Apache: SSL-Webanwendungen hinter Reverse Proxy

Möchte man eine Webanwendung (Beispielsweise Wordpress, Durpal, Magento) hinter einen Apache als Reverse Proxy betreiben, so wird man spätestens bei der Verwendung von SSL vor ein Problem gestellt:

Apache dient als SSL Backend, das heißt, dass das die Kommunikation zum Backend-Server unverschlüsselt erfolgt. Betreibt man nun eine Anwendung, die SSL erzwingt, wird man üblicherweise in einem Endlos-Loop landen.

Die Ursache ist folgende: Während der Client über SSL (HTTPS://...) auf die Anwendung zugreift, sieht der Backender Server nur (HTTP://...) und wird versuchen einen Redirect auf (HTTPS://...) durchzuführen. Für den Client erfogt immer eine Weiterleitung von HTTPS://.. auf HTTPS://... während das Backend immer nur HTTP://.. sieht.

VHost Konfigurieren

Mit eine einzelnen Konfigurationszeile im SSL-Vhost auf dem Proxyserver kann man dieses Problem umgehen:

In die SSL VHost konfiguration fügt man folgenden Zeile ein:

```
RequestHeader set X-Forwarded-Proto "https"
```

Das sieht dann Beispielsweise so aus:

```
<VirtualHost *:443>

        #Setze X-Forwarded-Proto auf HTTPS für das Backend
        RequestHeader set X-Forwarded-Proto "https"

        ServerName example.com
        #Notwendig, damit die vollständige URL an das Backend weitergegeben
wird.
        ProxyPreserveHost on

        #Alles was an example.com ankommt, wird an das Backend
weitergeleitet
        ProxyPass / http://10.0.0.1/ retry=0
        ProxyPassReverse / http://10.0.0.1/ retry=0
        #Mehr SSL Konfig
        .....
</VirtualHost>
```

Damit enthält der HTTP-Header die Information, dass es sich um einen SSL-Request handelt.

1. Möglichkeit: PHP-Anwendung anpassen

Manche PHP-Webanwendungen überprüfen nur den eigentlichen Protocol-Header mit $_SERVER['HTTPS']. Mit einer kleinen If-Anweisung kann man das Problem einfach umgehen:

```
if ($_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https'){
       $_SERVER['HTTPS']='on';
}
```

Der Code prüft dann den vom Proxy veränderten/gesetzten HTTP-Header und setzt das HTTPS Global entsprechend. Dieses Snippet kann man zum Beispiel in die Datei `wp-config.php` einer Wordpress Installation einfügen, damit ein SSL-Backend funktioniert.

2. Möglichkeit: Backend Apache2 Konfiguration ergänzen

`Diese Lösung soll falls möglich bevorzugt werden!` Unter dem Servernamen, wird der Eintrag “SetEnvIf X-Forwarded-Proto “^https$” HTTPS=on” im entsprechenden VirtualHost oder der conf.d-Datei ergänzt.

Beispiel:

```
# vim /etc/httpd/conf.d/matomo.conf
```

```
ServerName piwik.blackgate.org
SetEnvIf X-Forwarded-Proto "^https$" HTTPS=on

ServerTokens Prod
ServerSignature Off

<Directory /var/www/html/matomo>
    Options +FollowSymlinks
    AllowOverride All

    Require all granted
</Directory>

Header set X-Content-Type-Options: "nosniff"
Header set X-Frame-Options: "sameorigin"
```

3. Möglichkeit: .htaccess konfigurieren

Wenn man nicht im PHP-Code seiner Anwendung herumpfuschen will, dann kann man auch über die .htaccess Datei auf dem Backend-Server HTTPS vortäuschen.

Dazu genügt es folgende Zeile hinzuzufügen:

```
SetEnvIf X-Forwarded-Proto https HTTPS=on
```

Im Zusammenspiel mit der VHost Direktive auf dem Proxy sollte es nun keine Problem mehr geben

Last update: **2019/05/03 15:46**