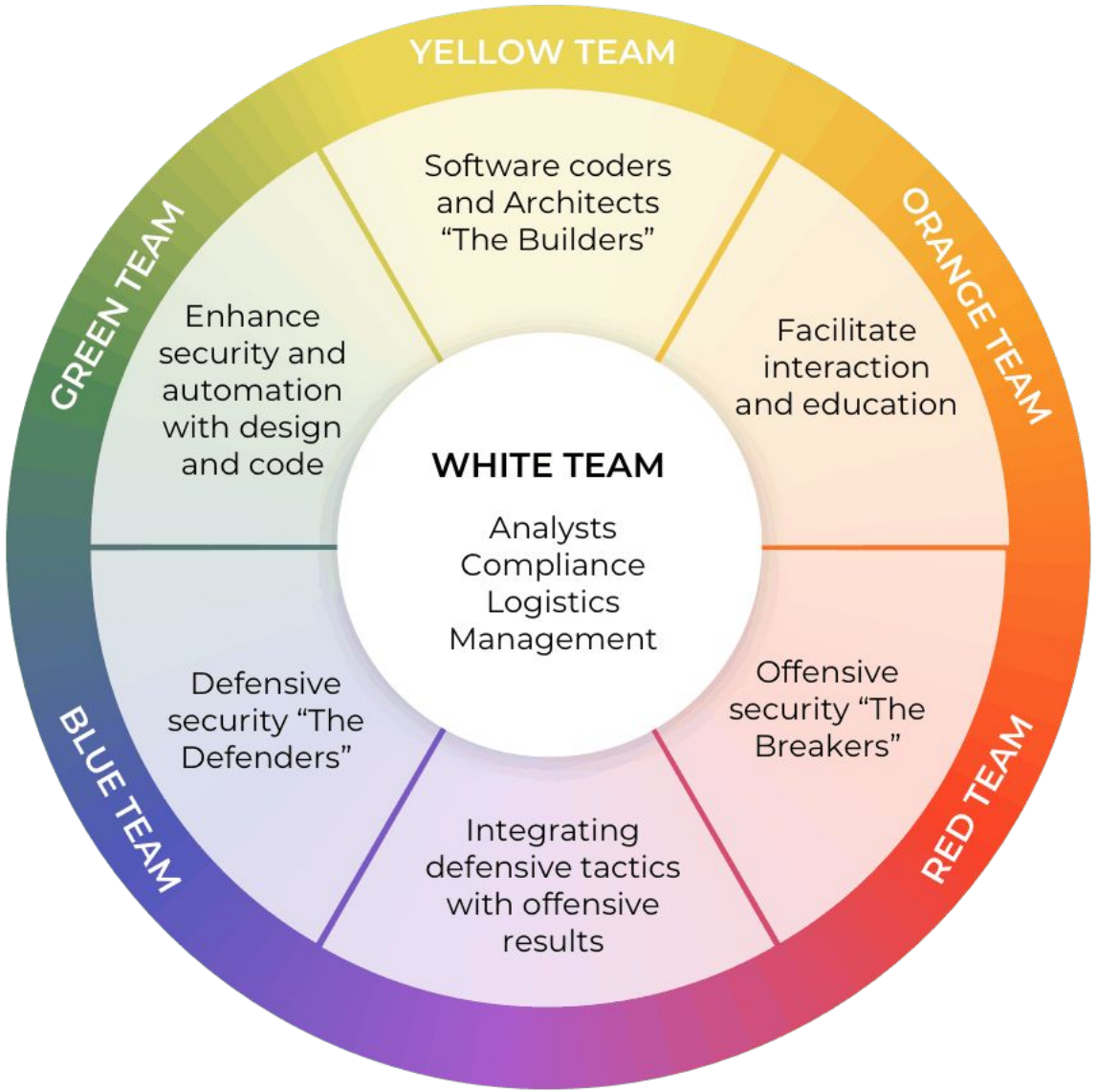
YELLOW TEAMS

Deel 1

PXL DIGITAL


YELLOW TEAM
Software coders and Architects "The Builders"
ORANGE TEAM
Facilitate interaction and education
RED TEAM
Offensive security "The Breakers"
Integrating defensive tactics with offensive results
BLUE TEAM
Defensive security "The Defenders"
GREEN TEAM
Enhance security and automation with design and code
WHITE TEAM
Analysts
Compliance
Logistics
Management

PXL DIGITAL


YELLOW TEAM
Software Builders
Application Developers
Software Engineers
System Architects

INPUT SANITIZATION

DON’T TRUST THE CLIENT

UPDATE YOUR SHIT

Wijsheid van Beckers™

PXL DIGITAL

Who is the OWASP® Foundation?

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

- Tools and Resources
- Community and Networking
- Education & Training

For nearly two decades corporations, foundations, developers, and volunteers have supported the OWASP Foundation and its work. Donate, Join, or become a Corporate Member today.

Project Spotlight: Top 10 Proactive Controls

OWASP PROActive CONTROLS FOR DEVELOPERS v 3.0

OWASP Top 10 Proactive Controls describes the most important control and control categories that every architect and developer should absolutely, 100% include in every project. The Top 10 Proactive Controls are **by** developers **for** developers to assist those new to secure development.

- C1: Define Security Requirements
- C2: Leverage Security Frameworks and Libraries
- C3: Secure Database Access
- C4: Encode and Escape Data
- C5: Validate All Inputs
- C6: Implement Digital Identity
- C7: Enforce Access Controls
- C8: Protect Data Everywhere
- C9: Implement Security Logging and Monitoring
- C10: Handle All Errors and Exceptions

For more information, see the complete document at the Top 10 Proactive Controls project page

Featured Chapter: Sydney

OWASP Sydney Chapter Leadership are on fire! Ric Campo, Jack Guildford, and Sherry Liu took over the Sydney OWASP chapter in July 2020 and hit the ground running and have not looked back. See virtual meeting activity reach a new level. December started with OWASP Jingleton Hack for Beginners. This provided cybersecurity beginners with the opportunity to practice their skills and learn the basics of web application penetration testing. February experience their version of Hack the Box's King of the Hill called "Fight Club." This is a team event but if you do not have a team don't worry. One will be found for you! See Meetup for all details.

PROACTIVE CONTROLS

C1: Define Security Requirements
C2: Leverage Security Frameworks and Libraries
C3: Secure Database Access
C4: Encode and Escape Data
C5: Validate All Inputs
C6: Implement Digital Identity
C7: Enforce Access Controls
C8: Protect Data Everywhere
C9: Implement Security Logging and Monitoring
C10: Handle All Errors and Exceptions

https://github.com/OWASP/ASVS/tree/v4.0.2#latest-stable-version---402

PXL DIGITAL

C1 Define Security Requirements

- OWASP Application Security Verification Standard (ASVS)
- = Statement of needed security functionality
- Misuse cases <-> User stories
- Verifiable

2.1.9 Verify there are no default passwords in use for the application framework or any components used by the application (such as “admin/password”)

PXL DIGITAL

OWASP

Table of Contents

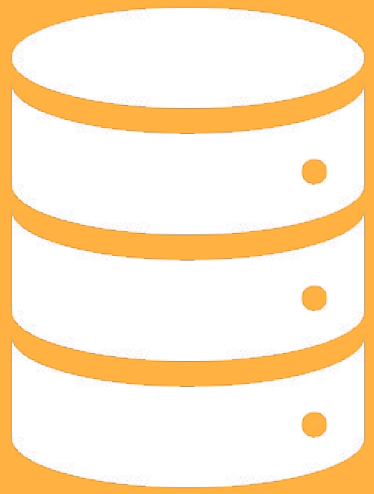
C2 Leverage Security Frameworks and Libraries

- Use libraries and frameworks from trusted sources that are actively maintained
- Maintain inventory catalog
- Proactively keep up-to-date (OWASP dependency check, Retire.js, ...)
- Encapsulate library and expose only the required behaviour

PXL DIGITAL

C3 Secure Database Access

- Secure queries (bobby-tables.com)
- Secure configuration
- Secure authentication
- Secure Communication



PXL DIGITAL

PROACTIVE CONTROLS

C4: Encode and Escape Data

Eg. Encoding: **<** wordt **&lt**

Eg. Escaping: **\”**

Mostly to protect against injection attacks

C5: Validate All Inputs

Syntactisch (eg. correct date format) + Semantic (eg. Start date before end date)

PXL DIGITAL

C6 Implement Digital Identity

- Level 1: Passwords
- Level 2: MFA
- Level 3: Cryptographic Based Authentication
- Session management

PXL DIGITAL

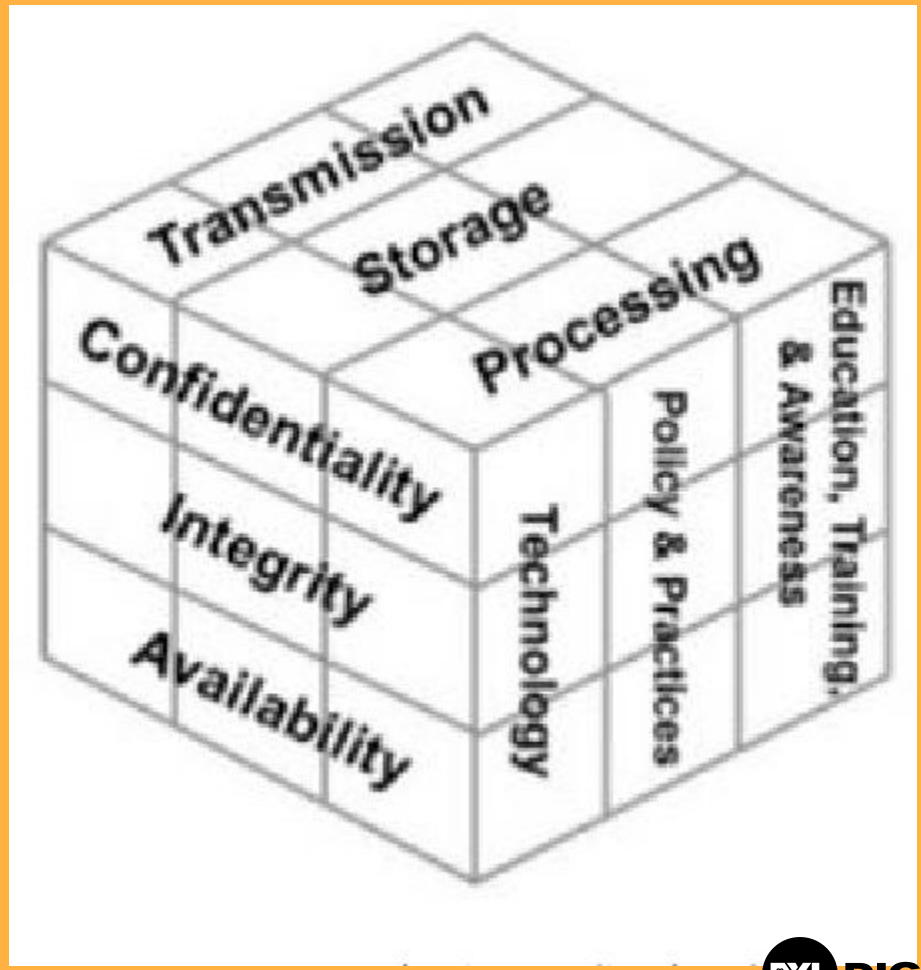
PROACTIVE CONTROLS

C7: Enforce Access Controls

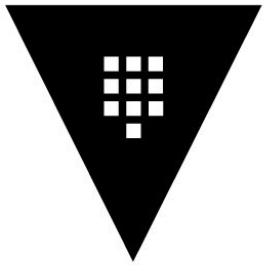
- Design up front
- Force all requests -> Access Control check
- Deny by default
- Principle of least privilege
- Don’t hardcode rules
- Log

C8: Protect Data Everywhere

- Encrypt in transit
- Encrypt @ rest
- Key/secrets managements
- Classification



PXL DIGITAL



HashiCorp Vault

Qualys® SSL Labs

PXL DIGITAL

PROACTIVE CONTROLS

C9: Implement Security Logging and Monitoring

C10: Handle All Errors and Exceptions

```
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
  goto fail;
  goto fail;
... other checks ...
fail:
  ... buffer frees (cleanups) ...
  return err;
```

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'select * from users; --' and password = '' at line 1

PXL DIGITAL

INPUT SANITIZATION

DON’T TRUST THE CLIENT

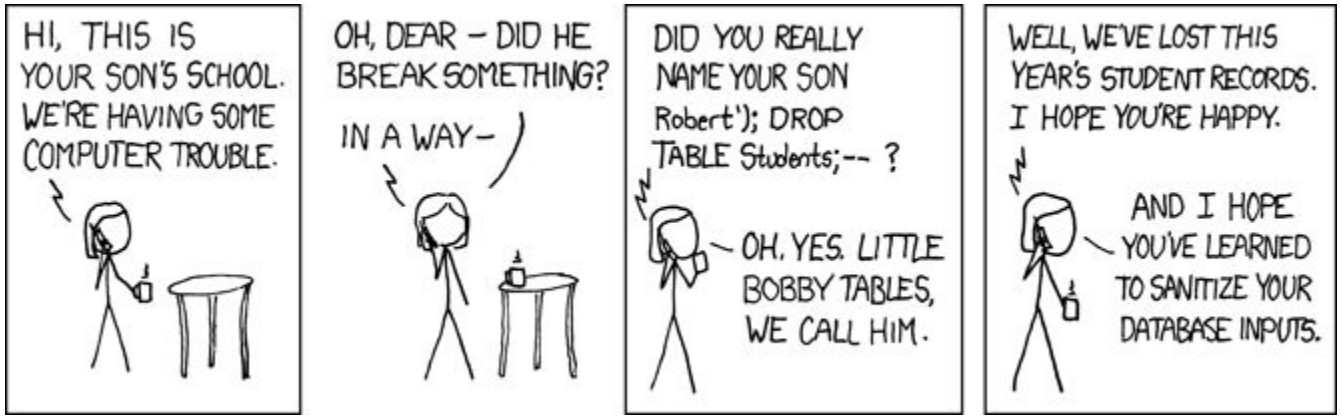
UPDATE YOUR SHIT

Wijsheid van Beckers™

PXL DIGITAL

Top 10 Web Application Security Risks

1. **Injection**. Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. **Broken Authentication**. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
3. **Sensitive Data Exposure**. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
4. **XML External Entities (XXE)**. Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
5. **Broken Access Control**. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
6. **Security Misconfiguration**. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
7. **Cross-Site Scripting (XSS)**. XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data


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.

A1:2017-Injection

Languages: [en] de

← 2017 Top 10

OWASP Top Ten 2017
PDF version

A2:2017-Broken Authentication →

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.		Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.	

Is the Application Vulnerable?

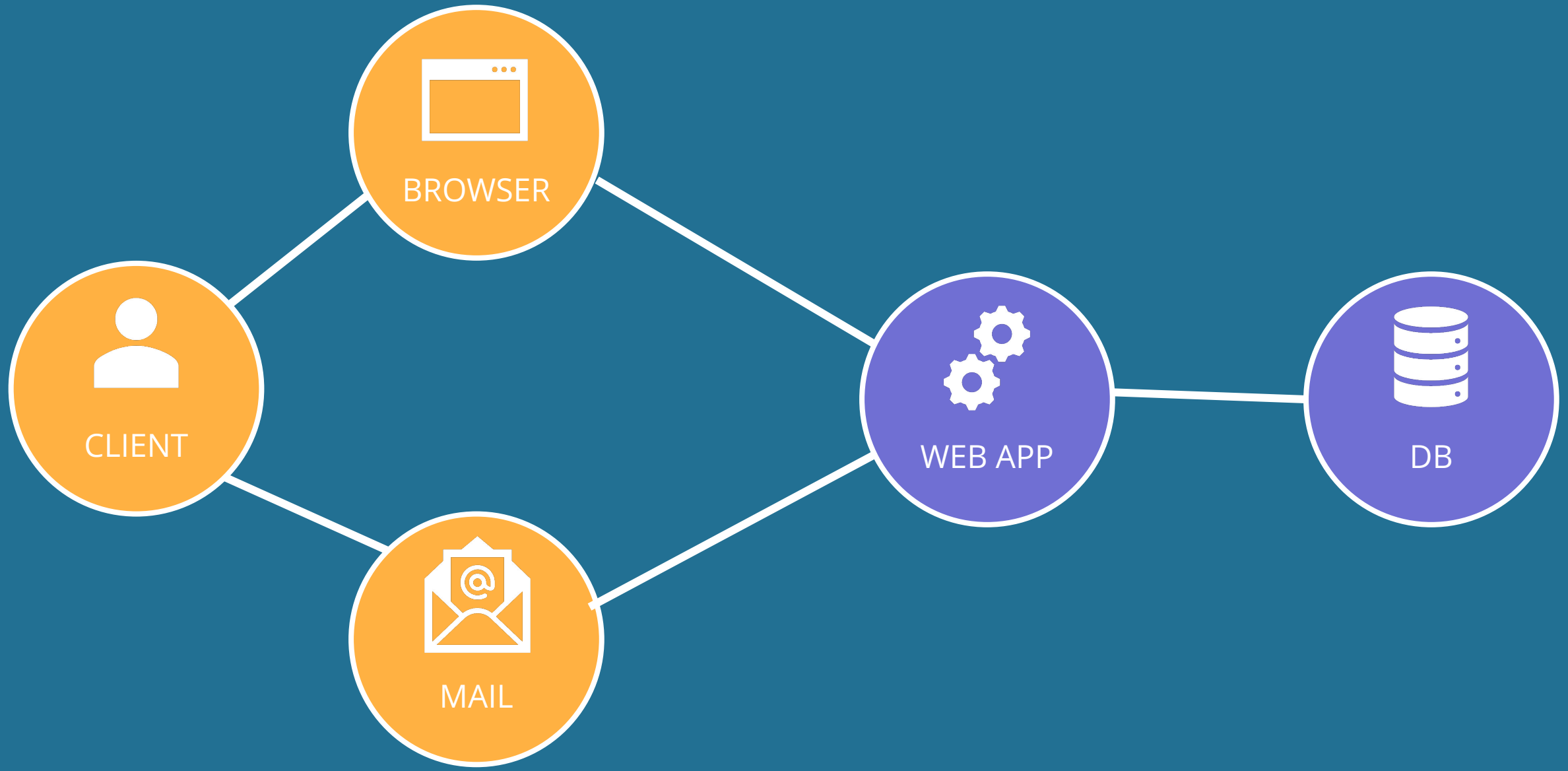
An application is vulnerable to attack when:
* User-supplied data is not validated, filtered, or sanitized by the application.
* Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
* Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
* Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.

Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if

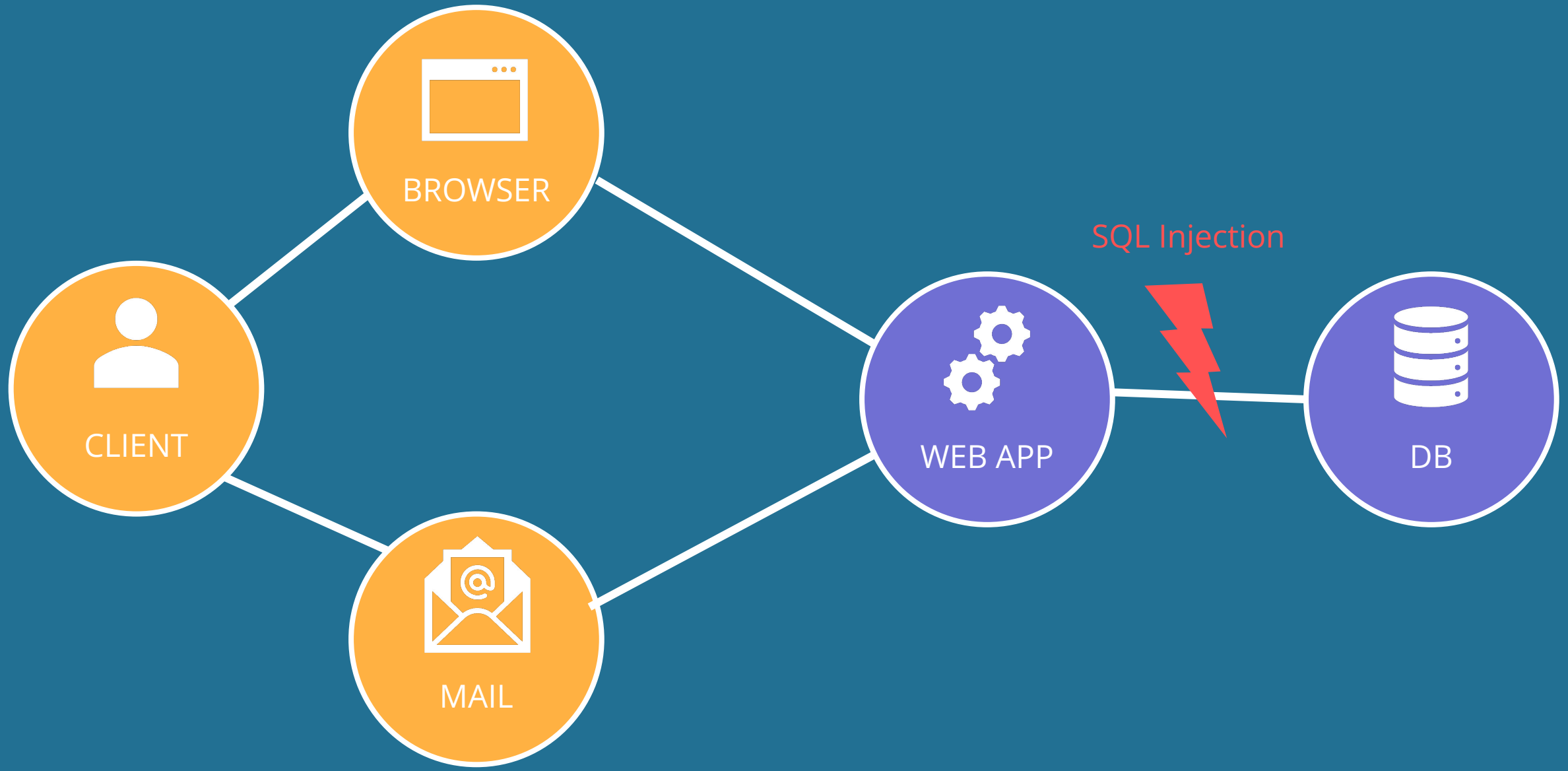
How to Prevent

Preventing injection requires keeping data separate from commands and queries.
* The preferred option is to use a safe API, which avoids the use of the interpreter entirely or provides a parameterized interface, or migrate to use Object Relational Mapping Tools (ORMs).

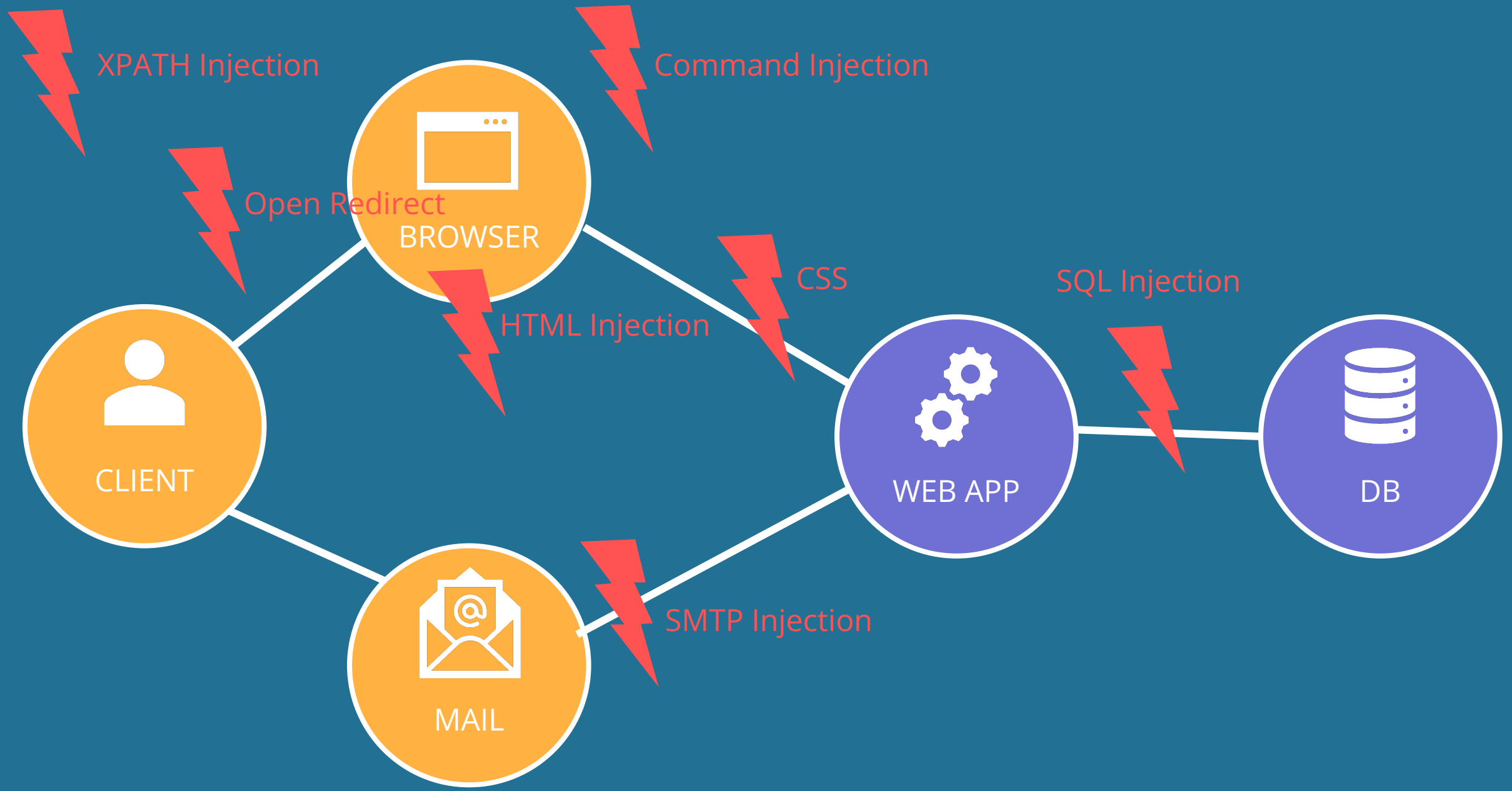
Note: Even when parameterized, stored procedures can still introduce SQL injection if PL/SQL or T-SQL concatenates queries and data, or executes hostile data with EXECUTE IMMEDIATE or exec().
* Use positive or “whitelist” server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications.
* For any residual dynamic queries, escape special characters using the specific escape syntax for that interpreter.



PXL DIGITAL



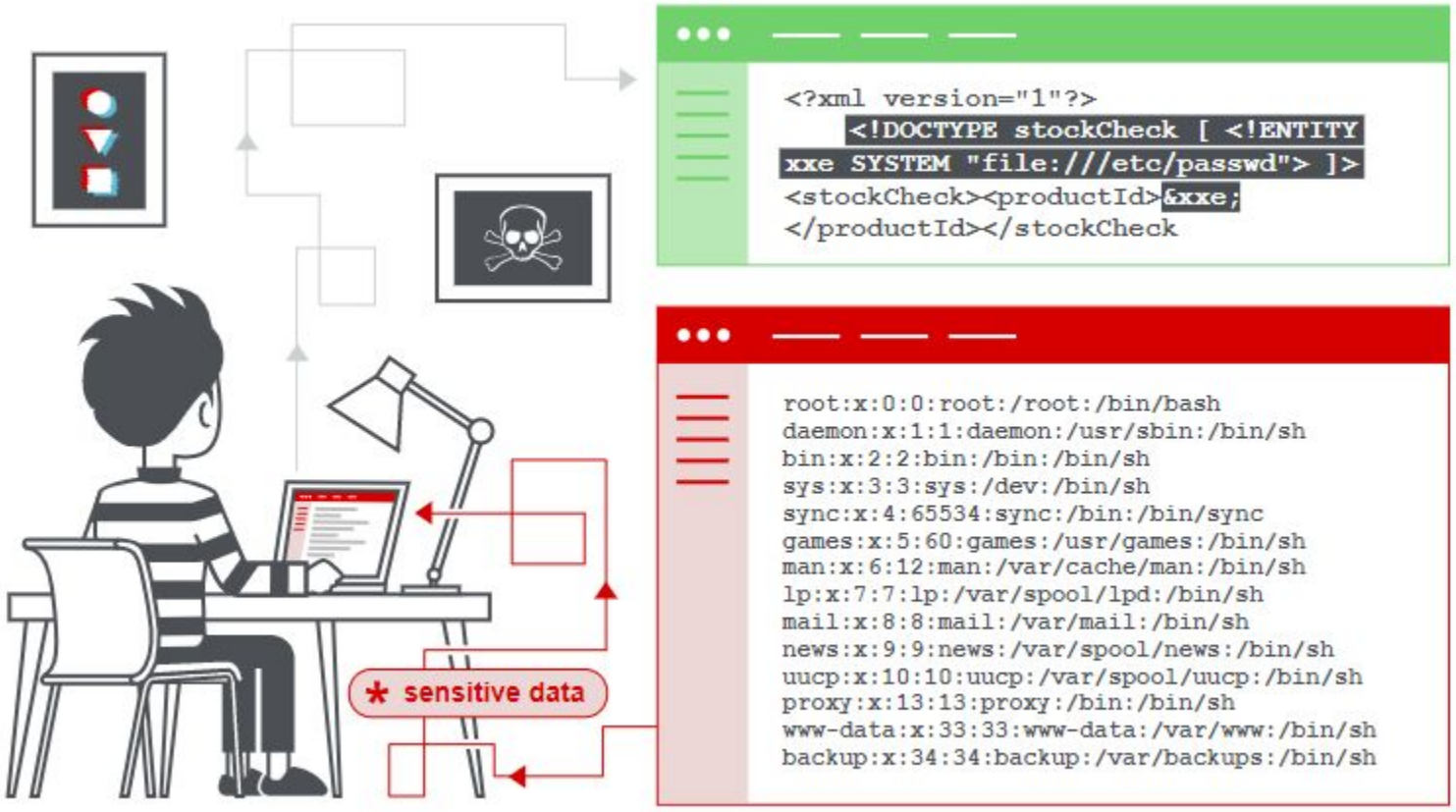
PXL DIGITAL


XPATH Injection
Command Injection
Open Redirect
BROWSER
CSS
HTML Injection
SQL Injection
CLIENT
WEB APP
DB
SMTP Injection
MAIL

PXL DIGITAL

XML External Entity (XXE) Injection

In top 10 als: Security Misconfiguration

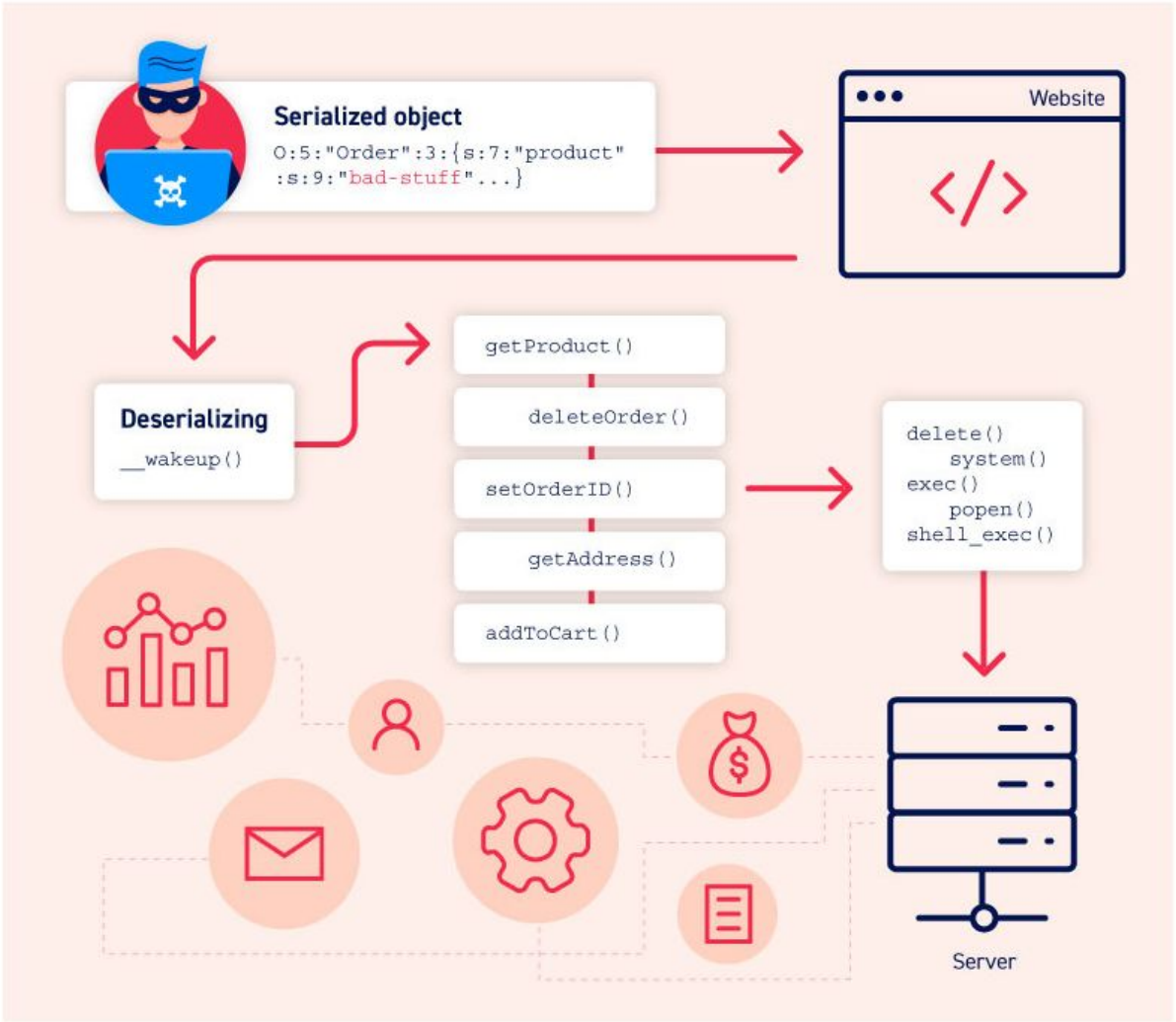


https://portswigger.net/web-security/xxe

PXL DIGITAL

Insecure Deserialization

In top 10 als: Software and data integrity failures



https://portswigger.net/web-security/deserialization

PXL DIGITAL

OWASP

CHEAT SHEET SERIES PROJECT

Life is too short • AppSec is tough • Cheat!

PXL DIGITAL

INPUT SANITIZATION

DON’T TRUST THE CLIENT

UPDATE YOUR SHIT

Wijsheid van Beckers™

PXL DIGITAL

- NDC Conferences 2020: The OWASP Top Ten Proactive Controls 2018 – YouTube, Jim Manico: https://www.youtube.com/watch?v=VmkEUpY6HeY
- NDC Conferences 2020: From the OWASP Top Ten(s) to the OWASP ASVS – YouTube, Jim Manico: https://www.youtube.com/watch?v=CCIO3GOaFe0
- GOTO 2018: From the OWASP Top Ten(s) to The OWASP ASVS – YouTube, Philippe De Ryck: https://pragmaticwebsecurity.com/talks/owaspasvs.html
- All Things WebSecurity related: https://pragmaticwebsecurity.com/recordings.html
- Play by Play, OWASP top 10 2017 https://app.pluralsight.com/library/courses/play-by-play-owasp-top-ten-2017/table-of-contents

PXL DIGITAL