PENTESTING

Uriel Martin eta Oihan Torrontegi

2025eko azaroaren 14an

Gaien Aurkibidea

1 Sarrera

Pentesting-a ebaluazio kontrolatu bat da, aplikazio batean kalteberatasunak identifikatzeko eta lehenesteko, ustiatu aurretik. Lan honetan, gure web sistemaren analisia egingo dut ZAPekin, OWASP Top 10 (2021) delakoaren jardunbidea oionarri. Zerbitzua hedatu eta http://localhost: 81/gainean eskaneatze automatikoa egin ondoren, ohiko ahuleziak antzeman ziren: CSRF babesik eza, segurtasun-goiburuak (CSP, X-Frame-Options, X-Content-Type-Options), atributu segururik gabeko cookieak (HttpOnly eta SameSite), zerbitzariaren informazio-erakusketa (Server eta X-Powered-By), eta faltak autentifikazioan eta saioen kudeaketan. Aurreko entregan, hasierako diagnostiko bat aurkezten genuen, eta, horretan oinarrituta, arrisku bakoitzak zer dakarren eta arrisku hori arintzeari nola eman lehentasuna deskribatu genituen. Oraingoan aurrekoak zuzendu ditugu, eta ahuleziak zuzzentzeko egin ditugun pausoak azalduko ditugu.

2 Sarbide-kontrola haustea (A01:2021)

Webgunean erabiltzaile estandarraren eta administratzailearen arteko bereizketa inplementatu dugu: roletan oinarritutako sarbide-kontrola gehituz, eta baimena zerbitzarian aplikatzen. "lehenespenez ukatu" printzipioa aplikatu da, endpointak eta eragiketa sentikorrak mugatu dira eta datu konfidentzialak babestu dira. Neurri horiekin, baimendu gabeko ekintzen arriskua eta informazio mugatua eskuratzea arindu dugu.

Horretarako gure datu basea aldatu dugu, `is_admin` zutabea sortu dugu, honek erabiltzaile bat administratzaile gisatzat edo ez gorde ahal dugu, 1 edo 0 bat erabiltzen, hurrenez hurren:

```
ALTER TABLE users ADD COLUMN is_admin TINYINT(1) NOT NULL DEFAULT 0;
UPDATE users SET is_admin = 1 WHERE username = 'juan';
```

`login.php` ere aldatu behar izan dugu, bertan `is_admin` zutabea kontsultatzen dugu ea administratzailea den ala ez, administratzaileak beste metodo batzuk dituelako. `registrer.php` ere aldatu dugu, erabiltzaile berri batek kontu berri bat sortzen duenean `is_admin` 0 dela ziurtatuz:

```
try {
       $stmt = $mysqli->prepare("INSERT INTO users (fullname, dni,
           ↪ phone, birthdate, email, username, password, is_admin)
           ↪ VALUES (?,?,?,?,?,?,SHA2(?,256), 0)");
       if (!$stmt) throw new Exception('Error prepare:
           ↪ '.$mysqli->error);
           $stmt->bind_param('sssssss', $fullname, $dni, $phone,
               ↪ $birthdate, $email, $username, $password);
           if ($stmt->execute()) {
               header('Location: /login');
               exit;
           } else {
               $errors[] = 'Errorea erregistratzean: '.$stmt->error;
           }
       } catch (Exception $e) {
           $errors[] = 'Excepction insertatzean: '.$e->getMessage();
       }
```

Azkenik administratziale eragiketak babestuko ditugu, aldatu, kendu eta gehitu eragiketak. Orduan, `add_item.php`, `modify_item.php` eta `delete_item.php`-n sesio baieztapena ipiniko dugu hasieran.

```
session_start();
if (!isset($_SESSION['is_admin']) || $_SESSION['is_admin'] != 1) {
    http_response_code(403);
    die('Ez daukazu baimenik');
}
```

Administratzaileek soilik diska berriak gehitu ahal dituzte, orduan `index.php`-n bakarrik hauek ikusi ahal izango dute "Diska berria"esteka `is_admin` 1 baldin bada.

```
session_start();
.
.
.
<?php if (isset($_SESSION['is_admin']) && $_SESSION['is_admin'] == 1): ?>
    <a href="/add_item">Diska berria jarri</a>
<?php endif; ?>
```

3 Akats kriptografikoak (A02:2021)

Ahultasun hau kriptografiaren erabileraren edo haren gabeziaren ondorioz sortzen zen: pasahitz finkoak, ahul zifratze-algoritmoak, gakoen kudeaketa txarra, datuen transmisio ez-zifratua eta ziurtagiriak gaizki egiaztatzea. Gure sisteman ez zeuden datu biltegiratzeak eta transmisioak behar bezala zifratuak, eta horrek informazio sentikorra lortzea eragin zezakeen. Arazoa konpontzeko, biltegiratutako eta transmisio bidezko datuetarako zifratze-mekanismo seguruenak ezarri ditugu, pasahitz eta gakoen kudeaketa egokia ezarri, eta ziurtagiri eta egiaztapen prozedurak indartu ditugu. Hainbat funtzio eta hainbat script ipiniko ditugu `db.php` errotzat hartzen, eta aldaketa puntualak aplikatuko gainerakoetan. Hauek dira `db.php`-n egingo ditgun aldaketak:

```
$mysqli = mysqli_init();
if ($USE_SSL) {
    $ssl_key = getenv('DB_SSL_KEY') ?: null;
    $ssl_cert = getenv('DB_SSL_CERT') ?: null;
    $ssl_ca = getenv('DB_SSL_CA') ?: null;
    if ($ssl_key || $ssl_cert || $ssl_ca) {
        $mysqli->ssl_set($ssl_key, $ssl_cert, $ssl_ca, null, null);
    }
    $client_flags = MYSQLI_CLIENT_SSL;
} else {
    $client_flags = 0;
}

if (!$mysqli->real_connect($DB_HOST, $DB_USER, $DB_PASS, $DB_NAME,
    ↪ ini_get("mysqli.default_port"), null, $client_flags)) {
    die('DB connect error: ' . mysqli_connect_error());
}
$mysqli->set_charset("utf8mb4");


if (php_sapi_name() !== 'cli') {
    $secure = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on')
        ↪ || getenv('FORCE_HTTPS') === '1';
    session_set_cookie_params([
        'lifetime' => 0,
        'path' => '/',
        'domain' => $_SERVER['HTTP_HOST'] ?? '',
        'secure' => $secure,
        'httponly' => true,
        'samesite' => 'Lax'
    ]);
    if (session_status() === PHP_SESSION_NONE) session_start();
}

if (!extension_loaded('sodium')) {
```

```
}

function get_app_key(): string {
    $b64 = getenv('APP_ENC_KEY') ?: '';
    if ($b64 === '') {
        throw new Exception('Ez dago APP_ENC_KEY');
    }
    $key = base64_decode($b64, true);
    if ($key === false || strlen($key) !==
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_KEYBYTES) {
        throw new Exception('APP_ENC_KEY baliogabea da (32 byte base64
            ↪ izan behar du');
    }
    return $key;
}

function encrypt_field(string $plaintext): string {
    $key = get_app_key();
    $nonce =
        ↪ random_bytes(SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES);
    $cipher =
        ↪ sodium_crypto_aead_xchacha20poly1305_ietf_encrypt($plaintext,
        ↪ '', $nonce, $key);
    return base64_encode($nonce . $cipher);
}

function decrypt_field(string $b64): ?string {
    $raw = base64_decode($b64, true);
    if ($raw === false || strlen($raw) <
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES) return
        ↪ null;
    $key = get_app_key();
    $nonce = substr($raw, 0,
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES);
    $cipher = substr($raw,
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES);
    $plain = sodium_crypto_aead_xchacha20poly1305_ietf_decrypt($cipher,
        ↪ '', $nonce, $key);
    if ($plain === false) return null;
    return $plain;
}
?>
```

Orain zatika azalduko da aurreko kodea. Hasteko, lehengo kode zatiak, TLS/SSL konexio bat konfiguratu eta eskatzen du datu-basera, iragaitzazko datuak babesteko.

```
$mysqli = mysqli_init();
if ($USE_SSL) {
    $ssl_key = getenv('DB_SSL_KEY') ?: null;
```

```
    $ssl_cert = getenv('DB_SSL_CERT') ?: null;
    $ssl_ca = getenv('DB_SSL_CA') ?: null;
    if ($ssl_key || $ssl_cert || $ssl_ca) {
        $mysqli->ssl_set($ssl_key, $ssl_cert, $ssl_ca, null, null);
    }
    $client_flags = MYSQLI_CLIENT_SSL;
} else {
    $client_flags = 0;
}
```

Hurrengo kode zatiak, saioaren cookieak atributu seguruak izatea ziurtatzen du (soilik HTTPS bidezkoa denean, HttpOnly JS bidez sarbidea blokeatzeko) eta saioa behar bezala hastea.

```
if (php_sapi_name() !== 'cli') {
    $secure = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on')
        ↪ || getenv('FORCE_HTTPS') === '1';
    session_set_cookie_params([
        'lifetime' => 0,
        'path' => '/',
        'domain' => $_SERVER['HTTP_HOST'] ?? '',
        'secure' => $secure,
        'httponly' => true,
        'samesite' => 'Lax'
    ]);
    if (session_status() === PHP_SESSION_NONE) session_start();
}
```

Orain, egiaztatzen da libsodium luzapena eskuragarri dagoela eta funtzioak definitzen ditu gakoa maneiatzeko eta eremuak zifratzeko/deszifratzeko: `encrypt_field() / decrypt_field()` eskaintzen du ausazko nonce bat erabiliz datuak zifratu eta deszifratzeko, eta base64 edo null itzultzen du huts egiten badu.

```
    if (!extension_loaded('sodium')) {

}

function get_app_key(): string {
    $b64 = getenv('APP_ENC_KEY') ?: '';
    if ($b64 === '') {
        throw new Exception('Ez dago APP_ENC_KEY');
    }
    $key = base64_decode($b64, true);
    if ($key === false || strlen($key) !==
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_KEYBYTES) {
        throw new Exception('APP_ENC_KEY baliogabea da (32 byte base64
            ↪ izan behar du');
    }
    return $key;
```

```
}

function encrypt_field(string $plaintext): string {
    $key = get_app_key();
    $nonce =
        ↪ random_bytes(SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES);
    $cipher =
        ↪ sodium_crypto_aead_xchacha20poly1305_ietf_encrypt($plaintext,
        ↪ '', $nonce, $key);
    return base64_encode($nonce . $cipher);
}

function decrypt_field(string $b64): ?string {
    $raw = base64_decode($b64, true);
    if ($raw === false || strlen($raw) <
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES) return
        ↪ null;
    $key = get_app_key();
    $nonce = substr($raw, 0,
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES);
    $cipher = substr($raw,
        ↪ SODIUM_CRYPTO_AEAD_XCHACHA20POLY1305_IETF_NPUBBYTES);
    $plain = sodium_crypto_aead_xchacha20poly1305_ietf_decrypt($cipher,
        ↪ '', $nonce, $key);
    if ($plain === false) return null;
    return $plain;
}
```

Bestalde, libsodium kriptografia liburutegi bat da, zifratze-, sinatze- eta hashing-eragiketa arrunten inplementazio erraz eta zuzenak eskaintzen ditu.

4 Injekzioa (A03:2021)

Injekzio-ahultasunak, XSS arriskuak eta CSRF erasoak prebenitzeko beharrezko zuzenketa guztiak aplikatu ditugu. SQL Injekzioa saihesteko, Prepared Statements erabiltzea mantendu eta optimizatu da idazketa-operazio guztietan, eta statement handles-ak modu egokian ixten. XSS Prebentziorako, estandarizatu dugu htmlspecialchars(..., ENT_QUOTES, 'UTF-8') erabiltzea irteera guztietan (erroreak eta formularioen sticky balioak), nabigatzailean kode maltzurrak exekutatzea ekiditeko. Azkenik CSRF babeserako, segurtasun-sistema zentralizatu bat gehitu duugu `db.php` fitxategian, POST eskaerak egiten dituzten formulario guztiak kanpoko ekintzak babesteko.

```
    function csrf_get_token(): string {
    if (session_status() !== PHP_SESSION_ACTIVE) {
        if (php_sapi_name() !== 'cli') {
            session_start();
        }
    }
    if (empty($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf_token'];
}

function csrf_token_input(): string {
    $token = htmlspecialchars(csrf_get_token(), ENT_QUOTES, 'UTF-8');
    return '<input type="hidden" name="csrf_token" value="' . $token .
        ↪ '">';
}

function csrf_validate_request(): bool {
    if ($_SERVER['REQUEST_METHOD'] !== 'POST') return true;
    $sessionToken = $_SESSION['csrf_token'] ?? '';
    $posted = $_POST['csrf_token'] ?? '';
    if (!is_string($posted) || !is_string($sessionToken)) return false;
    return hash_equals($sessionToken, $posted);
}
```

```
    if (!empty($error)) echo "<p
        ↪ style='color:red'>".htmlspecialchars($error, ENT_QUOTES,
        ↪ 'UTF-8')."</p>"
```

5 Segurtasun-konfigurazio ez nahikoa (A05:2021)

Azken berrikuspenen ondoren, segurtasuneko HTTP goiburuak (CSP, X-Frame-Options, X-Content-Type-Options: nosniff) behar bezala konfiguratu ditugu.

Gainera, cookie-ak HttpOnly eta SameSite atributuekin markatu ditugu saioak babesteko, eta Server goiburuko informazioa ezkutatu dugu. `db.php`-n hurrengo aldaketak egin ditugu:

```
    function set_security_headers() {
    $csp = [
        "default-src 'self'",
        "script-src 'self' 'unsafe-inline'",
        "style-src 'self' 'unsafe-inline'",
        "img-src 'self' data:",
        "font-src 'self' data:",
        "connect-src 'self'",
        "object-src 'none'",
        "base-uri 'self'",
        "form-action 'self'",
        "frame-ancestors 'self'"
    ];
    header('Content-Security-Policy: ' . implode('; ', $csp));
    header('X-Frame-Options: SAMEORIGIN');
    header('X-Content-Type-Options: nosniff');
    header('Referrer-Policy: strict-origin-when-cross-origin');
    header("Permissions-Policy: geolocation=(), camera=(),
        ↪ microphone=()");
    $secure = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on')
        ↪ || getenv('FORCE_HTTPS') === '1';
    if ($secure) {
        header('Strict-Transport-Security: max-age=31536000;
            ↪ includeSubDomains; preload');
    }
    header('Server: SecureApp');
    header_remove('X-Powered-By');
}
```

- Content-Security-Policy (CSP):baliabideak 'self'-ra mugatzen ditu, `img data:`; 'unsafe-inline' ere sartzen ditu, gaur egungo handler inline-ekin bateragarria delako.
- X-Frame-Options: SAMEORIGIN eta frame-ancestors 'self' CSPn zuzendu, clickjacking-etik babesten.
- X-Content-Type-Options: nosniff ipini MIME sniffing ekiditzeko
- Referrer-Policy: strict-origin-when-cross-origin.
- Permissions-Policy: edukiera mugatuekin.
- Strict-Transport-Security (HSTS): konexioa ssegurua bada.
- Server goiburua normalizatzen du eta X-Powered-By ezabatzen du ahal bada

index.php eta logout.php eguneratu egin dira ere, saioa banaka hasi ordez db.php sartzeko, horrela, ibilbide guztiek goiburu eta saio-konfigurazio berak erabiltzen dituzte.

```
require_once __DIR__ . '/db.php';
```

6 Osagai kalteberak eta zaharkituak (A06:2021)

Azterketak osagaien bertsioen kudeaketan (bertsio zehaztugabeak, eguneratu gabeko softwarea) eta informazio-ihesen arloan (HTTP "X-Powered-By" goiburua) ahultasunak agerian utzi zituen, eta horiek ustiapen-arriskua handitzen zuten. Zuzenketa hauek aplikatu dira: Osagai guztiak bertsio zehatz eta egonkorretara eguneratu dira, eta etengabeko monitorizazio-prozesua ezarri da software guztia (liburutegiak, zerbitzariak) beti eguneratuta dagoela ziurtatzeko. Gainera, "X-Powered-By" HTTP goiburua ezabatu dugu, erasotzaileei gure sistema fingerprinting bidez identifikatzea eta ahultasun ezagunak ustiatzea zailtzeko. Horrela, aplikazioaren eraso-azalera murriztu da eta segurtasuna indartu da. `Dockerfile`-an hurrengoak ipini dira:

```
RUN echo "expose_php=Off" > /usr/local/etc/php/conf.d/security.ini \
 && echo "display_errors=Off" >> /usr/local/etc/php/conf.d/security.ini \
 && echo "log_errors=On" >> /usr/local/etc/php/conf.d/security.ini

RUN sed -i 's/^ServerTokens .*/ServerTokens Prod/'
    ↪ /etc/apache2/conf-available/security.conf || true \
 && sed -i 's/^ServerSignature .*/ServerSignature Off/'
     ↪ /etc/apache2/conf-available/security.conf || true
RUN printf '%s\n' '<IfModule mod_headers.c>' \
                ' Header always unset X-Powered-By' \
                ' Header always unset X-AspNet-Version' \
                ' Header always unset X-AspNetMvc-Version' \
                '</IfModule>' >
                    ↪ /etc/apache2/conf-available/remove-powered-by.conf
                    ↪ \
 && a2enconf remove-powered-by

COPY src/ /var/www/html/

RUN chown -R www-data:www-data /var/www/html \
 && find /var/www/html -type f -exec chmod 640 {} \; \
 && find /var/www/html -type d -exec chmod 750 {} \;
```

Azkenik, `db.php`-n, hauek aldatu ditugu aurkeztutako ahultasunak zuzentzeko:

```
    header('Content-Security-Policy: ' . implode('; ', $csp));
    header('X-Frame-Options: SAMEORIGIN');
    header('X-Content-Type-Options: nosniff');
    header('Referrer-Policy: strict-origin-when-cross-origin');
    header("Permissions-Policy: geolocation=(), camera=(),
        ↪ microphone=()");
    header_remove('X-Powered-By');
    header('Server: SecureApp');
```

Ahulezi hauek zuzendu ostean, hurrengo mezua pantailaratu zuen ZAP-ek: Cabecera Content Security Policy (CSP) no configurada url: http://localhost:81/sitemap.xml. Errorea `.htaccess`-n aplikatzen ari dela sitemap.xml-n X-Debug-Htaccess. `.htaccess`etik CSPa bortxatu egingo dugu:

```
  <Files "sitemap.xml">
  Header always unset Content-Security-Policy
  Header always set Content-Security-Policy "default-src 'self';
      ↪ script-src 'self'; style-src 'self'; img-src 'self' data:;
      ↪ font-src 'self' data:; connect-src 'self'; object-src 'none';
      ↪ base-uri 'self'; form-action 'self'; frame-ancestors 'self'"
</Files>
```

7 Erroreak

7.1 Adierazpen erregularrak

Warning bat agertu zen adierazpen erregularrean `\uXXXX` motako ihesak erabiltzen ari ginelako. PCRE2k (PHPk erabilitako liburutegia) ez ditu notazio horiek jasaten `\u` patroien barruan. Horregatik huts egiten du patroiaren konpilatze exekuzioan. Errore hau konpontzeko `registrer.php`-n `fullname` baieztapena aldatu behar da:

```
    if (!preg_match('/^[\p{L}\s]+$/u', $fullname)) $errors[] = 'Izen
        ↪ eta abizenak textua bakarrik.';
```

7.2 "Commands out of sync" login.php-n

MySQLi-k "Commands out of sync; you can't run this command now in /var/www/html/login.php:20 Stack trace: #0 /var/www/html/login.php(20): mysqli-¿prepare('UPDATE users SE...') #1 main thrown in /var/www/html/login.php on line 20"imprimatzen zuen. Errore hau konpontzeko, gorde behar da SELECT emaitza konexio berean beste statement-ak exekutatu aurretik:

```
    $stmt->store_result(); //
          if ($stmt->num_rows === 1) {
              $stmt->bind_result($u, $hash, $is_admin);
              $stmt->fetch();
              $stmt->close();
```

7.3 Ignoring session_start()

Aurreko arazoa konpondu ostean errore hau agertu zen: Notice: session_start(): Ignoring session_start() because a session is already active in /var/www/html/-login.php on line 3. Errorea pantaiaratzen zen gure scriptak detektatzen zuelako sesioa hasita zegoela, `session.auto_start` automatikoki aktibatzen zelako `pho.ini`-n. Konponbidetzat,`session_start()` soilik exekutatuko da sesioa ez badago zabalik:

```
   if (function_exists('session_status')) {
   if (session_status() === PHP_SESSION_NONE) {
       session_start();
   }
} else {
   if (session_id() === '') {
       session_start();
   }
}
```

7.4 session_set_cookie_params()-n errorea

Saioaren inguruko ariketaren bat gauzatzen saiatzen garenean hurrengo errorea agertzen da, Injekzioko (A03:2021) zuzenketak egin ostean: Session cookie parameters cannot be changed when a session is active in /var/www/html/db.php on line 41. Errorea gertatzen da `session_set_cookie_params()` deitzen ari delako saioa hasi ondoren (`session_start()` exekutatu ondoren). PHPk ez du uzten saio-cookiearen parametroak aldatzen saioa aktibo dagoenean.

Hau ekiditzeko, `db.php`-n, `session_status()` jarri behar da `session_set_cookie_params()` eta `session_start()` baino lehen.

8 Beste batzuk

8.1 Hasierara botoia

Orrialde guztietan "Hasierara" botoia, hasierako orrialdera joateko botoia `index.php`, ipini dugu.

```
<a href="/" class="back-btn">Hasierara</a>
```