

Sicherheitsproblem Webserver

Die größte Schwachstelle in Firmennetzen oder E-Commercelösungen sind Webserver. Das beste Beispiel dafür, dass am Markt verfügbare Webserver Schwachstellen aufweisen, stellt der IIS – das Webserverprodukt der Firma Microsoft dar. Im Durchschnitt werden pro Monat etwa zwei neue Sicherheitsschwächen öffentlich bekannt (siehe www.securityfocus.com; Betrachtungszeitraum Mai 01-Oktober 02). Doch auch das Vertrauen in das Open Source Produkt Apache ist nach der *Chunked Encoding Vulnerability* (URL:) nicht mehr ungebrochen.

Das Knacken von Webservern hat sich schon zum Volkssport entwickelt: die meist jugendlichen Täter präsentieren Kopien der von ihnen modifizierten Websites in Archiven wie Alldas (<http://defaced.alldas.org>).

E-Commerce-Anbieter versuchen verzweifelt, bezüglich der Vielzahl der angebotenen Patches auf dem Laufenden zu bleiben - was sich nicht einfach gestaltet: hat man beispielsweise die deutsche Version des IIS im Einsatz, so sind Patches oft erst nach Wochen verfügbar. Selbst die renommierte Unternehmensberatung Gartner Group warnt vor dem Einsatz des IIS in ihrer Studie "*The code red worm shows that you can't always patch fast enough*".

Doch obwohl andere Services als HTTP/HTTPS - wenigstens bei professionellen Lösungen - durch Firewallmechanismen weggefiltert werden, ist die eingesetzte Serversoftware (Apache, IIS, etc.) nur eines von vielen Einfallstoren, über die Angreifer in Webserver eindringen.

Webapplikationen sind meist individuell programmiert. In Perl, ASP oder anderen geeigneten Programmiersprachen wird die E-Commerce-Applikation, mit der der Benutzer kommuniziert, entwickelt. Doch gerade in dieser Individualität liegt die Problematik. Ein kleines Team von Webentwicklern erstellt eine individuelle, teils komplexe Lösung, die beliebigen Angriffen aus dem Internet ausgeliefert ist. Eine Qualitätssicherung findet nicht statt. Die Praxis zeigt, dass es sich bei den Webentwicklern in aller Regel nicht um ausgebildete Informatiker handelt, die Disziplinen wie Softwareentwicklung oder ingenieurmäßiges Vorgehen erlernt haben. Meist werden die Web-Applikationen von Quereinsteigern erstellt, die unter- oder fehlqualifiziert sind. Teilweise werden die Webshops auch von so genannten Multimedia-Agenturen errichtet, deren Kernkompetenz im Content- oder Design-Bereich liegt. Es ist daher kaum verwunderlich, dass E-Commerce-Lösungen ernst gemeinten Angriffen selten standhalten.

Never Trust Input Data

Der naivste und - aus der Perspektive eines Angreifers erfolgversprechendste - Fehler besteht darin, dass den Eingabedaten des Anwenders blind vertraut wird. Hierfür ein Beispiel aus der Praxis, dessen Verständnis Grundkenntnisse von HTTP voraussetzt. Ein Kunde einer Online-Bank kann verschiedene Konten verwalten. Jedes Konto wird durch eine eindeutige Konto-Nummer identifiziert. Meldet sich der Kunde über Web bei seiner Bank an, so werden ihm die Namen und Nummern seiner Konten aufgelistet. Per Mausklick kann der Kunde eines seiner Konten auswählen, zu dem ihm dann Details angezeigt werden. Beim Klick auf eines der Konten, wird per HTTP einfach die Kontonummer des Kontos zur Bank geschickt. Arbeitet der Kunde mit einem gewöhnlichen Webbrowser, so ist es ihm nicht möglich, Zugriff auf ein fremdes Konto zu erlangen. Modifiziert ein böswilliger Kunde aber die HTTP-Kommunikation, so ist er in der Lage, eine

fremde Kontonummer zur Bank zu schicken - und erhält so Zugriff auf ein fremdes Konto. Voraussetzung ist eine schlampig erstellte Webapplikation, bei der die Auswahl des Anwenders nicht überprüft wird. Das leider nur selten befolgte Paradigma lautet: "Never trust input data". Hier ein weiteres Beispiel, wie Ladendiebe in Webshops einbrechen. Eins ist bei jedem Webshop gleich: der Kunde hat die Möglichkeit, eine Ware in einen virtuellen Einkaufskorb zu legen:



Bei der Analyse dreier Webshops wurde festgestellt, dass beim Klick auf den Button mit der Bedeutung "diesen Gegenstand in den Einkaufskorb legen" nicht nur die Produkt-ID, sondern auch der Preis vom Anwender zum Webserver geschickt wird. Beim führenden Outdoor-Versender Globetrotter (www.globetrotter.de) sieht das etwa so aus:

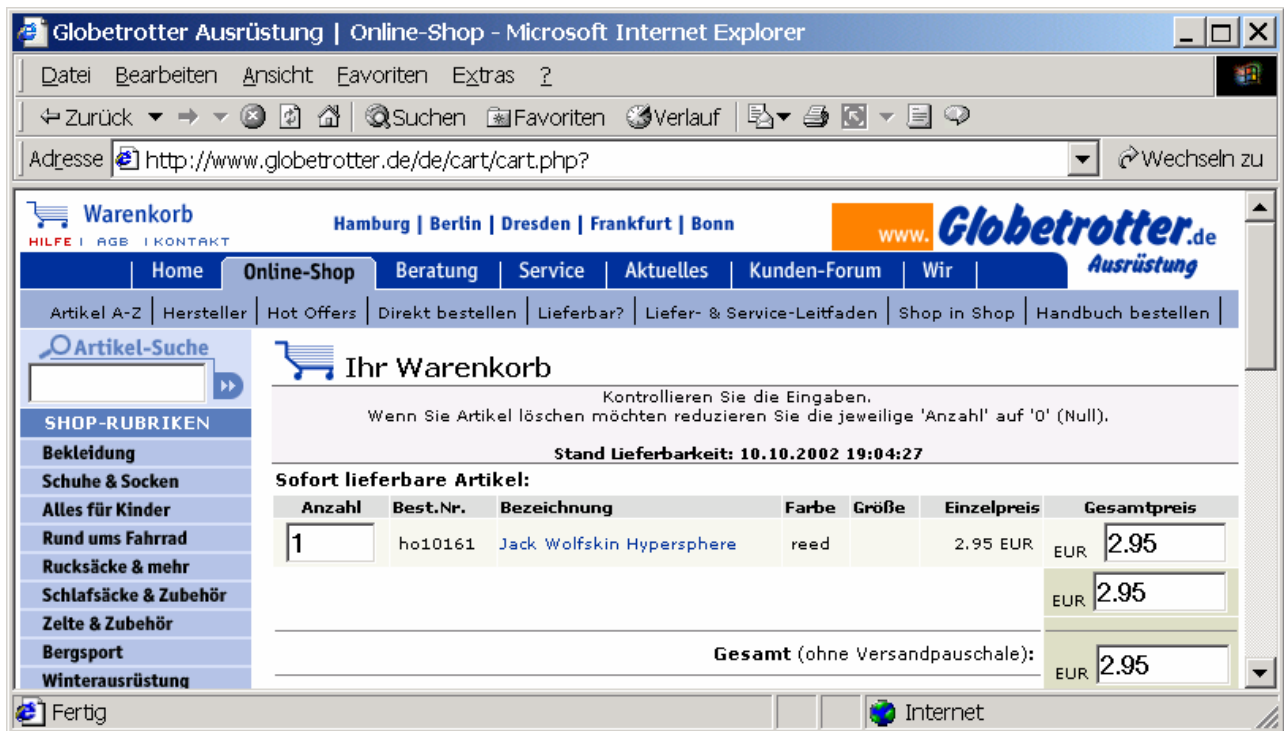
```
POST /de/cart/cart.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
Referer:
http://www.globetrotter.de/de/shop/detail.php?&mod_nr=ho10161&artbez=Jack+Wolfskin+Hypersphere&k_id=06&h_kat=Zelte%2C+Cam핑m%F6bel
Accept-Language: de,en-us;q=0.5
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.globetrotter.de
Content-Length: 256
Connection: Keep-Alive
Cache-Control: no-cache
```

```
amount=1&mod_nr=ho10161&art_nr=ho10161&artbez=Jack+Wolfskin+Hypersphere&vpreis=299.95&farbe=reed&groesse=&l_status=0&mwst=2&rabatt=&detail=1&follow_page=%2Fde%2Fcart%2Fcart.php%3F&cart_type=globe&put=1&In+den+Warenkorb+legen.x=19&In+den+Warenkorb+legen.y=7GET /media/korrektur.gif HTTP/1.1
```

Zu sehen ist also, wie ein Produkt (genauer: ein Zelt zu 299,95 Euro) in den Warenkorb gelegt wird. In einer simulierten Attacke wurde die HTTP-Anfrage wie folgt modifiziert:

```
amount=1&mod_nr=ho10161&art_nr=ho10161&artbez=Jack+Wolfskin+Hypersphere&vpreis=299.95&farbe=reed&groesse=&l_status=0&mwst=2&rabatt=&detail=1&follow_page=%2Fde%2Fcart%2Fcart.php%3F&cart_type=globe&put=1&In+den+Warenkorb+legen.x=19&In+den+Warenkorb+legen.y=7GET /media/korrektur.gif HTTP/1.1
```

Wie unschwer zu erkennen ist, wurde die Anfrage nur in einem Punkt verändert: der Preis beträgt nun nicht mehr 299,95 Euro - sondern nur noch 2,95 Euro. Ein Blick in den Einkaufskorb beweist, dass das Täuschungsmanöver erfolgreich ist.



Bei Sicherheitsüberprüfungen - auch im Banken und Finanzsektor - zeigt sich, dass ein Großteil der Webapplikationen anfällig gegen solche Manipulationen ist. (Anm.: Der Webshop-Betreiber wurde vor Veröffentlichung dieses Artikels auf die Schwäche aufmerksam gemacht und hat sie mittlerweile behoben.)

Rechtlicher Rahmen

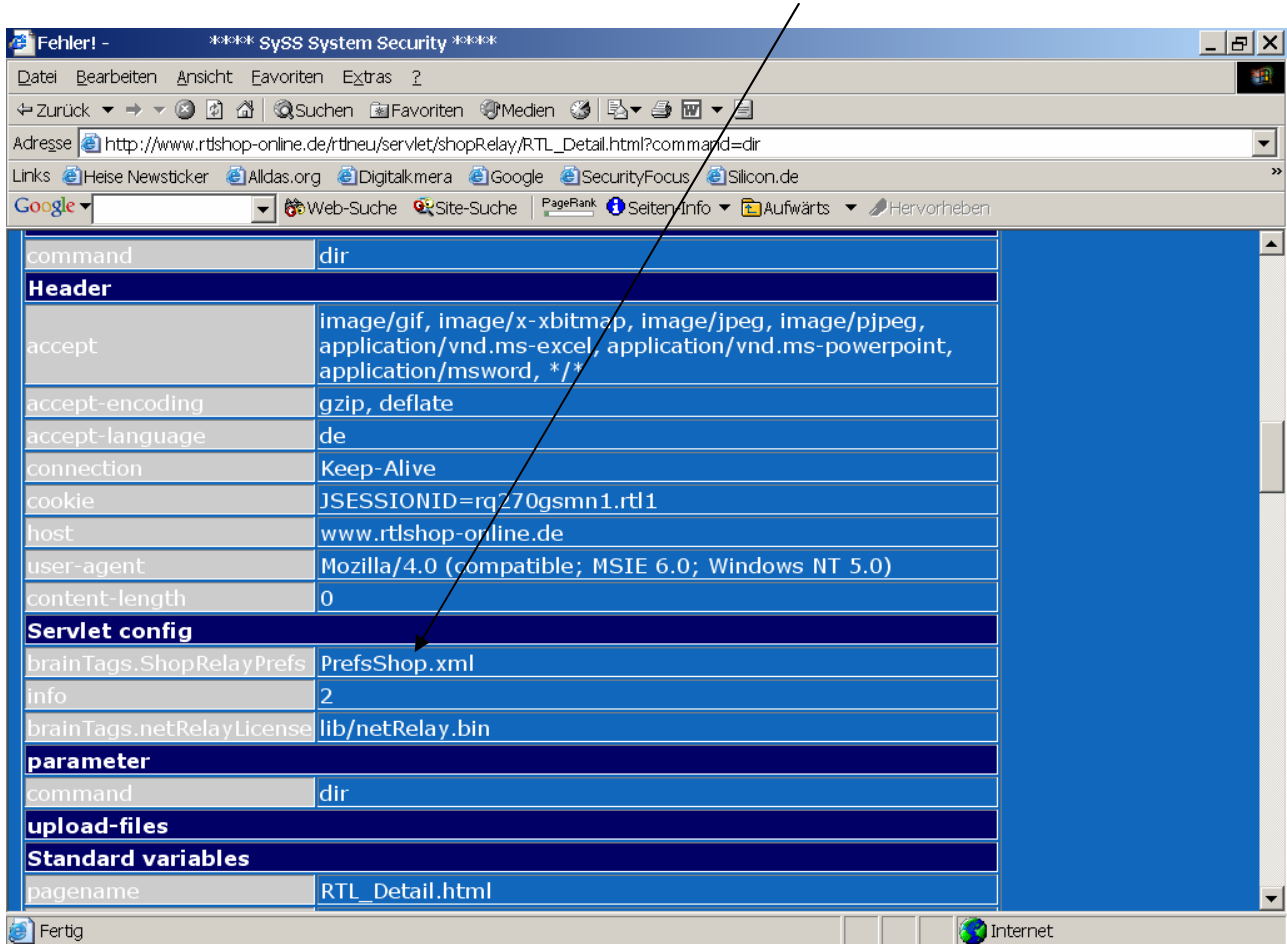
Interessant ist die Frage, inwiefern ein Trickdiebstahl auf die beschriebene Weise überhaupt strafbar ist. Der Trickbetrug wird in seine einzelnen Elemente zerlegt:

Perspektive des Benutzers	Juristische Interpretation
Kunde liest Produktbeschreibung und Preis	Angebot: Der Händler bietet ein Produkt zu seinen Konditionen an.
Produkt wird in den Einkaufskorb gelegt - der Preis wird allerdings von 299,95 Euro auf 2,95 Euro reduziert. Der Kunde klickt daraufhin auf „bestellen“.	Auftrag, wobei der Preis vom Preis im Angebot abweicht. Das ist - zumindest im B2B-Sektor gängige Praxis.
Benutzer erhält Auftragsbestätigung per Email - mit dem niedrigen Preis.	Nach Eingang der Auftragsbestätigung ist der Handel perfekt.

Es ist äußerst fraglich, inwiefern ein Richter dieser Argumentation folgen würde. Schließlich ist es alles andere als üblich, die HTTP-Anfragen selbst zu schreiben.

Sicherheitsprobleme mit dem „RTL-Shop“

Der RTL-Shop leidet unter einem ganz anderen Problem: verändert man ein paar Zeichen in der URL, so wird eine Webseite mit Debugging-Informationen angezeigt; in dieser Webseite erhält man einen Hinweis auf die XML-Datei PrefsShop.xml:



The screenshot shows a web browser window titled "Fehler! - SySS System Security". The address bar contains the URL: `http://www.rtlshop-online.de/rtheu/servlet/shopRelay/RTL_Detail.html?command=dir`. The browser displays a debugging page with the following content:

command	dir
Header	
accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
accept-encoding	gzip, deflate
accept-language	de
connection	Keep-Alive
cookie	JSESSIONID=rq270gsmn1.rtl1
host	www.rtlshop-online.de
user-agent	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
content-length	0
Servlet config	
brainTags.ShopRelayPrefs	PrefsShop.xml
info	2
brainTags.netRelayLicense	lib/netRelay.bin
parameter	
command	dir
upload-files	
Standard variables	
pagename	RTL_Detail.html

A red arrow points from the text above to the 'brainTags.ShopRelayPrefs' entry in the 'Servlet config' section of the table.

Ruft man nun die Konfigurationsdatei PrefsShop.xml per Webbrowser ab, so stellt man fest, dass in der Konfigurationsdatei die IP-Adresse, LoginID und Passwort des Datenbankservers stehen.

```
<?xml version="1.0" ?>
- <Preferences>
- <database>
  <standard_DB value="jdbc:interbase://192.168.1.11//datafiles/rtlshop/rtl.gdb" />
  <dbGateway value="brainTags.jOmniGate.interbase.Interbase_Gateway" />
  <db_User value="SYSDBA" />
  <db_UserPassword value="9ajasy" />
  <db_driver value="interbase.interclient.Driver" />
  <numberOfConnections value="8" />
  <buildDatabase value="true" />
  <addNewFields value="true" />
  <alterFields value="true" />
- <standardTables>
  <standardTable table="bt_MapperTableInfo"
    class="brainTags.netRelay.mapper.MapperTableInfo" uid="bt_MapperTableInfo"
    recordIdentifiers="id" idCompute="_STANDARD" />
  <standardTable table="bt_Sendeplan" class="brainTags.shopRelay.rtl.Sendeplan"
    uid="bt_Sendeplan" recordIdentifiers="id" idCompute="_STANDARD" />
  <standardTable table="bt_BertelOrder" class="brainTags.shopRelay.rtl.BertelOrder"
    uid="bt_BertelOrder" recordIdentifiers="id" idCompute="_STANDARD" />
  <standardTable table="bt_BertelOrderItem"
    class="brainTags.shopRelay.rtl.BertelOrderItem" uid="bt_BertelOrderItem"
    recordIdentifiers="id" idCompute="_STANDARD" />
</standardTables>
</database>
</Preferences>
```

Ohne weiteres ist ein Eindringen in den Datenbankserver der E-Commercelösung allerdings nicht möglich, da die vorliegende IP-Adresse (siehe RFC-1918) über das Internet nicht zu erreichen ist.

RTL wurde über die Sicherheitsschwäche informiert und hat diese bereits geschlossen.

Angriffe auf HTTPS-gesicherte Webserver

Die oben gezeigte Analyse (und natürlich auch die Manipulation) der Kommunikation zwischen Anwender und Web-Applikation ist alles andere als trivial wenn die gesamte Kommunikation aufgrund einer Verschlüsselungsschicht (nämlich SSL) verborgen bleibt. Der Schleier der Verschlüsselung lässt sich beispielsweise durch Dug Songs Webmitm lüften. Webmitm eignet sich hervorragend, um fremde HTTPS-Kommunikation unter Anwendung einer Man-in-the-middle Attacke auszuspähen. Für die gezielte Manipulation der Kommunikation ist Webmitm allerdings ungeeignet. Hervorragend ist ein geschicktes Geflecht verschiedener Stunnels. Stunnel (www.stunnel.org) ist ein Tool, dass traditionelle Services "ssl-ifizieren" kann, d.h. man kann HTTP in HTTPS umwandeln und umgekehrt.