragung des verschlüsselten Sitzungsschlüssels an den Server
- Entschlüsseln des Sitzungsschlüssels mit dem privaten Schlüssel auf dem Server
- Optionale Clientauthentifizierung (Client muss dabei aber ein gültiges Zertifikat haben

Kommunikation erfolgt weiterhin nur symmetrisch mit dem Sitzungsschlüssel verschlüsselt.
→ gute Lösung zwischen sicherer (sehr asymmetrischer) und schneller (symmetrischer) Verschlüssellung
Sämtliche Informationen (auch URL und Formularinhalte) werden verschlüsselt.

9.5. Host- und Client- Authetisierung

Der Client kann sich zeimlich sicher sein, wer der Server ist, da dieser ein Zertifikat bestizt. Der Server hat aber keine Möglichkeit zu prüfen, ob der Client der richtige Empfänger ist. Wenn sich nur der Server authentisieren muss, nennt man es https mit Host Authentisierung. Muss sich auch der Client authetisieren (z.b.: Home-Banking) spricht man von Client Authentisierung. Dabei muss der private Schlüssel dem Client (also der Person am Client) eindeutig zuordenbar sein.

9.6. x.509 Zertifikate

Standardformat der ITU-T für Zertifikate. Beinhaltet Namen und digitale Signatur des Ausstellers sowie Infos über die Identität des Inhabers. Basis für SSL und S/MIME. Zertifikate werden von einer anerkannten Zertifizierungsstelle ausgestellt.

9.6.1. x.509 Zertifikatsformat

Heutzutage wird das X.509-Zertifikat oft für die Internetnutzung, wie e-Commerce verwendet, und nicht mehr primär für den X.500 Verzeichnisdienst .
Wenn Alice eine Ware von Bob über das Internet kaufen möchte, aber diese Transaktionen über einen sicheren Weg vollziehen will, dann empfiehlt es sich, dies mittels Verschlüsselung und Authentifizierung zu machen. Bei einem Public Key-Verfahren ist es für Alice kein Problem, den öffentlichen Schlüssel von Bob zu bekommen; aber wie weiß Alice, dass Bob wirklich Bob ist? Dies ist die Aufgabe von Zertifikaten. Zertifikate binden die Identität, den Public-Key, und eine Menge anderer Information aneinander. Der schematische Aufbau eines Zertifikates:

9. 7. Erstellen von Zertifikaten

1. Schlüsselpaar muss erstellt werden
2. Verschicken des öffentlichen Schlüssels mit weiteren Informationen wie Name, als Certificate Signing Request (CSR) an die Zertifizierungsstelle (CA)
3. Prüfen und Bestätigen der Richtigkeit der Angaben durch die CA

Möglichkeiten ein Zertifikat zu erhalten:

1. Signierung des Server Zertifikates durch das TrustCenter
2. Selber TrustCenter spielen (Zertifikat durch ein weiteres zertifizieren)

9.7.1. Certificate Siging Request

Erforderliche Schritte zur Key-Generierung und Installation eines eigenen Zertifikats:

1. Serverkey für Apache Webserver erstellen (TripleDES und PEM verschlüsselt)
2. Backup des Serverkeys
3. PEM-entschlüsselte Version des Serverkeys für automatisches Start des Servers wichtig → muss vor Zugriff von Unbefugten geschützt werden
4. CSR mit dem privaten Schlüssel des Servers → Ausgabe ist PEM-formatiert
5. Eingeben des korrekten FQDN (Fully Qualified Domain Name), wenn OpenSSL nach dem Common Name fragt
6. Antwort von CA
7. Es existieren zwei Dateien server.key und server.crt
8. Einbinden der Dateien in die Konfigurationsdatei httpd.conf des Apache (Will man die Eingabe des Passwortes bei jedem Serverstart erfahren sollte als CertificateKeyFile der entschlüsselte Serverkey eingegeben werden.

9.7.2. Eigenes TrustCenter

1. Generieren eines eigenen RSA-key für das TrustCenter (oder ein bestehendes CA-Zertifikat verwenden)
2. ca.key sollte gebackuped und das Passwort sicher aufbewart werden
3. Erstellen eines selbsunterschriebenen CA-Certifikates mit dem RSA-key
4. Benutzen des CA-Zertifikates um CSR des Webservers zu unterschreiben und echte SSL-Zertifikate zu generieren
5. ggf. Anpassungen in der Konfigurationsdatei von OpenSSL vornehmen

10. Common Gateway Interface (CGI) (momentan Version 1.1)

Schnittstelle des Webservers, die es erlaubt, Anfragen eines Webservers an Programme weiterzureichen und ausführen zu lassen, zb. Formulareingabenverarbeitung, Speicherung von Daten auf dem Server oder Auslesen solcher Daten per Java, C++ oder diverser Skriptsprachen wie Perl, php. -> Webseiten als Oberflächen für „Anwendungen“

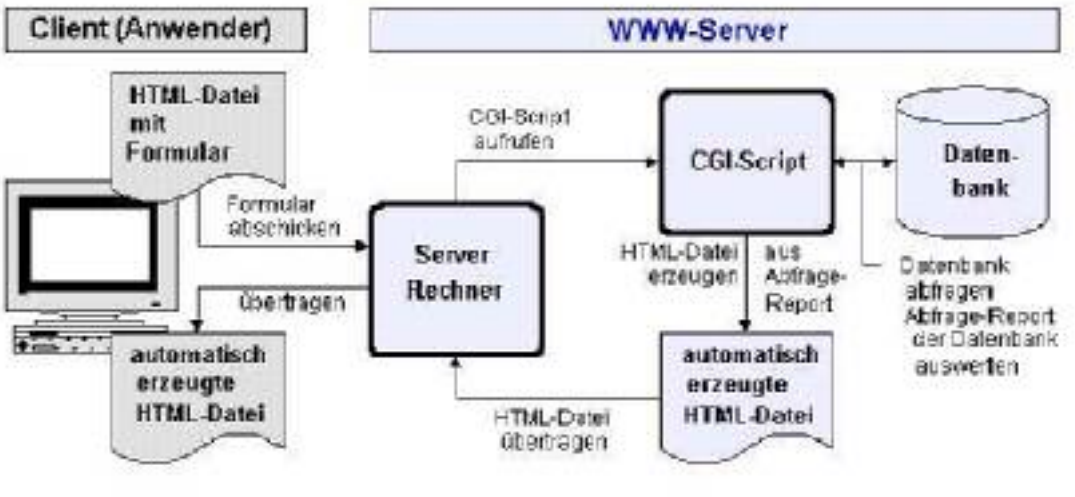

Abbildung 10.1: CGI

Speicherung von CGI-Umgebungsvariablen zur Verarbeitung von Daten.

Netscape: (NS)API oder Windows ISAPI optimieren die Performance für die Serversoftware des jeweiligen Herstellers.

Um CGI-Skripte nutzen zu können: Einbindung Modul mod_cgi

10.1 Verzeichnis als CGI-Verzeichnis deklarieren

Zur Nutzung einer CGI-Schnittstelle wird Zugriff auf ein Serververzeichnis mit CGI-Programmen benötigt (cgi-bin) Konfigurationsanweisung, damit Server weiß, dass sich hier ausschließlich CGI-Programme und Skripts befinden, die bei Zugriff ausgeführt werden, falls es die Dateirechte zulassen:

ScriptAlias /cgi/bin/ /usr/local/apache/cgi-bin/ (veraltet)

Nach apache.org

```
<Directory /usr/local/apache/cgi-bin>
        AllowOverride None
        Options ExecCGI         //Aktivierung der Ausführung von CGI-Skripten
        SetHandler cgi-script   //Setzen des Handlers
</Directory>
```

10.1.1 Eigenes CGI-Verzeichnis für jeden Benutzer

Neben DocumentRoot eigenes CGI-Verzeichnis für Benutzer bzw. cgi-fähig machen eines Unterverzeichnisses des Homeverzeichnisses:

```
<Directory /home/*/cgi-bin/>
        Options ExecCGI         //Aktivierung der Ausführung von CGI-Skripten
        SetHandler cgi-script   //Setzen des Handlers
</Directory>
```

10.2 Datei als CGI-Skript deklarieren

```
<Directory /usr/local/apache/htdocs/test>
        <Files script.sh>
        SetHandler cgi-script     //Setzen des Handlers
        </Files>
</Directory>
```

Hier muss allerdings auch ExecCGI für das Verzeichnis aktiviert sein.

10.3 Dateiendung zur Kennzeichnung von CGI-Skripten

Bei genereller Nutzung von CGIs in den normalen Verzeichnissen Setzen des cgi-script-Handlers für bestimmte Dateiendungen (hier cgi und pl), natürlich auch hier Setzung von ExecCGI notwendig:

```
AddHandler cgi-script cgi
AddHandler cgi-script pl
```

11. Logdateien

Überblick über Funktion und Auslastung

11.1 Error Log

Zuweisung der Logdatei durch ErrorLog in der Konfigurationsdatei

11.2 Access Log

AccesLogs im Common Log Format (CLF)

host rfc931 authuser [date/time] „request“ status bytes

host: IP-Adresse des abrufenden Rechners, je nach Serverkonfig auch Auflösung des DNS-Namens

[date/time]: Datum und Uhrzeit der Anfrage -> Schluß aus Verteilung der Abrufzeiten auf Clientel (wie Heimwerker, Schüler, Werktätige) und Länge des Besuchs
mit begrenzter Genauigkeit Schlüsse auf Ladezeiten

request:

- **http-Version**
- **http-Methode**: GET, HEAD, PUT, POST, DELETE
- **URL** Ermittlung der Abfragehäufigkeit einer Seite, evt. Feststellen von Verbindungsproblemen oder großen Dateien

status: http Status Code

bytes: (ohne Header) Ermittlung des übermittelten Datenvolumens und der Spitzenzeiten der Serverauslastung
wenn Angabe von Größe der Datei abweicht: Verbindungsabbrüche, Downloadclients mit mehreren Verbindungen oder Fortsetzung nach Abbruch

Format der Eintragungen kann mit festgelegten Formatanweisungen angepasst werden (FormatLog //Anzeige welcher Werte in dem mit CustomLog spezifizierten Logfile) Mögliche weitere Angaben:

Referrer: Adresse der zuvor besuchten Seite: Reihenfolge der Benutzung, Verlinkung von externen Seiten, Feststellen von unerlaubter Nutzung eigener Inhalte in fremden Frames.
Suchdienste -> evt. Anpassung der Suchbegriffe oder Anpassung der robots.txt
Sprachen der Benutzer und evt. Die Gegend, aus der sie stammen

Browserkennung: verwendeter Browser mit Programmversion, Sprachversion, Betriebssystem (Teilweise Angabe der Version) oder angepasste Browserversionen
Auch Suchmaschinen und Downloadprogramme.