# Web Application Honeypots with Focus on SQL Injection Emulation Capabilities

(30.4.2015)

# Agenda

- Introduction
- Related Work
- GlastopfInjectable
- Testing
- Future
- Conclusion

# Introduction
## Definitions

- **SQL Injection**

  The attacker inserts or “injects” a SQL query via the input data of the web application [1].

- **Honeypot**

  "A honeypot is a security resource whose value lies in being probed, attacked, or compromised" [2].

  → find motives and tactics of attackers or early warnings for new attacks

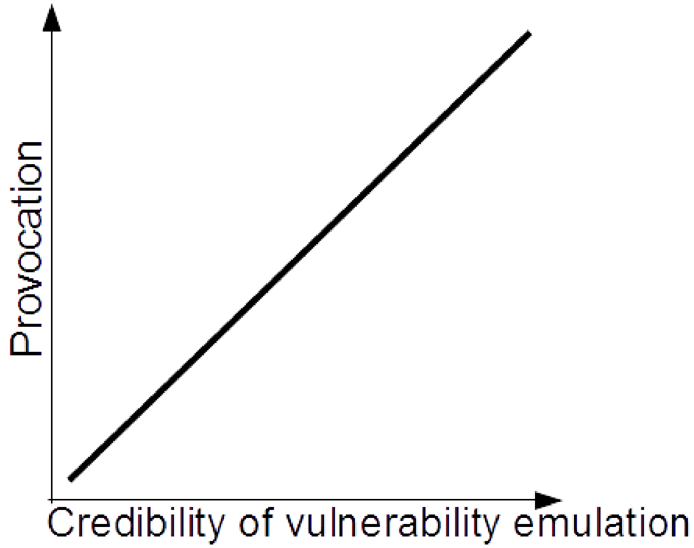
- **Web application honeypot**

  A honeypot that behaves like a web server and provides an HTML attack surface with known web application vulnerabilities [3].

# Introduction
## Motivation

### Goal

Development of a web application honeypot (GlastopfInjectable) that convinces the attacker that his SQL injections were successful. Maximize accuracy, convincibility and provocation.



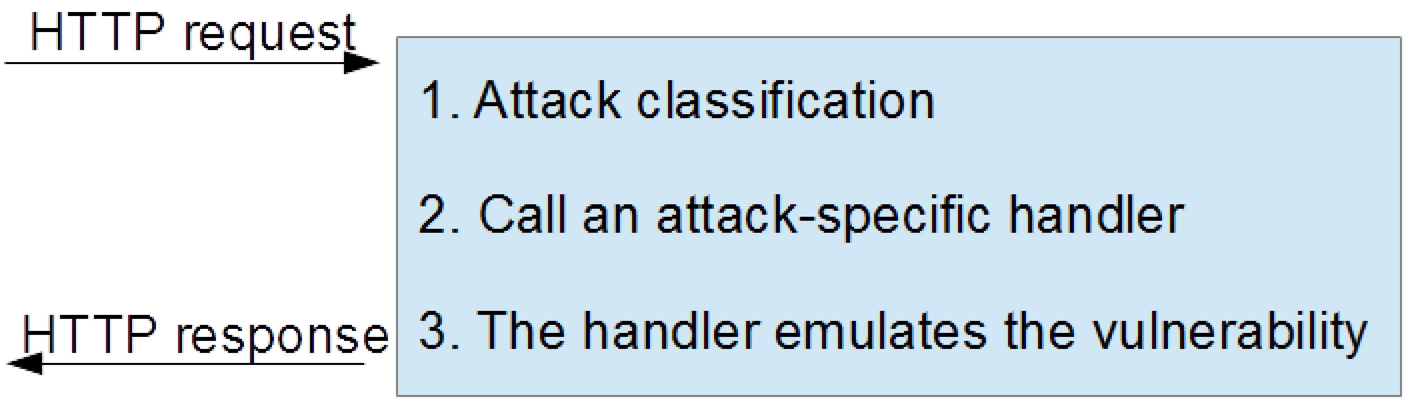
### Research Questions

- Is the SQL injection emulation’s behavior accurate enough to compete with a real vulnerability?
- Is an attacker convinced successfully?
- Is GlastopfInjectable capable of running in productive environments?

# Related Work
## The Web Application Honeypot Glastopf

- Glastopf is a low-interaction web application honeypot
- GlastopfInjectable improves and is based on Glastopf
- Written in Python
- It has a minimal HTTP request handler
- It emulates vulnerabilities

### Handling Procedure



HTTP request

1. Attack classification
2. Call an attack-specific handler
3. The handler emulates the vulnerability

HTTP response

[4]

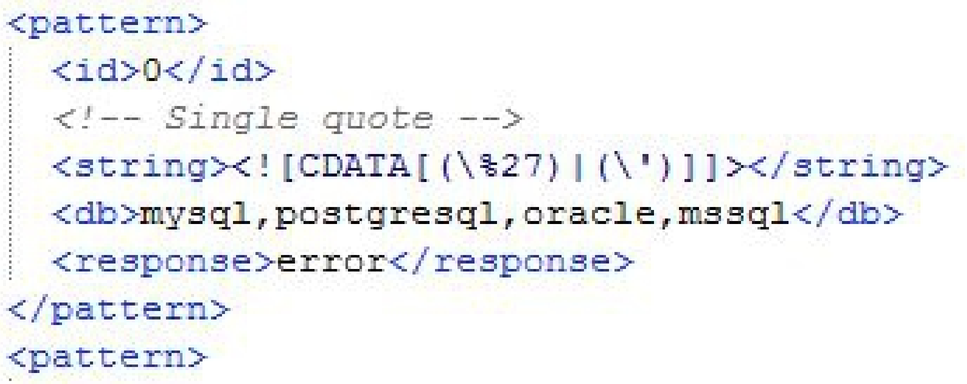
**Example**

1. SQL keyword
2. SQLiEmulator
3. Embed a MySQL error message in the HTTP response

# Related Work
## Glastopf's SQLiEmulator

### Procedure

- parses the injected string
- matches the tokens to patterns
- chooses a predefined response



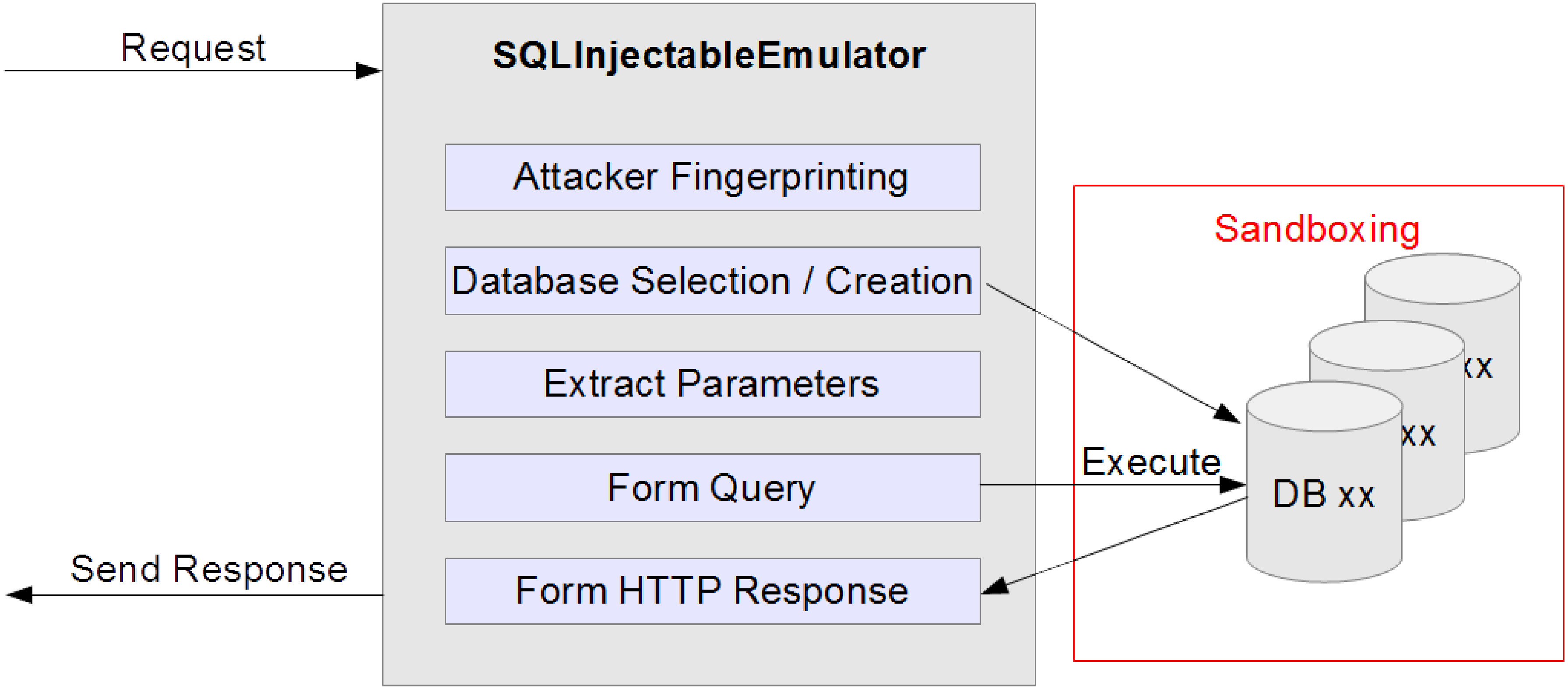
```
<pattern>
  <id>0</id>
  <!-- Single quote -->
  <string><![CDATA[(\%27)|(\')]]></string>
  <db>mysql,postgresql,oracle,mssql</db>
  <response>error</response>
</pattern>
<pattern>
```

### Disadvantages

- Response rigidity
- Just a few patterns and one predefined response are defined
- Pattern complexity (conditional statements, nested statements,...)
- No persistence (e.g. DROP table)

[3]

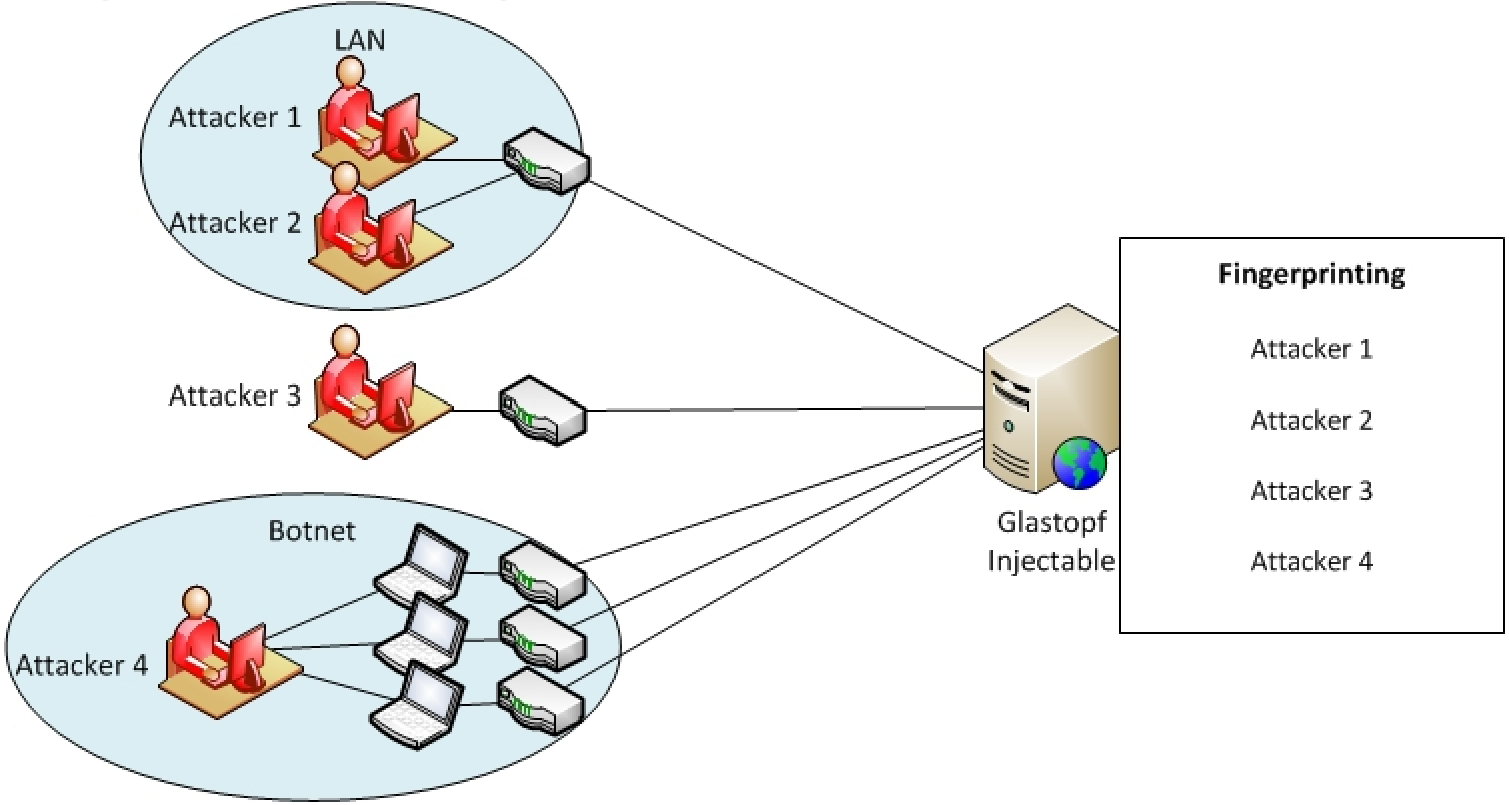
# GlastopfInjectable
## SQLInjectableEmulator Architecture



- High-interaction approach with SQL query execution in SQLite databases
- Attacker fingerprinting for isolation between attackers support of multi stage attacks → every attacker works on a distinct database copy, reused for revisitation

# GlastopfInjectable

## Fingerprinting



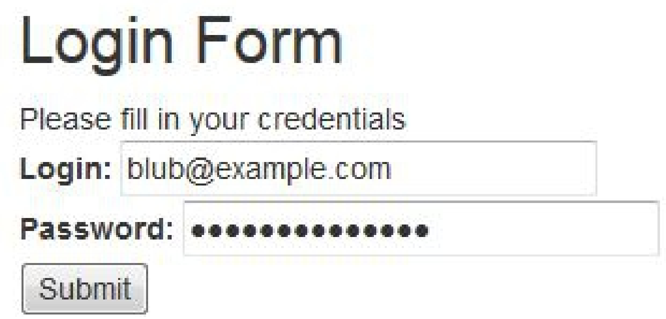
- The IP address is insufficient to recognize attackers
- GlastopfInjectable uses passive fingerprinting: IP address + HTTP headers

→ A high probability that different attackers using the same IP address are distinguished

[5]

# GlastopfInjectable

## Example SQL Injection



Login Form

Please fill in your credentials

**Login:** blub@example.com

**Password:** ●●●●●●●●●●●●●●

**User input:**

blub@example.com

bla' OR '1' = '1

- **User selection query for authentication**:

  "SELECT * FROM users WHERE email = ‘" + login +
  "’ AND password =‘" + password + "'"

- **Comment insertion query**:

  "INSERT INTO comments (comment) VALUES (‘" + comment + "’)"

→

SELECT * FROM users WHERE email = ‘blub@example.com’ AND
password =‘bla' OR '1' = '1’

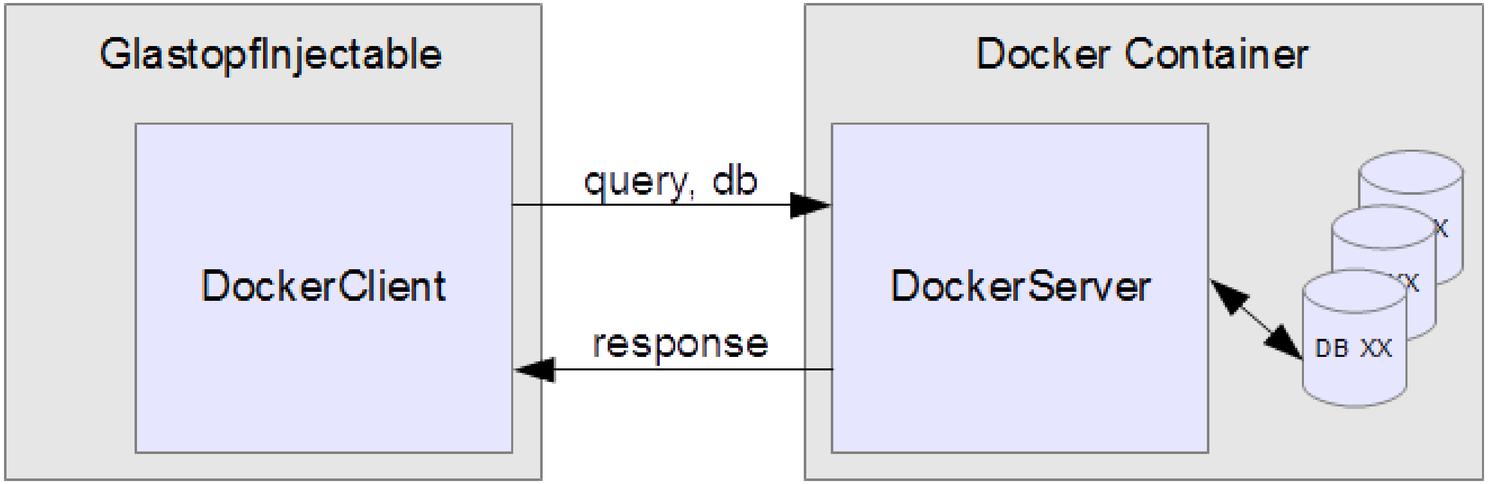
Tautology to bypass authentication!

# GlastopfInjectable
## Sandboxing

Worst case: SQL Injection can lead to machine compromise!

e.g. the stored procedure ***xp_cmdshell*** in Microsoft SQL Server 2000 executes any given command as an OS command shell [6]

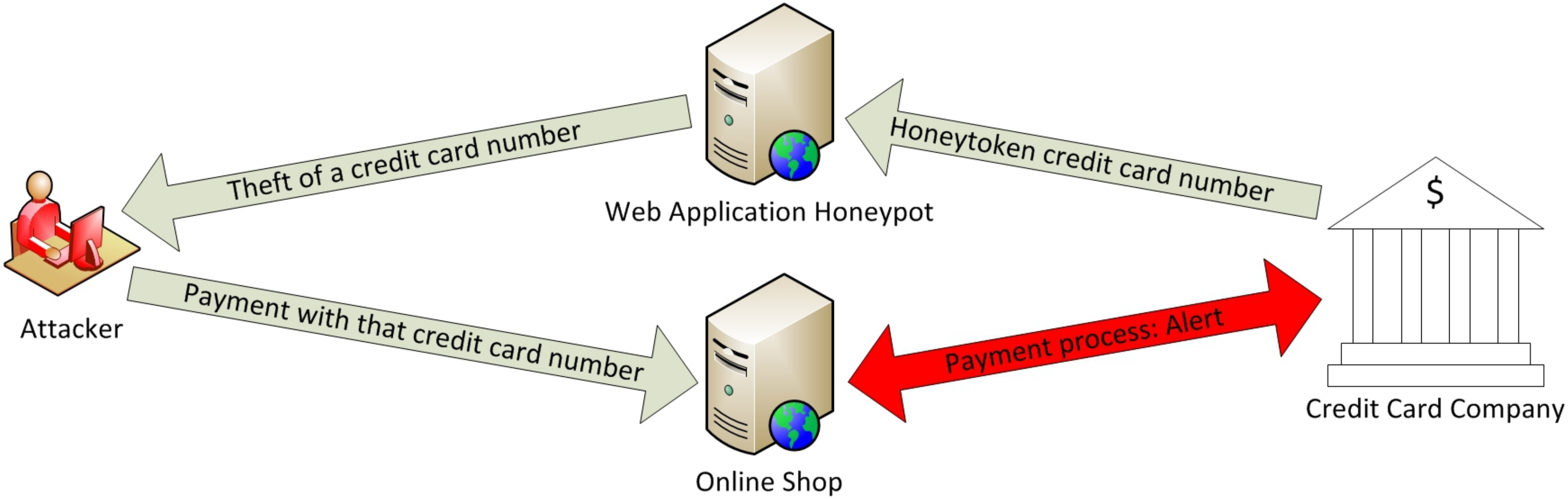
→ Query execution in a virtual environment (Docker container)



# GlastopfInjectable

## Honeytokens

- The attacker's attempt of unauthorized data access can be used by the honeypot to spread disinformation
- **Honeytokens = Disinformation that is used for tracking later** [7]
- Example: False credit card numbers are issued by credit card companies. During a fraudulent transaction they trigger alert. [8]


Theft of a credit card number
Web Application Honeypot
Honeytoken credit card number
$
Attacker
Payment with that credit card number
Payment process: Alert
Credit Card Company
Online Shop

# GlastopfInjectable
## Adaption to sqlmap's injection techniques

- **Sqlmap= SQL injection attack tool** [9]
- Sqlmap's tests for the following **injection techniques**:

  - Boolean-based blind,
  - Error-based,
  - Union query-based,
  - Stacked queries,
  - Time-based blind,
  - Inline queries [9].

- **Goal**:

  Improvement of GlastopfInjectable's response accuracy so that sqlmap finds as many techniques to be successful as possible.

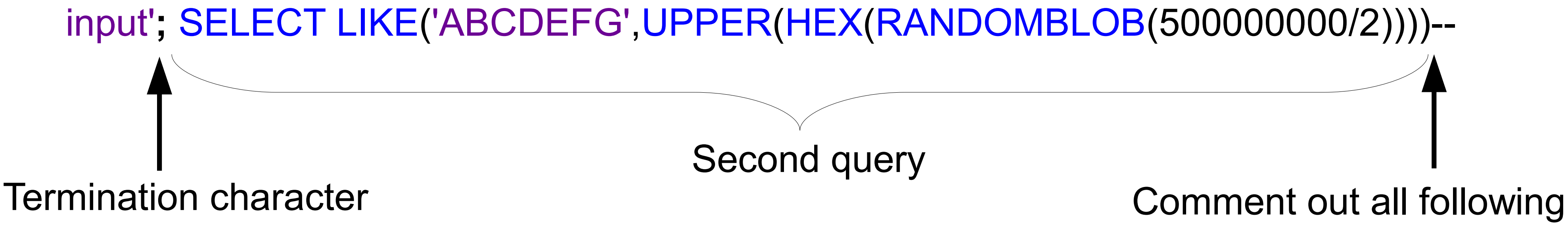
- **To examine**:
  - What strings does sqlmap inject?
  - How does sqlmap determine from the HTTP response that its SQL injection was successful?

# GlastopfInjectable

## Sqlmap Stacked Queries

- What are stacked queries?
  A string consists of multiple queries, that are given to the execute-function at once [10]

- Example stacked query that sqlmap uses for SQLite targets:



```
input'; SELECT LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))--
```

Termination character

Second query

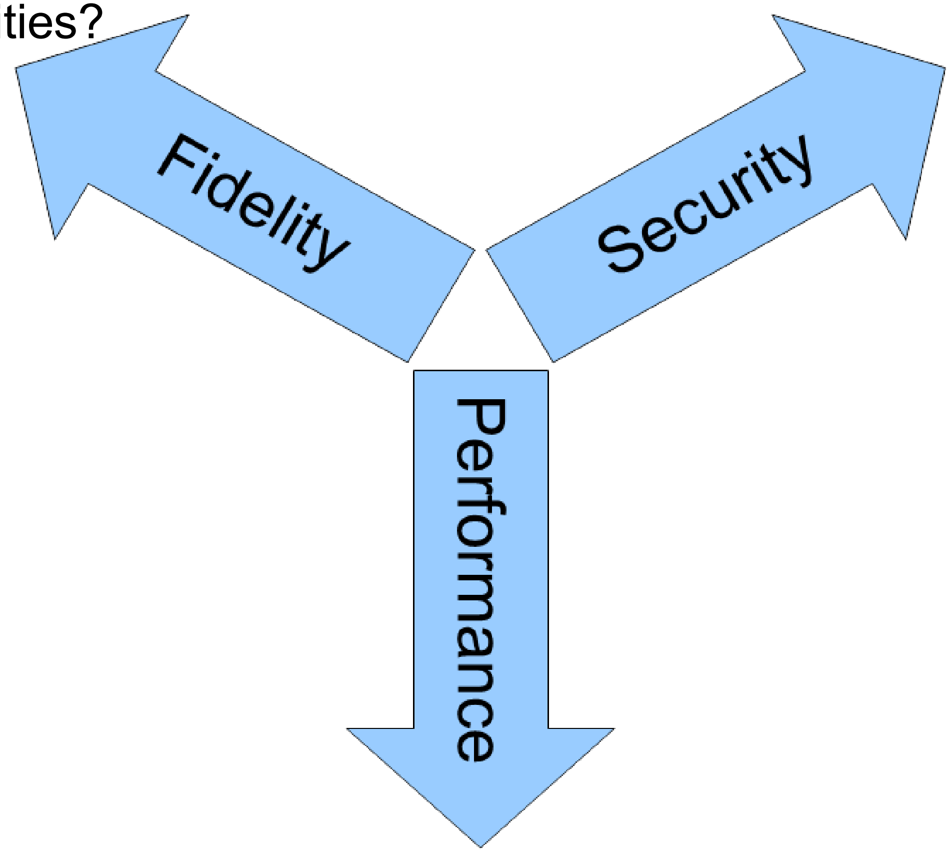
Comment out all following

- Sqlmap expects the HTTP response to be delayed

- Adaption in GlastopfInjectable:
  1. Make sure that the execution is able to handle stacked queries
     → split into several queries and execute all separately
  2. Time-based SQL injections need to work

# Testing
## Criteria

- Does GlastopfInjectable convince automated tools of its vulnerabilities?
- Do humans detect that they are attacking a honeypot?



Unintended vulnerabilities?

The round trip time is increased compared to the SQLiEmulator

[11]

# Testing – Fidelity towards Attack Tools

## Sqlmap Results

```
python sqlmap.py --url "http://192.168.56.101:8181/test.php
?login=blub@example.com&password=blub" --dbms sqlite
 ...
sqlmap identified the following injection points with a total of
100 HTTP(s) requests:
 ---
Place: GET
Parameter: login
Type: boolean -based blind
...
Type: UNION query
...
Type: stacked queries
Title: SQLite > 2.0 stacked queries (heavy query)
Payload: login=blub@example.com’; SELECT LIKE(‘ABCDEFG ’,UPPER(HEX(
RANDOMBLOB(500000000/2))))--&password=blub
...
Type: AND/OR time-based blind
---
Place: GET
Parameter: password
...
```

# Testing
## Real Attacks

- **Scanning tools**, e.g. searching for common vulnerabilities at PHP websites.

- A massive amount of the **SQL keywords** SELECT and UNION:
  Among 3807 requests are 2499 requests that contain the SELECT keyword.

- **Botnet finding**
  An attack sequence of requests with a frequent change of IP addresses all with the same upper and lower case spelling peculiarities.

```
/ConnectComputer/phpwcms/include/inc_ext/spaw/dialogs/
show_an.php?id=99999.9’+UnIoN+AlL+SeLeCt+
CaSt(0x393133353134353632312e39+as+char)+and+‘0’=‘0
```

- Attackers failed to **identify** the correct **parameters** for SQL injection.
  A popular paramter is “id”.

# Future

- Real attacks naive and do not identify parameters correctly

  → Acceptance of further parameters

- Current fingerprinting methods can be spoofed

  → A combination of further fingerprinting methods, e.g. botnet tracking or spelling-timing analysis

- SQLite was a bad choice for attracting attackers as most attack tools are specialized in MSSQL or MySQL

  → Integration of further DBMS and dynamic selection according to the SQL syntax of the attacker

# Conclusion

- GlastopfInjectable is able to run in productive environments.
- The high-interaction SQL execution and sqlmap adjustments lead to a high response accuracy to SQL injections.
- GlastopfInjectable convinces the attacker that his SQL injection was successful.

# Thank you for your attention!

# References

[1] https://www.owasp.org/index.php/SQL_Injection, visited on 26.04.2015

[2] Lance Spitzner. Honeypots: tracking hackers, volume 1. Addison-Wesley Reading, 2003

[3] Honeynet Project. Cyber Fast Track: Web Application Honeypot - Final Report. Technical report, Honeynet Project, 2012

[4] Lukas Rist, Sven Vetsch, Marcel Kossin, and Michael Mauer. Know your tools: Glastopf - a dynamic, low-interaction web application honeypot. The Honeynet Project, 2010

[5] Henning Tillmann. Browser Fingerprinting. 2013. http://www.henning-tillmann.de/2013/10/browser-fingerprinting-93-der-nutzer-hinterlassen-eindeutige-spuren, visited on 13.11.2014

[6] Microsoft. Microsoft SQL Server - xp_cmdshell. http://technet. microsoft.com/en-us/library/aa260689%28v=sql.80%29.aspx , 2014. visited on 19.12.2014

[7] Lance Spitzner. Honeytokens: The other honeypot, 2003. bandwidthco.com/sf_whitepapers/honeypots/Honeytokens%20-%20The%20Other %20Honeypot.pdf, visited on 12.12.2014.

[8] Thomas M Chen and John Buford. Design Considerations for a honeypot for SQL Injection Attacks. In Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference, pages 915–921. IEEE, 2009.

[9] Miroslav Stampar. sqlmap - Usage. https://github.com/sqlmapproject/sqlmap/wiki/Usage, 2014. visited on 2.1.2015.

[10] SQLINJECTION.NET. Stacked Queries. http://www.sqlinjection.net/stacked-queries/. visited on 3.3.2015.

[11] Niels Provos and Thorsten Holz. Virtual honeypots: from botnet tracking to intrusion detection. Pearson Education, 2007.