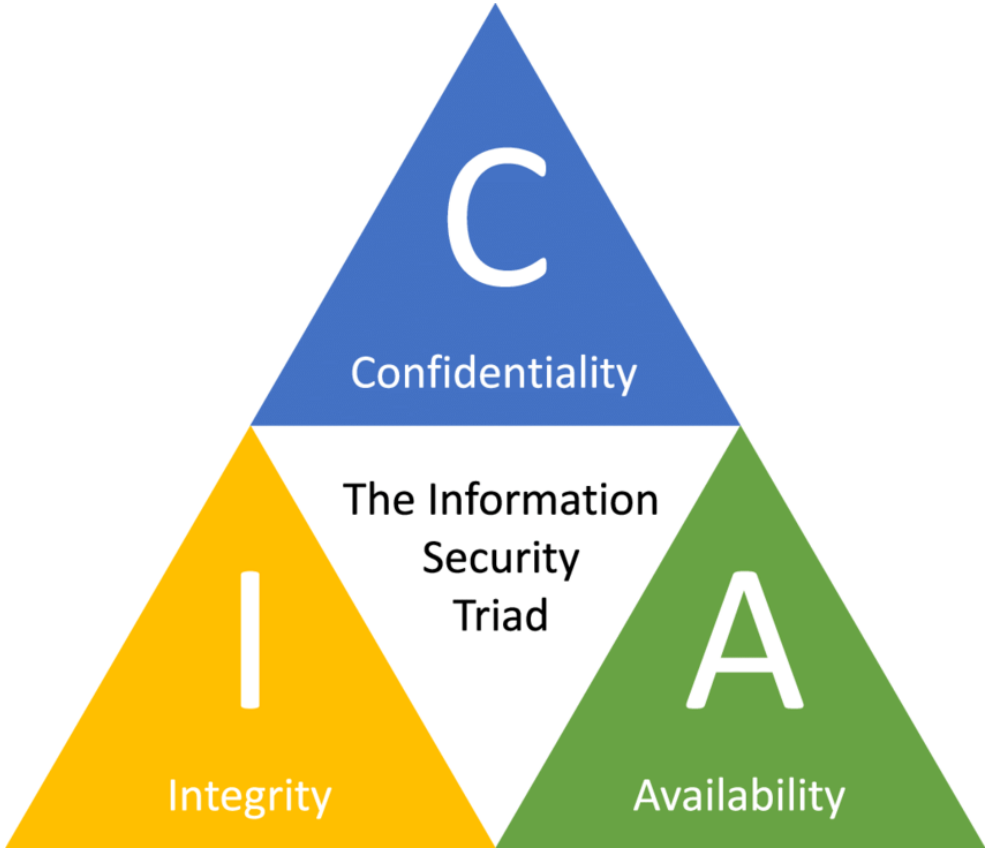
# Web Security

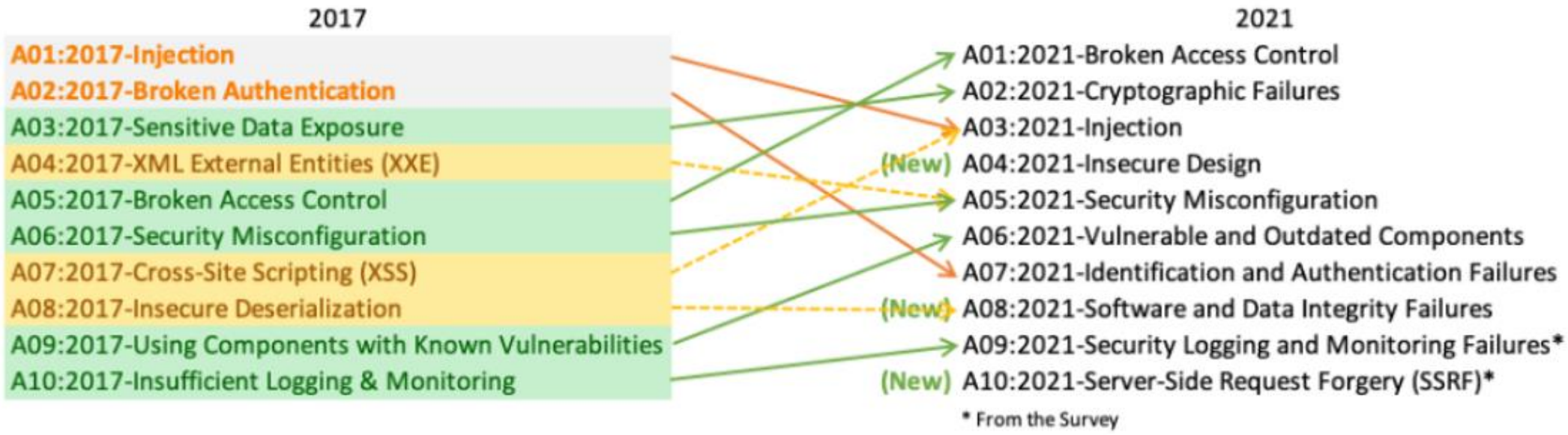
# What is Web App Security?

# Why is web security testing important?

# Security key



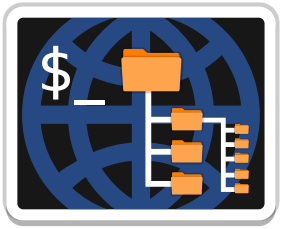
# OWASP Top Ten

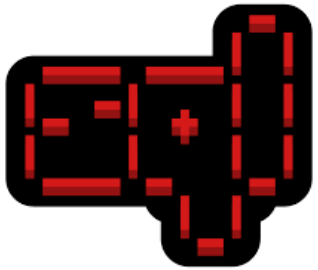


# A03: Injection – SQLin & XSS

# Tools



# Tools: NMAP


**Host discovery**

**Service detection**

**Operating system detection**

**TCP/UDP scan**

**Login credentials brute force**

# Tools: SkipFish

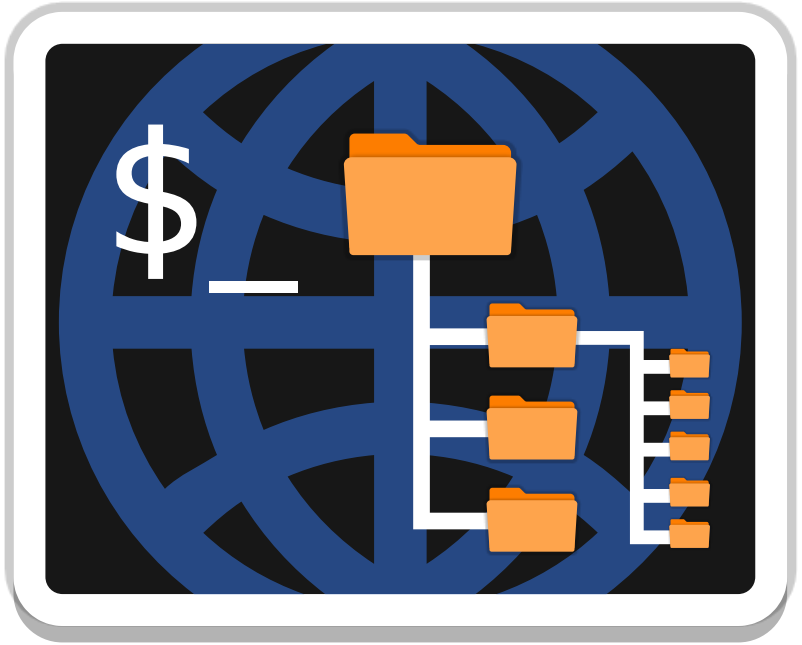

**Open source**

**Fully automated**

**Large number of modules**

**Enumeration**

# Tools: dirb

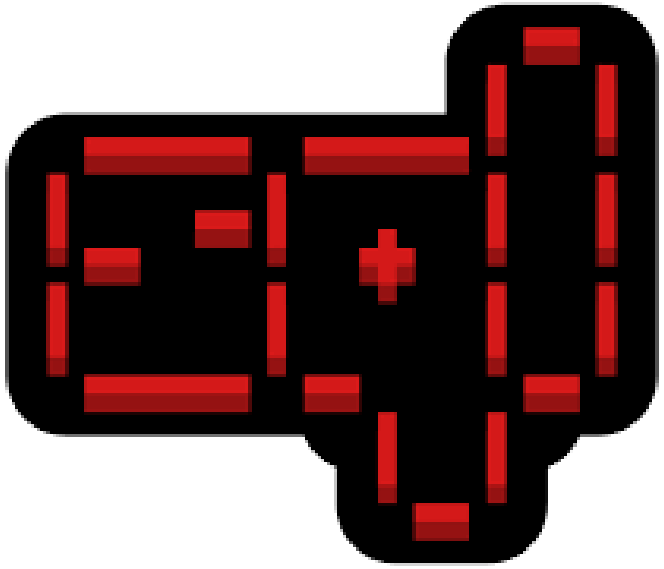



**Dictionary based**

**Preconfigured attack wordlists**

**Used by professionals**

# Tools: SQLmap



**Support popular database engines**

**Support multiple injection techniques**

**Software fingerprinting**

**Support password hash crack**

**Support to download and upload any file from db**

# Exploits & CVE

https://www.exploit-db.com/

https://www.vulnerability-lab.com

https://vuldb.com/

https://cve.mitre.org/

https://www.cvedetails.com/

https://nvd.nist.gov/

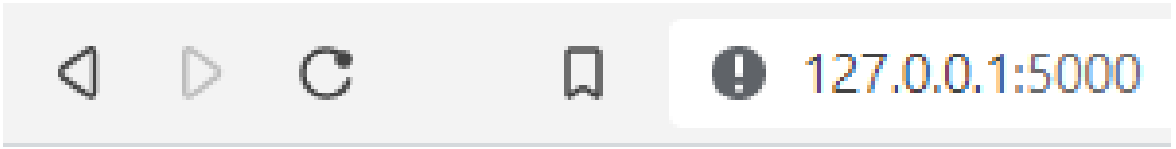
# Practical Example

The application is available at:
https://github.com/PiotrKontowicz/ExampleSQL

Endpoints:

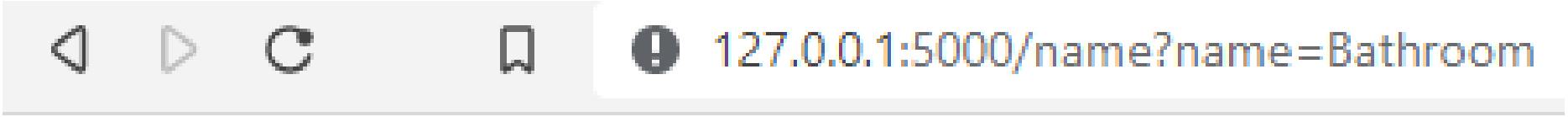
- http://127.0.0.1:5000/
- http://127.0.0.1:5000/name
- http://127.0.0.1:5000/save

# http://127.0.0.1:5000/



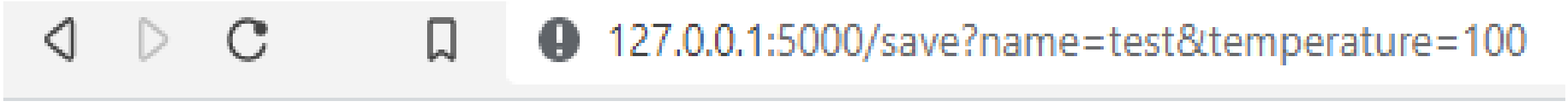

| Id | Name | Temperature |
|---|---|---|
| 1 | Bathroom | 36 |
| 2 | Kitchen | 6 |

# http://127.0.0.1:5000/name




| Id | Name | Temperature |
|---|---|---|
| 1 | Bathroom | 24 |

# http://127.0.0.1:5000/save




ok

# Example payloads:

- **1);DROP TABLE#**
- **1),('hacker',-100)#**
- **Bathroom' UNION SELECT table_name, null, null FROM information_schema.tables where table_schema ='test_app’#**
- **Bathroom' UNION SELECT id, name, NULL from admins#**

# How to prevent SQLin?

**Do not use data from any input directly to create SQL query, like this:**

```
'SELECT * FROM data where name=\'{}\''.format(name)
```

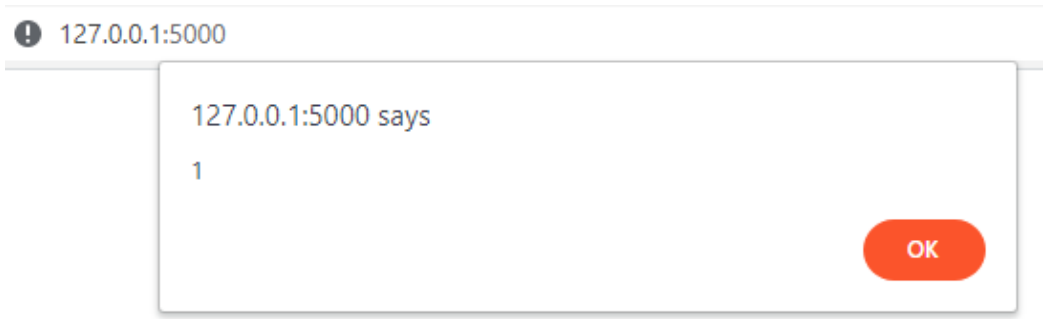
**Or this:**

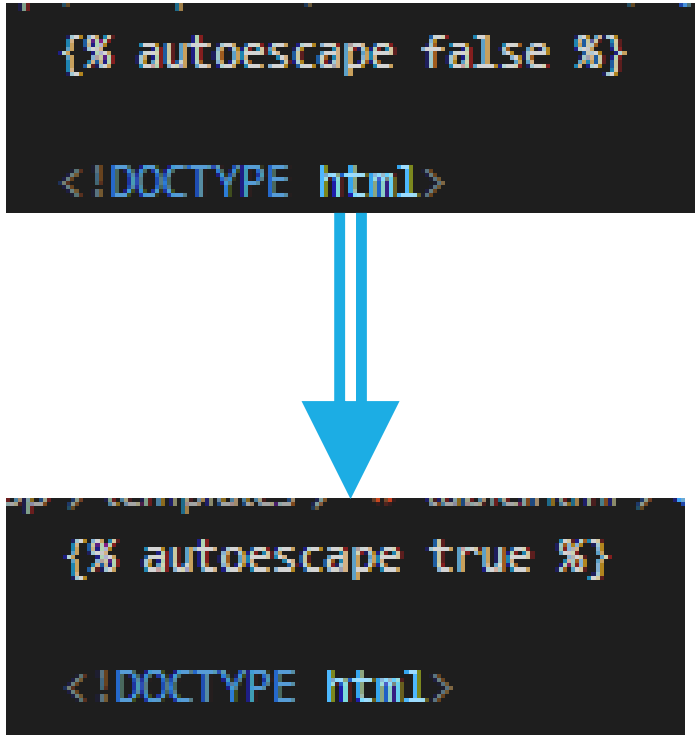
```
'SELECT * FROM data where name=\'' + name +'\‘’
```

Use parameterized query or prepared statements:

- how-and-why-to-use-parameterized-queries
- how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements

# XSS


127.0.0.1:5000
127.0.0.1:5000 says
1
OK


{% autoescape false %}
<!DOCTYPE html>
{% autoescape true %}
<!DOCTYPE html>

# Additional resources:

https://tryhackme.com/

https://portswigger.net/web-security

https://academy.hackthebox.com/

https://cryptohack.org/

https://www.hacker101.com/

# Thank You!

For Your Attention

Piotr Kontowicz
piotr.kontowicz@put.poznan.pl