Apache - Aufbau sicherer Webserver - Part 1

== Beschreibung ==
Es gibt keinen rundum Schutz, doch es gibt gewisse Einstellungen und Konfigurationen, die getroffen werden sollten. Es gibt zwei grundsätzliche Betrachtungsweisen, wenn es um die Sicherheit eines Webservers geht. Man muss sich um die Sicherheit der Webanwendungen kümmern, in den häufigsten Fällen sind dies: PHP – Skripte oder auf JAVA , .NET basierende Applikationen. Aber die sicherste Webapplikation bringt nichts, wenn der Daten – und Webserver nicht ausreichend vor Hackangriffen geschützt ist.
== Apache oder Internet Information Services ==
Ein wichtiger Schritt ist die Wahl des Webservers, es gibt zwar nicht die schlechteste oder die beste Webserversoftware, man wählt diese je nach Einsatzgebiet. Wenn man Webapplikationen basiert auf ASP.NET benutzen möchte, muss man IIS (Internet Information Services) von Microsoft benutzen.
In unsere Fall liegt ein Linux Server vor und die Webanwendungen basieren hauptsächlich auf PHP und MySQL, daher benutzen wir Apache (Ver. 2.2).
Alles weitere dreht sich nun um Apache, allerdings lassen sich die meisten Hinweise auch auf andere Webserversoftware übertragen z.B. auch auf IIS.
[Blockierte Grafik: http://img573.imageshack.us/img573/7885/apachehttpserverproject.jpg]
== Installation Deinstallation ==
Je nachdem, welches Betriebssystem benutzt wird, variiert die Installation ein bisschen. Eine Anleitung dazu findet man auf ? httpd.apache.org/docs/2.2/install.html
Hinweise zu PHP findet man hier ? de.php.net/manual/de/
Auch bei dieser Installation sind kleine Unterschiede zwischen den Betriebssystemen.
Die Wiederherstellung von MySQL in PHP ist hier ausführlich erklärt ? de2.php.net/manual/de/mysql.installation.php
Man sollte alles selbst konfigurieren und installieren, da man dann den gesamten Überblick über die Einstellungen behält. Man sollte keinen vorgefertigten Anwendungspakete wie XAMPP auf Webservern installieren, da sich dieses an die lokale Testumgebung anpasst. Die dort enthaltenen Konfigurationseinstellungen gewähren allen Benutzern einen sehr weitreichenden Zugriff, der von Angreifern ausgenutzt werden kann. Daher empfiehlt sich der Einsatz von XAMPP nur für lokale Testumgebungen, jedoch nicht für Webserver im Internet. Nachdem wir den Webserver sowie PHP und MySQL installiert haben, wird nun die Konfiguration unter dem Aspekt der Sicherheit im Detail betrachtet. Denn auch wenn die Webserversoftware sowie Module aus Haus aus relativ sicher sind, lässt sich mit der einen oder anderen Einstellung die Sicherheit des Gesamten Systems noch einmal immens erhöhen.
== Geziehlter Angriff ==
Um einen Webserver gezielt anzugreifen, benötigt man zu aller erst möglichst viele Informationen wie Namen und Version des installierten Webservers sowie der installierten Module. Der einfachste Weg um an solche Informationen zu gelangen, liegt im erzeugen eines Fehlers. Rufen wir um dies zu erreichen einfach eine Date auf, von der wir uns sicher sind, dass diese nicht existiert. In Betrieb ist ein Server der hinsichtlich der Sicherheitseinstellungen noch nicht über angepasste Konfigurationen verfügt. Um zu erfahren, welcher und vor allem welcher Version des Webservers eingesetzt wird, rufen wir einfach die URL localhost/foo.html auf, wobei diese Datei natürlich nicht existiert. Aus diesem Request resultiert nun ein „404 Error – Objeckt nicht gefunden!“ Fehler, welcher zurück an den an den Browser geschickt wird. Mit der Information, dass die Datei nicht existiert, werden gleichzeitig wichtige Informationen über das System mitgeliefert. In unserem Fall enthält die Webserver Signatur folgenden Inhalt.
== Signierung ==
Quellcode

1. localhost
2. 13/12/10 20:52:46
3. Apache/2.2.8 (Win32) DAV/s mod_ssl/2.2.8 OpenSSL/0.9.8g mod_autoindex_color PHP/5.2.5

== Informationen durch die Signierung ==
Neben dem Webserver Apache erhalten wir außerdem Informationen über die eingesetzte Version, das Serversystem, Modul sowie die verwendete Version von PHP. Als Angreifer kann man mit Hilfe dieser Informationen nun gezielt nach Sicherheitslücken in der Webserversoftware oder in der Skriptsprache suchen.
== ServerToken ==
Um genau diese Herangehensweise direkt zu unterbinden, kann man in der Konfiguration des Apache Servers, die Option ServerToken abändern. Die Konfiguration des Webservers wird über die Datei „apache2.conf“ gesteuert. Diese Datei kann man mit einen beliebigen Editor bearbeiten und dabei unter folgenden Konfigurationseinstellungen wählen.

Quellcode

1. ServerTokens Prod[uctOnly]
2. Der Server sendet (z.B.): Server: Apache
3. ServerTokens Major
4. Der Server sendet (z.B.): Server: Apache/2
5. ServerTokens Minor
6. Der Server sendet (z.B.): Server: Apache/2.0
7. ServerTokens Min[imal]
8. Der Server sendet (z.B.): Server: Apache/2.0.41
9. ServerTokens OS
10. Der Server sendet (z.B.): Server: Apache/2.0.41 (Unix)
11. ServerTokens Full (oder nicht angegeben)
12. Der Server sendet (z.B.): Server: Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2

Alles anzeigen

Dadurch kann man die ausgegebene Signatur selbst anpassen und dem Angreifer so falsche Informationen liefern. Dies ist sogar noch eine bessere Möglichkeit, als die Information ganz zu verstecken, da dadurch der Angreifer völlig im Dunkeln tappt. Jedoch müssen wir um dies zu erreichen, den Code des Apache Webservers neu kompilieren. Nähere Informationen hierüber findet man auf der offiziellen Seite des Apache Projekts. Allerdings ist dies eine größere Änderung und es sind Programmierkenntnisse nötig.
== HTTPrint - Anwendung ==
Neben der Möglichkeit mit Hilfe eines erzeugte Fehlers Informationen über einen Webserver herauszufinden, gibt es ebenso Applikationen, die es ermöglichen z.B. kann die Anwendung „HTTPrint“ anhand des Musters den jeweiligen Webserver erkennen und Informationen darüber ausgeben. Die Software ist für Windows, Linux, Mac OS C sowie Free BSD gratis erhältlich, ach wenn es sich um ein Open Source Projekt handelt, können wir uns die Software unter net-square.com/httprint/ für die jeweilige Plattform herrunterladen. Wenn wir anschließend die Anwendung starten, geben wir einfach die Adresse der Website ein, von der wir den Webserver identifizieren möchten und daraufhin erhalten wir die jeweiligen Informationen.
[Blockierte Grafik: http://img194.imageshack.us/img194/1007/httprint.png]
Die Grafik stammt von net-square.com/httprint/#screenshots
== Mit Hilfe von mod_rewrite einen Angreifer verwirren ==
Ein möglicher Angreifer braucht möglichst viele Informationen über das anzugreifende System. Wenn man die Signatur des Webservers deaktiviert, ist es noch möglich diese zu erraten, welche Webserversoftware auf dem Server eingesetzt wird, so auch die Skriptsprache. Eine sehr weit verbreitete Skriptsprache ist PHP, die auch nicht vollkommen sicher ist. Das kann man schon daran erkennen, wenn man auf eine Website geht die z.B. so aussieht. example.com/example.php
Das bedeutet für uns als Administrator, dass wir so wenige wie möglich an Informationen über unseren Webserver freigeben wollen. Das wichtigste Modul dafür ist das „mod_rewrite“, welches man über die Apache Konfiguration aktivieren kann. Dazu öffnen wir die Datei „httpd.conf“ und suchen „LoadModul rewrite_modul modules/mod_rewrite.so“. Wenn am Zeilenanfang eine Raute zu sehen ist, muss diese entfernt werden und abgespeichert. Nach dem Neustart des Webservers können wir auf das Modul „mod_rewrite“ zurückgreifen. Mit Hilfe des Moduls, kann man nun die Endung .php oder auch .php4 vertuschen oder auch als .html ausgeben. Dadurch kommt eine Hacker nicht an die Information um welches Skript es sich handelt. Um dies zu testen, benennen wir eine URL mit Endung .php in .html um z.B. localhost/example.php in localhsot/example.html. Die geht mit folgendem Code:

Quellcode

1. RewriteEngine on
2. RewriteRule example.html$ example.php

auch URL's mit Attributen können umbenannt werden, z.B. localhost/test.php?id=404. Die funktioniert mit folgendem Code:

Quellcode

1. RewriteEngine on
2. RewriteRule ^inhalt_([0-9]+).html$ test.php?id=404

Danach können wir die URL localhost/inhalt_404.html. So verschleiert man ebenfalls das Vorkommen von PHP. Den Code können wir dann in die „.htaccess“ Datei einfügen und diese dann in das Hauptverzeichnis unserer Webanwendung hinein laden. Das ist in der Situation praktisch, wenn man keinen Zugriff auf die Config – Datei des Apache Servers hat. Hier gibt es Weiteres zu diesem Thema (Beispiele aus der Praxis und gute Randinformationen) ? modrewrite.de/. Mit den vorherigen Tipps, lässt es sich also einrichten, dass weniger Angriffsfläche für einen möglichen Hacker geboten wird.
== Quellen: ==
httpd.apache.org/docs/2.2/install.html
de.php.net/manual/de/
de2.php.net/manual/de/mysql.installation.php
net-square.com/httprint/
net-square.com/httprint/#screenshots
modrewrite.de/

Hier geht es weiter --> [wiki]Apache - Aufbau sicherer Webserver - Part 2[/wiki]