CyberSecurity: Principle and Practice

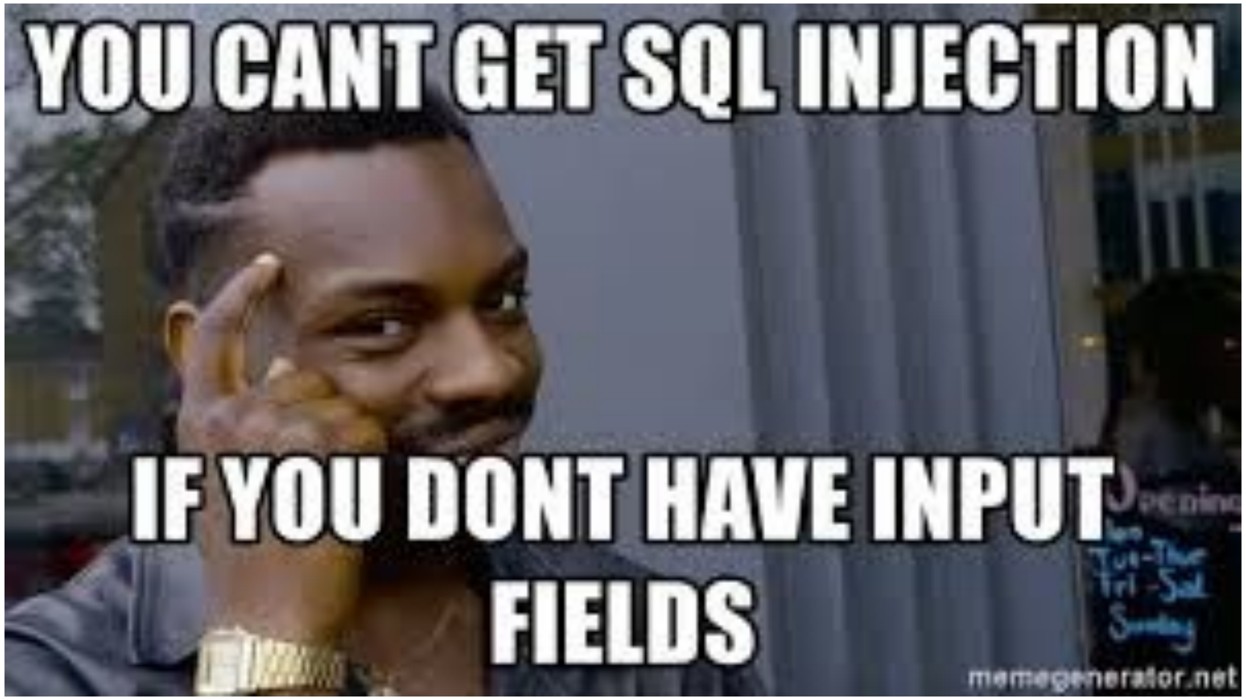
BSc Degree in Computer Science
2021-2022

Lesson 10: Injection Attacks

Teaching
Luca Pajola
pajola@math.unipd.it.
Pier Paolo Tricomi
pierpaolo.tricomi@phd.unipd.it

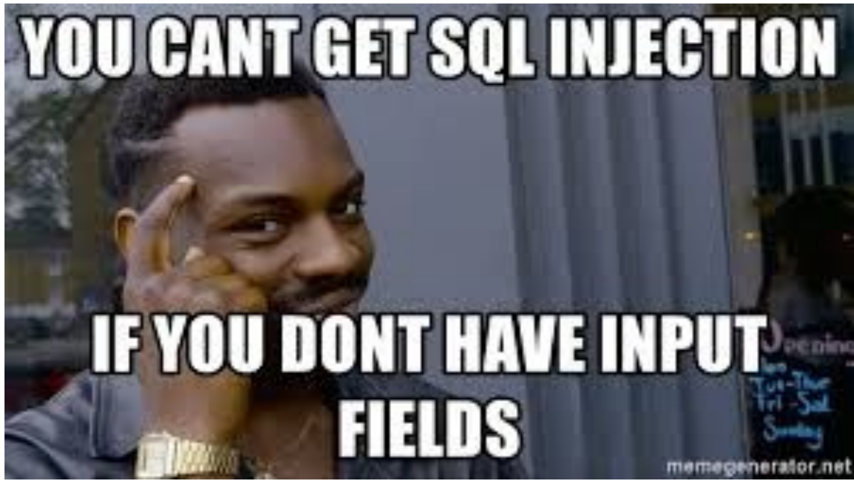
Università degli Studi di Padova
SPRITZ Security & Privacy Research Group
Dipartimento Matematica

Preliminaries



Introduction

- **Injection Attacks** are a class of attacks
- the attacker provides an untrusted input to our application
- the program processes the input and executes a function in an anomalous way
- it is considered the most dangerous class of attacks in web applications
- slides inspired by link



Case 1: Code Injection

- The attacker injects application code written in the application language
- potential impact: full system compromised
- the attacker might try to run OS command with program privileges
- e.g., in the following example we get the php version info

```
**
* Get the code from a GET input
* Example of Code Injection-
http://example.com/?code=phpinf
o();
*/
$code = $_GET['code'];

/**
* Unsafely evaluate the code
* Example - phpinfo();
*/

 eval("\$code;");
```

Case 2: CRLF

- Intro to CRLF:
    - a browser sends a request to a web server
    - the response of the web server contains
        - HTTP response header
        - a content (HTML page)
    - The two elements are separated with a combination of special characters
        - Carriage Return and a Line Feed (CRLF)
- the web server uses the CRLF to understand when new HTTP header begins and another one ends

Case 2: CLRF

- The attacker injects the CRLF characters into the input
- potential impact:
    - Cover other attacks
- for example, a web server might collect logs
    - 123.123.123.123 - 08:15 - /index.php?page=home
- if an attacker can execute the CRLF attack, he can fake the log content
    - /index.php?page=home&%0d%0a127.0.0.1 - 08:15 - /index.php?page=home&restrictedaction=edit
- this attack will produce two entries in the log file

Case 3: Cross-site Scripting (XSS)

- injection of custom scripts into a legitimate web server
  - usually JS codes
- this code is then executed in the victim’s browser
  - occur when the victim visits the web application
- the web page is the vehicle of the attack
  - e.g., forums, message boards, and web pages that allow comments

Case 3: Cross-site Scripting (XSS)

- e.g., a browser might have a function that shows the last comment published, available in the database

```
print "<html>"
print "<h1>Most recent comment</h1>"
print database.latestComment
print "</html>"
```

- the programmer assumption is that this should only contain text
- an attacker might inject the following:

```
<script>doSomethingEvil();</script>
```

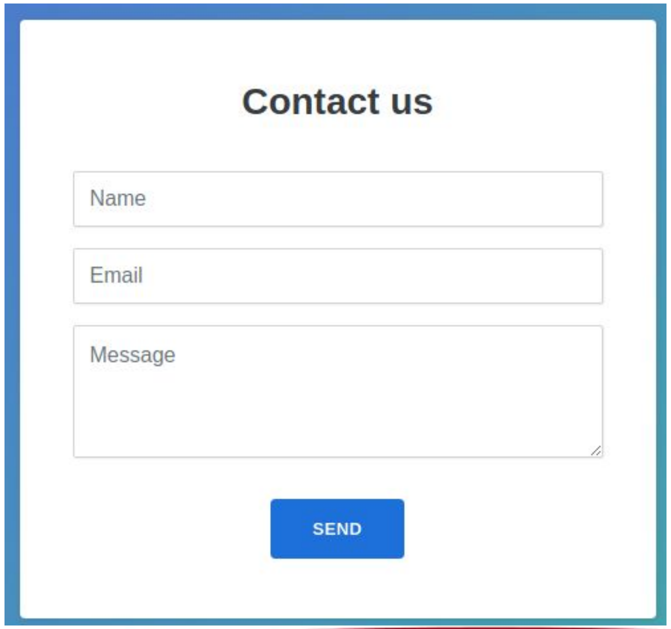
- The server will provide the following HTML:

```
<html>
<h1>Most recent comment</h1>
<script>doSomethingEvil();</script>
</html>
```

- when the victim loads the content, the script will be executed

Case 4: Email Header Injection

- Several web pages implement contact forms
- Most of the time, such contact forms set headers
- These headers are interpreted by the email library on the web server
  - turned into resulting SMTP commands
  - processed by the SMTP server
- A malicious user may be able to introduce additional headers into the message



Case 5: OS command Injection

- Injection of OS commands with users that run the application privileges
- For example, a PHP application might execute a ping to a given IP address

```
<?php
$address = $_GET["address"];
$output = shell_exec("ping -n 3 $address");
echo "<pre>$output</pre>";
?>
```

- The request is done via GET request
  - parameter name: address
- an attacker might request the following, displaying ping and list of files in the directory

```
http://example.com/ping.php?address=8.8.8.8%26ls
```

Case 6: SQL Injection

- injection of SQL instructions
- for example, given a database with credentials

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']

# SQL query vulnerable to SQLi
sql = “SELECT id FROM users WHERE username=’” + uname + “’ AND
password=’” + passwd + “’”

# Execute the SQL statement
database.execute(sql)
```

- The query returns always True if an attacker injects the following password:
  - password' OR 1=1

Exercises

1. Welcome to fun with flags.
2. Because creating real pwn challs was to mainstream, we decided to focus on the development of our equation solver using OCR.
3. "I have been told that the best crackers in the world can do this under 60 minutes, but unfortunately I need someone who can do this under 60 seconds."
4. I know my contract number is stored somewhere on this interface, but I can't find it and this is the only available page! Please have a look and get this info for me!

Questions? Feedback? Suggestions?