OWASP Top 10 Vulnerabilities:

Injection and Insecure Design

Daniela Tomás	up202004946@edu.fc.up.pt
Diogo Nunes	up202007895@edu.fc.up.pt
João Veloso	up202005801@edu.fc.up.pt

Introduction

This presentation analyzes two websites: one that highlights unsafe behaviors vulnerable to various types of cyberattacks, and another that shows safe precautions to prevent these vulnerabilities.

Injection (A03:2021)

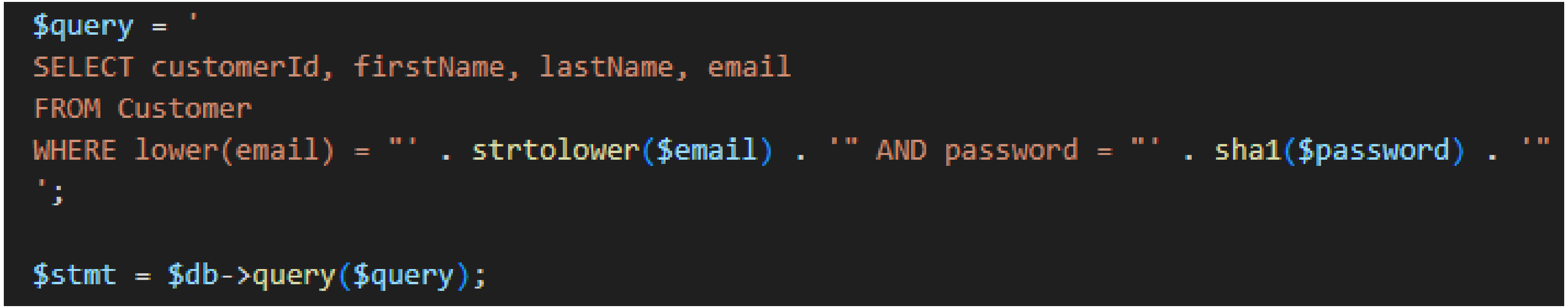
- Injections occur when an application sends untrusted data to an interpreter. This can trick the interpreter into executing unintended commands or accessing data without the required authorization.

- Injection flaws are very prevalent, particularly in legacy code, and are often found in SQL queries, LDAP queries, XPath queries, OS commands, program arguments, etc.

- Injection flaws are normally discovered when there is the ability to examine code directly, but they can also be found via testing, albeit with more difficulty. There are also tools to help discover them, such as fuzzers and scanners.

SQL Injection

Lack of Sanitization in Login

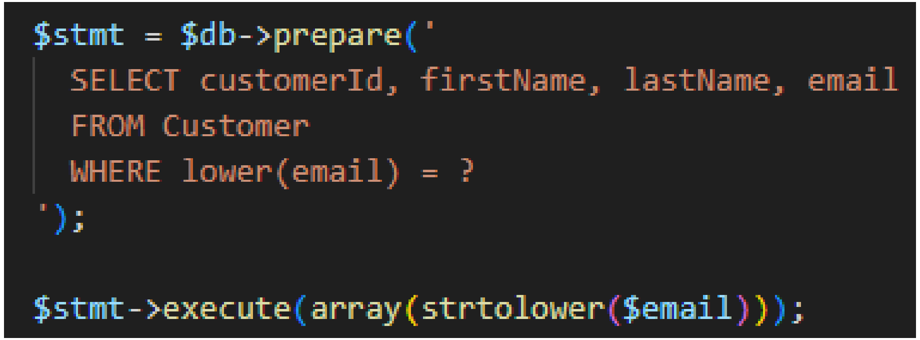


```
$query = '
SELECT customerId, firstName, lastName, email
FROM Customer
WHERE lower(email) = "' . strtolower($email) . '" AND password = "' . sha1($password) . '"
';

$stmt = $db->query($query);
```

SQL Injection

Countermeasures

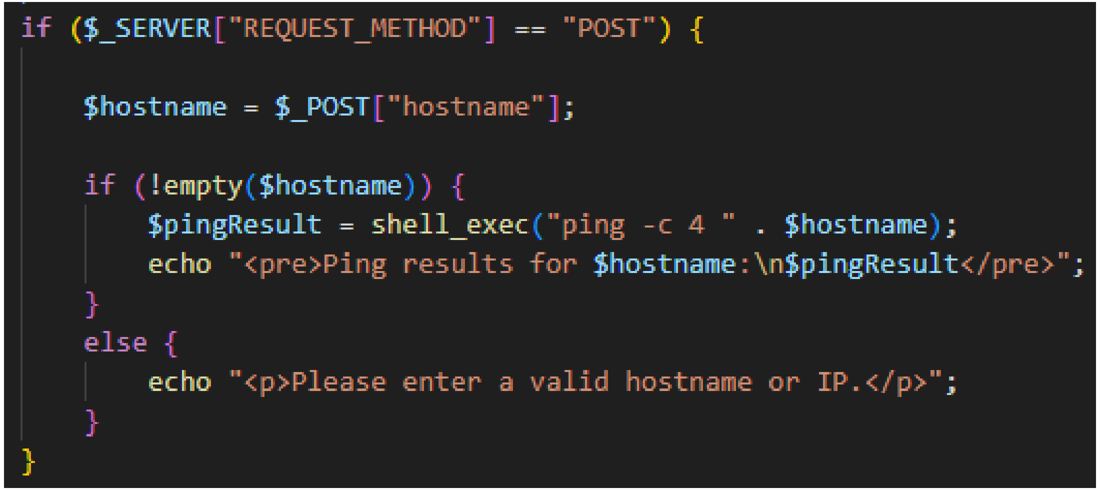
- Use of stored procedures
- Escaping all user supplied input



```
$stmt = $db->prepare('
  SELECT customerId, firstName, lastName, email
  FROM Customer
  WHERE lower(email) = ?
');

$stmt->execute(array(strtolower($email)));
```

Command Injection

Lack of Sanitization in Shell Commands

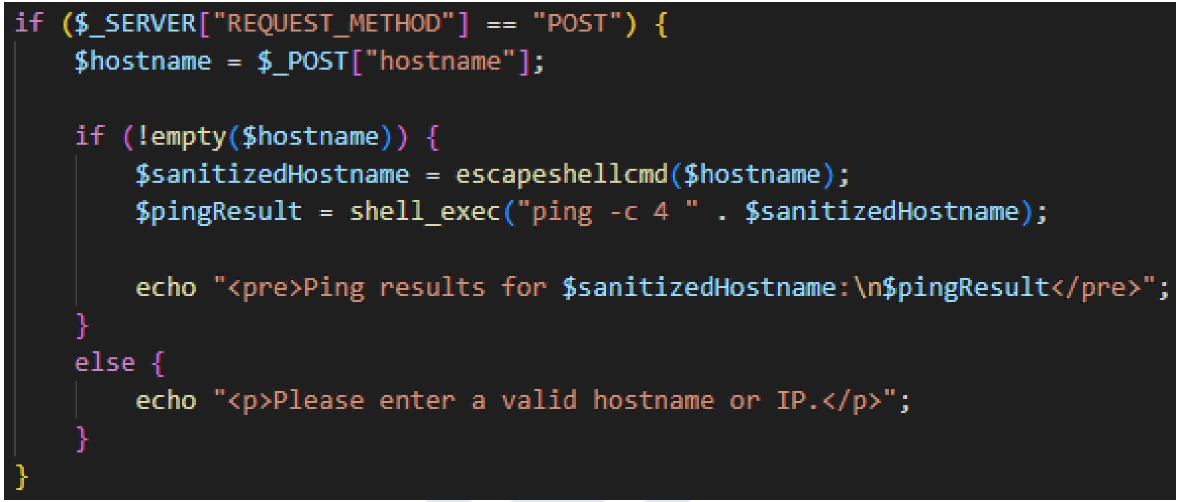


```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $hostname = $_POST["hostname"];

    if (!empty($hostname)) {
        $pingResult = shell_exec("ping -c 4 " . $hostname);
        echo "<pre>Ping results for $hostname:\n$pingResult</pre>";
    }
    else {
        echo "<p>Please enter a valid hostname or IP.</p>";
    }
}
```

Command Injection

Countermeasures

- Sanitized hostname in ping



```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $hostname = $_POST["hostname"];

    if (!empty($hostname)) {
        $sanitizedHostname = escapeshellcmd($hostname);
        $pingResult = shell_exec("ping -c 4 " . $sanitizedHostname);

        echo "<pre>Ping results for $sanitizedHostname:\n$pingResult</pre>";
    }
    else {
        echo "<p>Please enter a valid hostname or IP.</p>";
    }
}
```

Insecure Design (A04:2021)

- Insecure design vulnerabilities result from inadequate system architecture or design choices that provide security flaws. Attackers take advantage of these flaws to compromise the system’s availability, confidentiality, or integrity. Injection vulnerabilities can result from insecure design practices.

Password Encryption

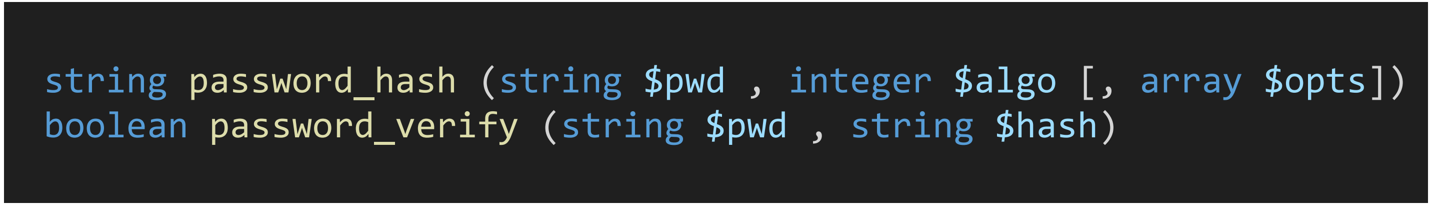
Weak Passoword Encryption (SHA1)

- The usage of SHA1 for password encryption is considered weak due to its susceptibility to brute-force and rainbow table attacks.
- In PHP, the *sha1* function generates a SHA-1 hash of a string.

Password Encryption

Countermeasures

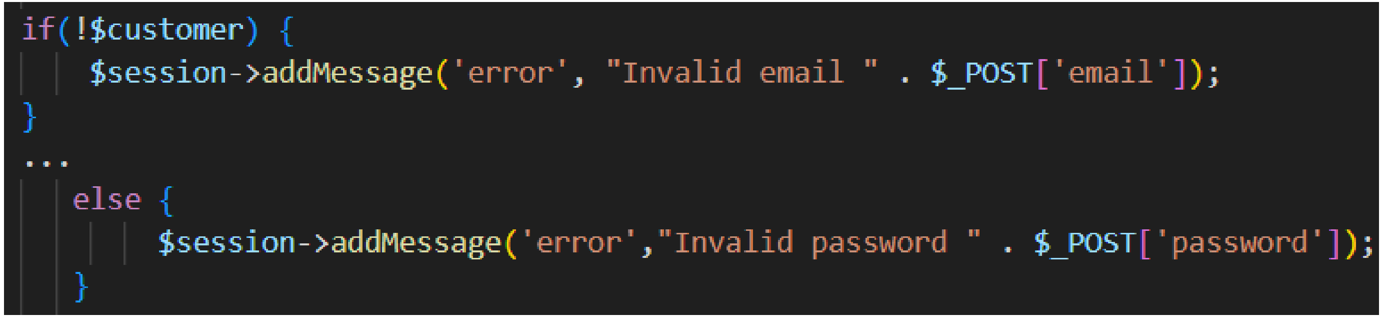
- Stronger Password Hashing Function (bcrypt)



```
string password_hash (string $pwd , integer $algo [, array $opts])
boolean password_verify (string $pwd , string $hash)
```

Error Messages

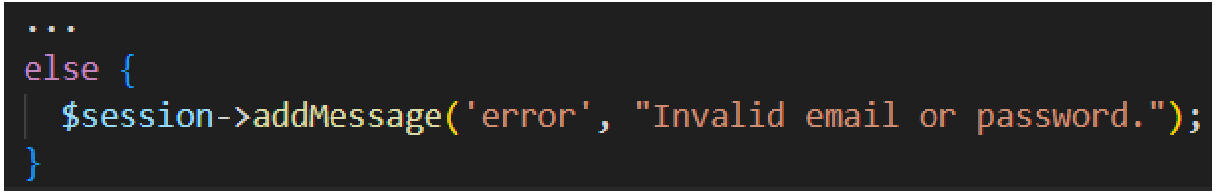
Detailed Error Messages



```
if(!$customer) {
    $session->addMessage('error', "Invalid email " . $_POST['email']);
}
...
    else {
        $session->addMessage('error',"Invalid password " . $_POST['password']);
    }
```

Error Messages

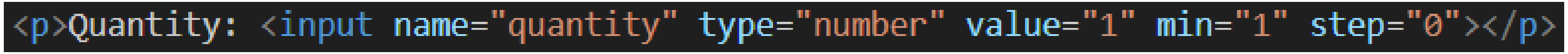
Countermeasures



```
...
else {
  $session->addMessage('error', "Invalid email or password.");
}
```

Stock Limits

No Maximum Quantity Limited to Stock

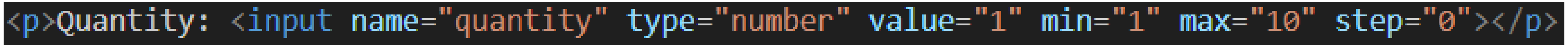


```html
<p>Quantity: <input name="quantity" type="number" value="1" min="1" step="0"></p>
```

Stock Limits

Countermeasures

- Enforcement of (Maximum) Stock Limits



```
<p>Quantity: <input name="quantity" type="number" value="1" min="1" max="10" step="0"></p>
```

Conclusion

A comprehensive strategy involving strong countermeasures like input validation, prepared statements, secure hashing functions, and system exposure reduction is crucial to combat injection and insecure design flaws in web applications.

References

- https://owasp.org/Top10/A04 2021-Insecure Design/
- https://owasp.org/Top10/A03 2021-Injection/
- The MITRE Corporation, Common Weakness Enumeration, “CWE-89: Improper Neutralization of Special Elements used in na SQL Command (’SQL Injection’)”. https://cwe.mitre.org/data/definitions/89.html, 2023.
- The MITRE Corporation, Common Weakness Enumeration, “CWE-78: Improper Neutralization of Special Elements used in na OS Command (’OS Command Injection’)”. https://cwe.mitre.org/data/definitions/78.html, 2023.
- The MITRE Corporation, Common Weakness Enumeration, “CWE-257: Storing Passwords in a Recoverable Format”. https://cwe.mitre.org/data/definitions/257.html, 2023.
- The MITRE Corporation, Common Weakness Enumeration, “CWE-209: Generation of Error Message Containing Sensitive Information”. https://cwe.mitre.org/data/definitions/209.html, 2023.
- https://owasp.org/www-project-top-ten/
- https://sectigostore.com/blog/what-is-owasp-what-are-the-owasp-top-10- vulnerabilities/
- Dafydd Stuttard and Marcus Pinto, The Web Application Hacker’s Handbook, 2nd ed, 2011.