

# Directory Traversal Attack



### Einleitung

Wenn man auf eine Internetseite geht werden Files interpretiert und angezeigt, die sich auf einem Server befinden. Diese Files, welche für die Website benötigt werden, befinden sich in einem dafür vorgesehenen Web-Verzeichnis.

Wie man aus Abbildung 1 entnehmen kann, macht der Client eine Anfrage auf den Server, dieser lädt die Datei, übergibt sie an den PHP-Interpreter und sendet ein File zurück an den Client, der es auf dem Browser angezeigt bekommt.

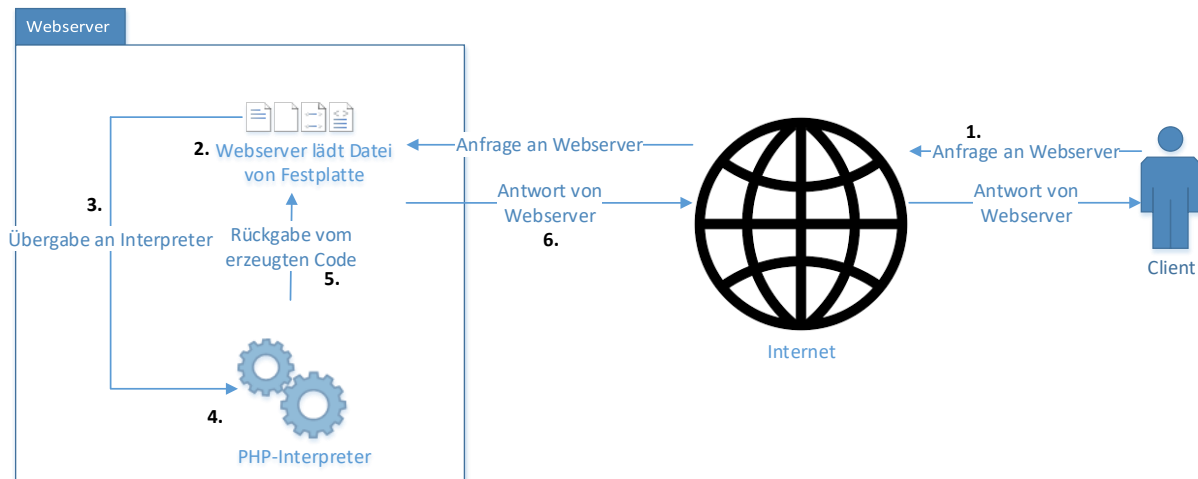


Abbildung 1 Webserver Anfrage mit PHP-Interpreter

Bei Abbildung 2 sieht man den gleichen Ablauf, wenn es sich nicht um ein PHP-File, sondern um ein zugängliches File auf dem Webserver handelt.

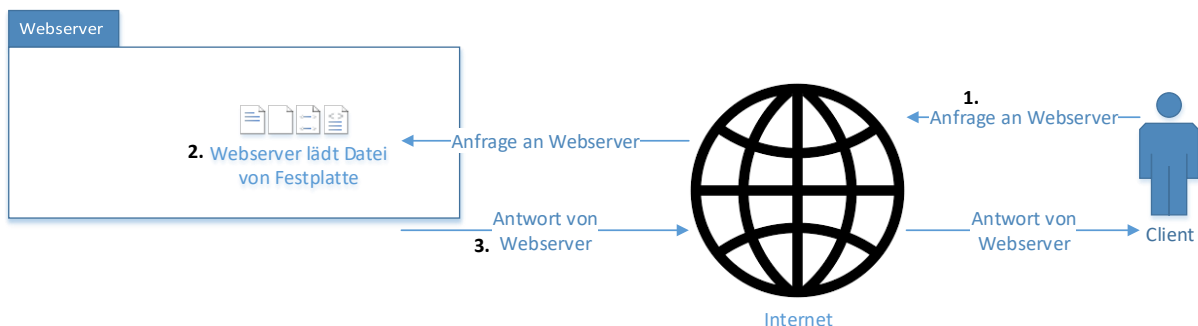


Abbildung 2 Webserver Anfrage ohne Interpreter

"Directory traversal" heisst auf Deutsch übersetzt Verzeichnis Durchquerung. Man durchquert und durchsucht die Verzeichnisse, um geschützte Files zu finden, oder Befehle auf dem Server auszuführen. Dies kann durch den PHP-Interpreter passieren, aber auch direkt durch den Web Server, welcher Dokumente direkt an den Client übertragen kann. Directory traversal Attacken werden durch Web-Requests auf Web Browsern, oder automatisiert mittels Skripte durchgeführt.

Um eine directory traversal Attacke durchführen zu können, muss man vertraut sein mit dem Aufbau von URL's, und Pfade in den Betriebssystemen. (Siehe Tabelle)

### Technische Ausführung

Directory traversal kann unterschiedlich angewandt werden. Hierbei kann man zwischen verschiedenen Angriffen unterscheiden:

- Angriffe, die auf dem Webserver öffentlich zugängliche Files suchen
- Angriffe, die über ein Parameter (POST, GET) auf ein anderes Directory zugreifen wollen

- Angriffe, die eine Variable findet, welche vom Programm falsch genutzt werden und mit denen man das Directory beeinflussen kann

### Angriffe auf dem Webserver

Wenn der Webserver auf den Server über sein Root (z.B. C:/xampp/htdocs/ ist das Root-Directory „/" vom Server) zugreifen kann, also auf das gesamte Verzeichnis, ist es möglich direkt auf das Filesystem zuzugreifen und Files auszulesen.




Abbildung 3 Direkter Zugriff auf Server

Diese Methode ist mittlerweile bei allen aktuellen Webservern automatisch gesperrt und somit nicht mehr anwendbar (ausser jemand verwendet zum Beispiel einen Apache Server V1.0).

Jedoch kann man auf dem Webserver durch diese Methode Files finden, indem man nicht über das Root-Verzeichnis vom Webserver hinausgeht, sondern im Webserver drin nach Files sucht. Es ist möglich, dass man Daten findet, die zugänglich sind, aber noch von niemandem gefunden worden sind.

### Angriffe über Parameter

Es ist möglich über Parameter Files auszulesen, oder Commands auszuführen. Wenn man erkennt, dass die Navigation direkt über den Parameter verläuft (siehe Abbildung 6) kann man auf Dokumente über dem Root-Directory vom Webserver zugreifen. Dies ist eine enorme Sicherheitslücke, da man alle Files vom gesamten Server auslesen kann, je nach Berechtigung vom Webserver.

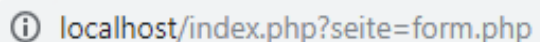


Abbildung 4 Mögliche URL

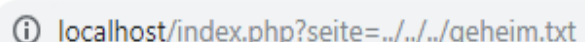
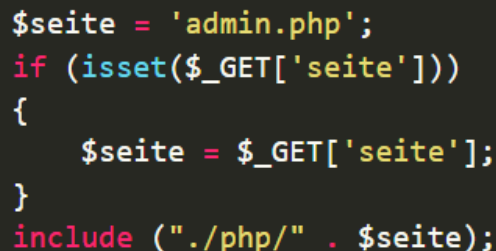


Abbildung 5 URL missbrauchen



```
$seite = 'admin.php';  
if (isset($_GET['seite']))  
{  
    $seite = $_GET['seite'];  
}  
include ("./php/" . $seite);
```

Abbildung 6 Direkt Files includen

In Abbildung 6 bekommt man die Seite als Parameter und fügt sie direkt in die Seite hinein, ohne zu kontrollieren was im Parameter drin ist. Es kann also irgendein Pfad in der Variable „\$seite“ enthalten sein und es wird direkt darauf zugegriffen (Pfad siehe bei Abbildung 5)

## Schutzmassnahmen

Als allgemeine Schutzmassnahmen kann man dem Webserver Rechte entziehen. Wenn der Webserver direkt nur auf seine Relevanten Verzeichnisse zugreifen kann, ist es nicht möglich Daten einzusehen, oder zu bearbeiten, welche nichts mit dem Webserver zu tun haben.

### Webserver konfigurieren

Man kann den Server entsprechend konfigurieren, sodass für Angreifer schon vom Webserver selbst Einschränkungen vorhanden sind.

#### .htaccess

Mit dem htaccess File kann man Einstellungen vornehmen, damit Angreifer weitergeleitet werden, oder auf gewisse Pfade nur begrenzte Zugriffe haben. Das htaccess-File existiert bei Apache Web

Servern. Bei anderen Web Servern können diese Eingrenzungen verschieden eingestellt werden. Siehe Tabelle im Anhang

### Schutzmassnahmen im Programm

Im Programm selbst sollte man niemals die erhaltenen Parameter, oder URL's direkt verwenden. Das heisst zuerst sollte man sie mit einer Liste vergleichen, oder direkt verändern.

```
$seite = $_GET['seite'];
if($seite == "anmeldung")
{
    include('./php/form.php');
}
else
{
    include('./php/admin.php');
}
```

Abbildung 7 Beispiel Schutzmassnahme

In Abbildung 7 sieht man, dass das „GET“ nicht verwendet wird, sondern im „Include“ direkt drin steht, welches File man nimmt.

## Anhang

### .htaccess Befehle

Befehl	Funktion
Options -Indexes	Verhindert das Browsen durch Ordner (Je nach Server ist das standardmässig eingeschaltet)
IndexIgnore *	Versteckt Inhalte von Ordner
IndexIgnore *.pdf	Zeigt beim Browsen entsprechende Dateiendungen nicht an
DirectoryIndex index.php	Standardseite ist nun index.php
RewriteEngine on RewriteCond %{REQUEST_FILENAME} !-f RewriteRule ^(.*) index.php/\$1 [L,QSA]	Alle URLs, welche nicht auf einen physikalischen Pfad (Datei oder Verzeichnis) zeigen, werden als PATH_INFO-Pfad an index.php übergeben (Siehe \$_SERVER['PATH_INFO']):
Deny from all Allow from 120.0.0.21	Niemand hat Zugriff, ausser IP: 120.0.0.21

### Nützliche Links

Beschreibung	URL
.htaccess – File Informationen Hier sind viele .htaccess-Einstellungen zu finden.	<a href="http://www.htaccess-guide.com/">http://www.htaccess-guide.com/</a>
Linux Cheat sheet	<a href="http://www.vulnerability-lab.com/resources/documents/587.txt">http://www.vulnerability-lab.com/resources/documents/587.txt</a>
Windows Cheat sheet	<a href="https://www.gracefulsecurity.com/path-traversal-cheat-sheet-windows/">https://www.gracefulsecurity.com/path-traversal-cheat-sheet-windows/</a>

### Grundsätzliche Zeichen

Beschreibung	URL
../	Pfad zurück
%2e%2e%2f	ASCII – Pfad zurück (hexadezimal)
%252e%252e%252f	Mit doppelt encodetem % = « %25 »