Finally getting rid of SQL injections by integrating sqlmap in your DevOps pipeline

DevSecOps Meetup – LAUSANNE

11.09.2017 – Jérémy MATOS

whois securingapps

- Developer background
- Spent last 10 years between Geneva and Lausanne working on security solutions and products
  - Focus on mobile since 2010
- Now freelance consultant in application security

  http://www.securingapps.com
- Services to build security in the development lifecycle
  - Mobile
  - Web
  - Cloud
  - Internet Of Things
  - Bitcoin/Blockchain

@SecuringApps

Introduction

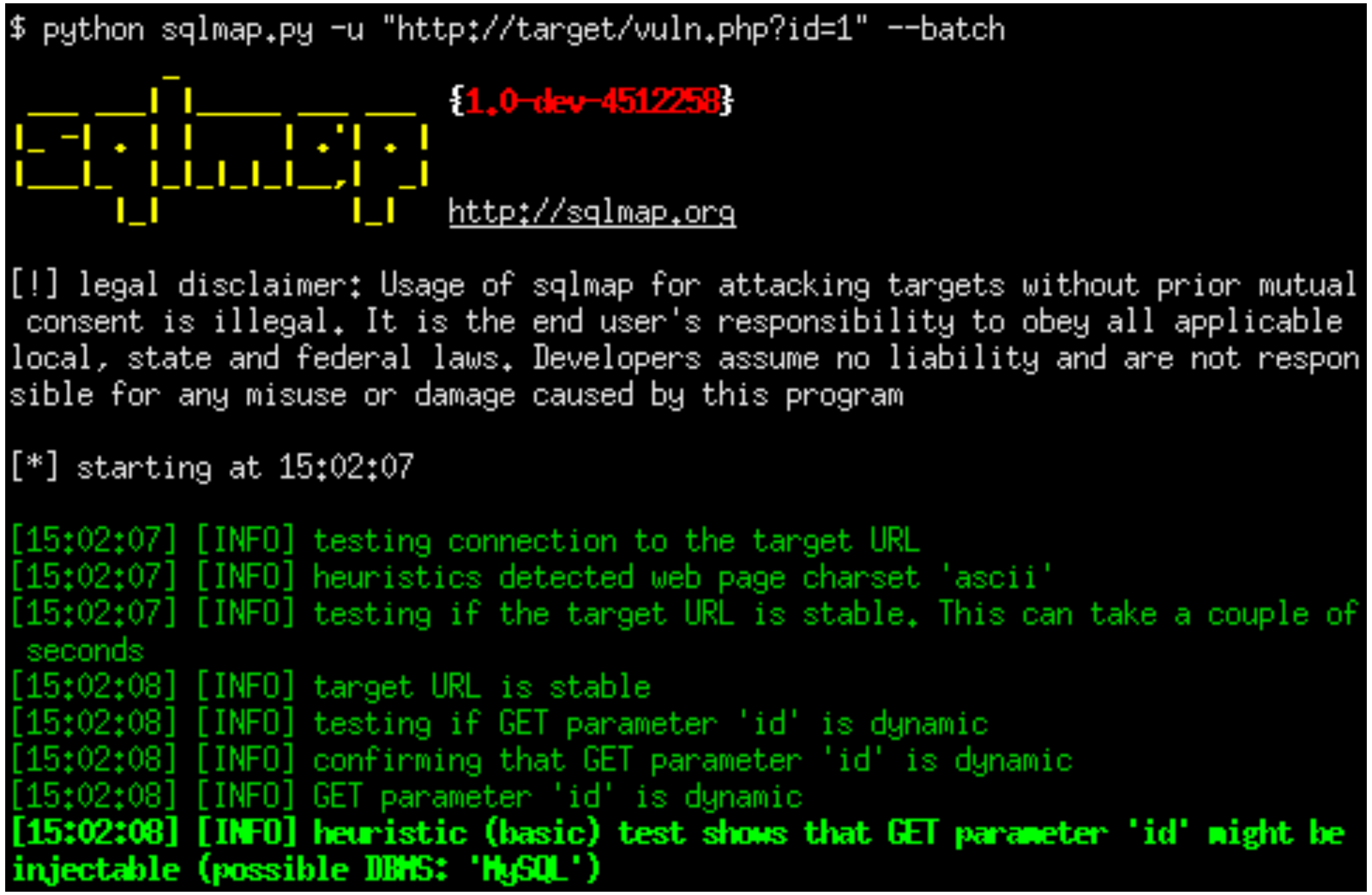
- Akamai's State of Internet Security report Q4 2016
  - 51% of observed web application attacks were SQL injections
- Solutions known for many years
  - Prepared queries
  - ORMs
  - Input validation / sanitization
- Yet massive dumps are regularly

Let’s finally get rid of those SQL injections

- Show how sqlmap can easily exploit SQL injections
  - On a relatively « modern and robust »
    - JAVA REST API
    - All inputs and outputs use JSON
  - i.e. not on old PHP software like WordPress..
- 3 examples inspired from real-world code reviews
  - Pathetic login service with copy-past from Stack Overflow
  - Prepared statement via Spring JDBC
  - ORM via Hibernate
- Source code: https://github.com/securingapps/badsql

sqlmap

- Great python CLI tool
- Automatically test very complex SQL attacks
- Detects database type and adapt injections
- Requires an url or input file with HTTP request
- Automagically finds parameters



```
$ python sqlmap.py -u "http://target/vuln.php?id=1" --batch
    _
 ___ ___| |_____ ___ ___  {1.0-dev-4512258}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

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

[*] starting at 15:02:07

[15:02:07] [INFO] testing connection to the target URL
[15:02:07] [INFO] heuristics detected web page charset 'ascii'
[15:02:07] [INFO] testing if the target URL is stable. This can take a couple of
 seconds
[15:02:08] [INFO] target URL is stable
[15:02:08] [INFO] testing if GET parameter 'id' is dynamic
[15:02:08] [INFO] confirming that GET parameter 'id' is dynamic
[15:02:08] [INFO] GET parameter 'id' is dynamic
[15:02:08] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable (possible DBMS: 'MySQL')
```

Pathetic login service

- Input
  - Userid
  - Password

- Output
  - Email if successful
  - NA otherwise

- Let’s see what sqlmap finds out about it
- And then have a look at the horrific source code

Get user sports service

- Input
  - Userid
  - Password (not checked for demo)

- Output
  - List of sports user likes
  - Empty list if nothing found for this user

- Uses prepared stament with Spring JDBC framework
- sqlmap will find nothing

Get user sports service extended

- Input
  - List of userid+password (password still not checked)
- Output
  - Merged list for all the users provided
- Also uses prepared stament with Spring JDBC framework
- But sqlmap finds issues ….
  - Let’s see why

Get user sports service 2

- Same API than before, but migrated to hibernate ORM

- sqlmap will find nothing on the single user service
- but still issues on the multiple user version

Using sqlmap as a safety net 1/2

- **No web application should go in production with sqlmap findings !**
- Running the tool with different recommended maturity levels
  - Level 1
    - Nightly scan on the master branch
    - Manual review of findings next morning
  - Level 2
    - On each pull request merged in the master branch
    - Make build fail if sqlmap can dump the list of tables
  - Level 3
    - On each pull request approved in whatever branch
    - Just after the manual code review
    - Make build fail if sqlmap finds any warning

Using sqlmap as a safety net 2/2

- Challenges
  - Have an up-to-date view of all the REST services available
    - Swagger ?
  - Generate automatically the HTTP request files for sqlmap

- Fixing the code
  - Add input validation in all your REST services
    - Special characters required for SQL injections is generally not valid business data
  - Avoid concatenating/replacing string by yourself
  - If really really needed, make sure input validation is strong

Conclusion

- SQL injections can happen even with quite modern frameworks
  - Complex queries often lead to issues because of hand-made built strings
  - ORM does not protect you in this case
  - Input validation will always save you
- sqlmap finds them very quickly
  - it can then dump all database really efficiently
- sqlmap should be embedded in your DevOps pipeline
  - it requires to have up-to-date sample requests for all your API calls

Any question ?

contact@securingapps.com

BLACK ALPS
CYBER SECURITY CONFERENCE
15-16 NOVEMBER 2017
Y-PARC, YVERDON-LES-BAINS, SWITZERLAND