Lovejoy Antiques and AWS VPC

Contents

Important Information 3
Task 0 – Self-reflection 4
Task 1 – User registration 5
Registration features and code screenshots 5
Database Tables 10
Why do you think it is secure? 12
Task 2 – Develop a secure login feature 13
Login features and code screenshots 13
Why do you think it is secure? 18
Task 3 – Implement password strength and password recovery 19
Password entropy, recovery code screenshots 19
Task 4 – Implement a “Evaluation Request” web page 25
Request Evaluation features and code screenshots 25
Why do you think it is secure? 28
Task 5 – Request Listing Page 29
Request Listing features and code screenshots 29
Why do you think it is secure? 31
Task 6 – AWS Virtual Private Cloud settings screenshots 31
Instances Running 31
Security groups rules 32
Routing tables 33
All subnets 33
Resource Map 35

Important Information

Code file location - https://1drv.ms/f/s!ApPLUhOl36G5gZ4KjSjTaYlWsojq7w?e=CdD9IS

Panopto recording - https://sussex.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=20b5765a-297d-4040-a8de-b23c0151ae38

SQL Database setup file - https://1drv.ms/u/s!ApPLUhOl36G5gaEOXFbaDLIGbRrluA?e=SNw88R

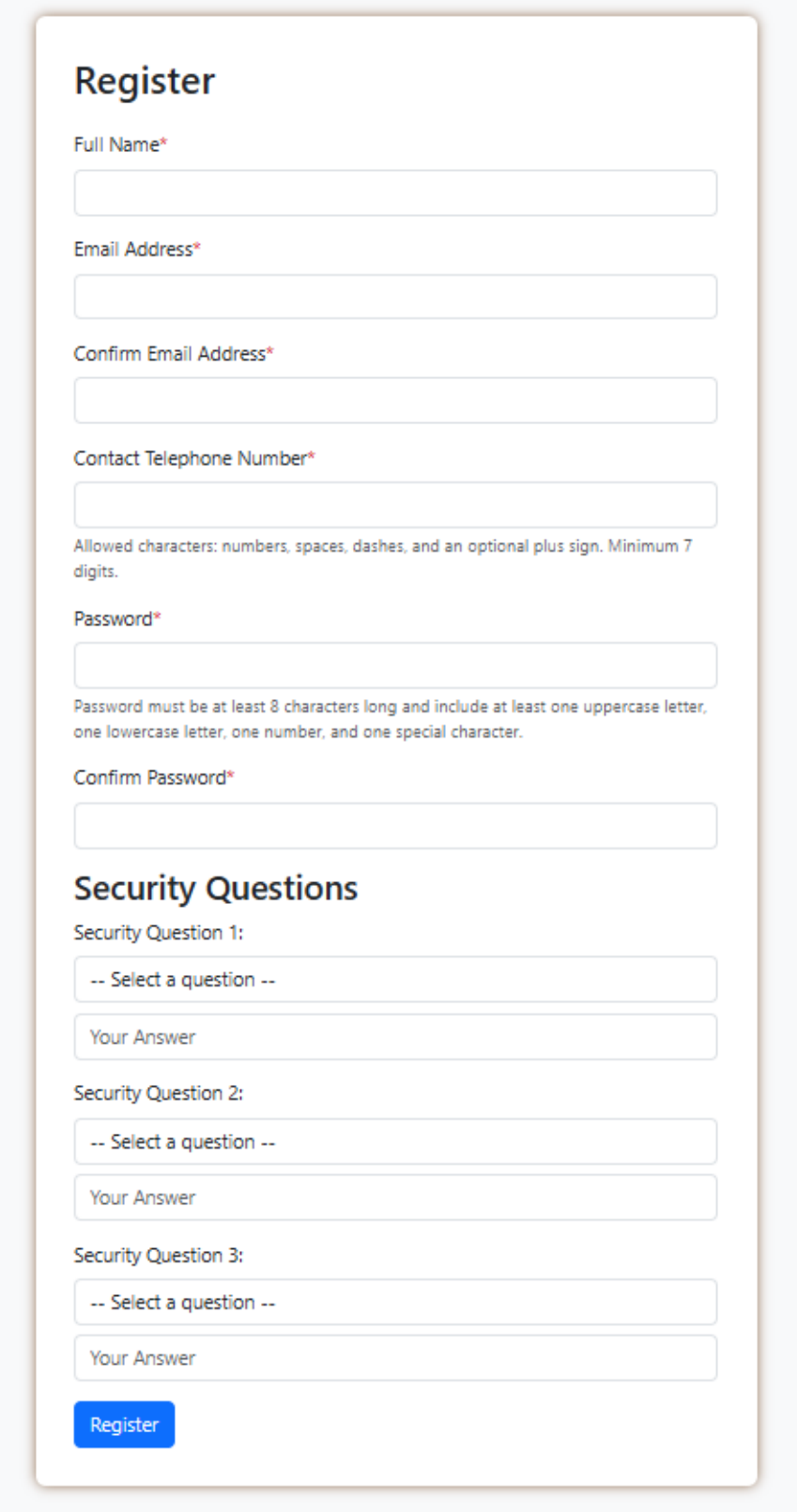
Task 0 – Self-reflection

Marking criteria	**Sub criteria**	**Tick/cross**	**Marks** (from the main marking grid, assign fair marks to yourself)
Password policy	Password entropy	X	13
	Security questions	X	
	Password recovery	X	
Vulnerabilites	SQL injection,	X	12
	XSS,	X	
	CSRF,	X	
	File Upload and	X	
	any other obvious vulnerability.	X	
Authentication/Encrypted storage	User registration, User login	X	9
	Email verification for registration,	X	
	2 factor authentications (PIN and or email)	X	
	Encrypted storage	X	
Obfuscation/Common attacks	Brute force attack – Number of attempts	X	8
	Botnet attack – Captcha	X	
	Dictionary attack/Rainbow table attack	X	
Features of web application	Database design	X	30
	User registration	X	
	User login	X	
	Forgot password	X	
	Evaluation	X	
	List evaluation	X	
VPC	Evidence provided	X	10
Video	All the marking criteria covered	X	6
Self-reflection	This marking grid fill out properly	X	4
			Total marks = 92

Task 1 – User registration

Registration features and code screenshots

User registration form:



- Within this form the user needs to enter in all of the credentials

Protection against CSRF and SQL Injection:

```
<!-- Registration Form -->
<form action="register.php" method="POST" novalidate>
    <!-- CSRF Token -->
    <input type="hidden" name="csrf_token" value="<?php echo escape(html: generateCsrfToken()); ?>">
```

```
/**
 * Verify CSRF token from form submission
 */
function verifyCsrfToken($token): bool {
    return isset($_SESSION['csrf_token']) && hash_equals(known_string: $_SESSION['csrf_token'], user_string: $token);
}
```

```
// CSRF token validation
if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
    $errors[] = "Invalid CSRF token.";
    return ['errors' => $errors, 'success' => $success];
}

// Retrieve and sanitize inputs
$email = trim(string: $_POST['email'] ?? '');
$confirm_email = trim(string: $_POST['confirm_email'] ?? '');
$password = $_POST['password'] ?? '';
$confirm_password = $_POST['confirm_password'] ?? '';
$name = trim(string: $_POST['name'] ?? '');
$phone = trim(string: $_POST['phone'] ?? '');

// Retrieve and sanitize security questions and answers
$security_question_1 = intval(value: $_POST['security_question_1'] ?? 0);
$security_answer_1 = strtolower(string: trim(string: $_POST['security_answer_1'] ?? '')); // Lowercased
$security_question_2 = intval(value: $_POST['security_question_2'] ?? 0);
```

```
/**
 * Generate CSRF token and store it in session
 */
function generateCsrfToken(): mixed|string {
    if (empty($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(string: random_bytes(length: 32));
    }
    return $_SESSION['csrf_token'];
}
```

Same origin policy and HTTPS

```
$CSP .= "X-Frame-Options: SAMEORIGIN ";
$CSP .= "Referrer-Policy: same-origin ";
$CSP .= "Strict-Transport-Security: max-age=31536000; includeSubDomains; preload ";
$CSP .= "Access-Control-Allow-Origin: localhost/lovejoy-antiques ";
$CSP .= "Access-Control-Allow-Methods: GET, POST ";
$CSP .= "Access-Control-Allow-Headers: Content-Type, Authorization ";

// Set the CSP header
header(header: "Content-Security-Policy: $CSP");
```

Session fixation

```
// Regenerate session ID to prevent session fixation
if (!isset($_SESSION['initiated'])) {
    session_regenerate_id(delete_old_session: true);
    $_SESSION['initiated'] = true;
}
```

Email

```php
// Validate email
if (empty($email)) {
    $errors[] = "Email is required.";
} elseif (!filter_var(value: $email, filter: FILTER_VALIDATE_EMAIL)) {
    $errors[] = "Invalid email format.";
} elseif ($email !== $confirm_email) {
    $errors[] = "Emails do not match.";
}
```

```php
// Check if email already exists
if (emailExists(pdo: $pdo, email: $email)) {
    $errors[] = "An account with this email already exists.";
    return ['errors' => $errors, 'success' => $success];
}
```

Password (1st image checks against 10 million common passwords)

```php
/**
 * Check the password is not in the common password list
 */
function isCommonPassword(string $password, string $file_path): bool {
    $handle = fopen(filename: $file_path, mode: 'r');
    if (!$handle) {
        die("Failed to open weak password file.");
    }

    while (($line = fgets(stream: $handle)) !== false) {
        if (trim(string: $line) === $password) {
            fclose(stream: $handle); // Close the file before returning
            return true;
        }
    }

    fclose(stream: $handle); // Ensure the file is closed
    return false;
}
```

```php
function validatePassword($password): array {
    $errors = [];

    // Load the weak password list
    $weak_passwords_file = __DIR__ . '\common_passwords\10-million-passwords.txt';
    if (!file_exists(filename: $weak_passwords_file)) {
        $errors[] = "File not found at for common passwords.";
        return $errors;
    }

    $isWeakPassword = isCommonPassword(password: $password, file_path: $weak_passwords_file);

    // Check if the password is provided
    if (empty($password)) {
        $errors[] = "New password is required.";
    } elseif (strlen(string: $password) < 8) {
        $errors[] = "Password must be at least 8 characters long.";
    }

    // Check complexity rules
    if (!preg_match(pattern: '/[A-Z]/', subject: $password)) {
        $errors[] = "Password must contain at least one uppercase letter.";
    }
    if (!preg_match(pattern: '/[a-z]/', subject: $password)) {
        $errors[] = "Password must contain at least one lowercase letter.";
    }
    if (!preg_match(pattern: '/[0-9]/', subject: $password)) {
        $errors[] = "Password must contain at least one number.";
    }
    if (!preg_match(pattern: '/[\W]/', subject: $password)) { // \W matches any non-word character (special characters)
        $errors[] = "Password must contain at least one special character.";
    }

    // Check if the password is too common
    if ($isWeakPassword) {
        $errors[] = "Password is too common, try another one.";
    }
```

```php
/**
 * Hash data using BCRYPT
 */
function hashData($data): string {
    return password_hash(password: $data, algo: PASSWORD_BCRYPT);
}
```

Other Validations

```php
// Validate name
if (empty($name)) {
    $errors[] = "Name is required.";
} elseif (strlen(string: $name) > 255) {
    $errors[] = "Name must not exceed 255 characters.";
}

// Validate phone
if (empty($phone)) {
    $errors[] = "Contact telephone number is required.";
} elseif (!preg_match(pattern: '/^\+?[0-9\s\-]{7,20}$/', subject: $phone)) {
    $errors[] = "Invalid telephone number format.";
}

// Validate security questions
if ($security_question_1 === 0 || $security_question_2 === 0 || $security_question_3 === 0) {
    $errors[] = "All three security questions must be selected.";
} else {
    // Ensure all selected questions are distinct
    if ($security_question_1 === $security_question_2 ||
        $security_question_1 === $security_question_3 ||
        $security_question_2 === $security_question_3) {
        $errors[] = "Security questions must be distinct.";
    }
}

// Validate security answers
if (empty($security_answer_1)) {
    $errors[] = "Answer to Security Question 1 is required.";
} elseif (strlen(string: $security_answer_1) > 255) {
    $errors[] = "Security Answer 1 must not exceed 255 characters.";
}

if (empty($security_answer_2)) {
    $errors[] = "Answer to Security Question 2 is required.";
} elseif (strlen(string: $security_answer_2) > 255) {
    $errors[] = "Security Answer 2 must not exceed 255 characters.";
}
```

XSS Protection (a sample of all the input forms)

```html
<!-- Name Field -->
<div class="mb-3">
    <label for="name" class="form-label">Full Name<span class="text-danger">*</span></label>
    <input type="text" class="form-control" id="name" name="name" required maxlength="255"
        value="<?php echo isset($_POST['name']) ? escape(html: $_POST['name']) : ''; ?>">
</div>

<!-- Email Field -->
<div class="mb-3">
    <label for="email" class="form-label">Email Address<span class="text-danger">*</span></label>
    <input type="email" class="form-control" id="email" name="email" required
        value="<?php echo isset($_POST['email']) ? escape(html: $_POST['email']) : ''; ?>">
</div>
```

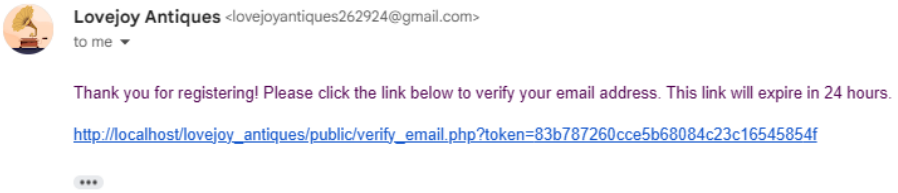
```php
/**
 * Escape output to prevent XSS
 */
function escape($html): string {
    return htmlspecialchars(string: $html, flags: ENT_QUOTES | ENT_SUBSTITUTE, encoding: "UTF-8");
}
```

Prepared statements

```php
// Begin a transaction
$pdo->beginTransaction();

// Insert the user into the users table
$stmt = $pdo->prepare("
    INSERT INTO users (name, email, password, phone)
    VALUES (:name, :email, :password, :phone)
");
$stmt->execute([
    ':name' => $name,
    ':email' => $email,
    ':password' => $hashed_password,
    ':phone' => $phone
]);
```

Verification Email (sent using PHPMailer)

```php
try {
    // Server settings
    $mail->isSMTP();
    $mail->Host       = 'smtp.gmail.com';
    $mail->SMTPAuth   = true;
    $mail->Username   = 'lovejoyantiques262924';
    $mail->Password   = 'ehfi dtpo fucz jmkl';
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port       = 587;
```



Token for the email

```php
// Generate a secure random token
$token = bin2hex(string: random_bytes(length: $token_length));

// Calculate expiration time
$expires_at = date(format: 'Y-m-d H:i:s', timestamp: strtotime(datetime: $validity_period));
```

Checks the token

```php
// Sanitize the token
$token = trim(string: $token);

if (empty($token)) {
    $errors[] = "Invalid verification token.";
    return ['success' => $success, 'errors' => $errors];
}

try {
    // Retrieve the token record
    $stmt = $pdo->prepare("
        SELECT user_id, expires_at
        FROM tokens
        WHERE token = :token AND type = 'verification'
        LIMIT 1
    ");
    $stmt->execute([':token' => $token]);
    $token_record = $stmt->fetch(PDO::FETCH_ASSOC);
```

If token correct

```php
if ($current_time <= $token_expiry) {
    // Update the user's verification status
    $update_stmt = $pdo->prepare("
        UPDATE users
        SET is_verified = 1
        WHERE id = :user_id
    ");
    $update_stmt->execute([':user_id' => $token_record['user_id']]);

    // Delete the token to prevent reuse
    $delete_stmt = $pdo->prepare("
        DELETE FROM tokens
        WHERE token = :token
    ");
    $delete_stmt->execute([':token' => $token]);

    $success = "Your email has been verified successfully! You can now <a href='login.php'>log in</a>.";
```

Resending verification email (checks if the user is locked from sending more verification emails and amount of attempts)

```php
// Update resend attempts in user_attempts table
$stmt = $pdo->prepare("
    INSERT INTO user_attempts (user_id, action_type, last_attempt, attempts, lock_until)
    VALUES (:user_id, :action_type, :current_time_insert, :attempts_insert, NULL)
    ON DUPLICATE KEY UPDATE
        last_attempt = :current_time_update,
        attempts = :attempts_update,
        lock_until = NULL
");
```

```php
// Rate limiting parameters
$current_time = new DateTime();
$last_attempt = $user['last_attempt'] ? new DateTime(datetime: $user['last_attempt']) : null;
$lock_until = $user['lock_until'] ? new DateTime(datetime: $user['lock_until']) : null;
$attempts = $user['attempts'] ?? 0;

// Check if user is currently locked out
if ($lock_until && $current_time < $lock_until) {
    $remaining = $current_time->diff(targetObject: $lock_until);
    $hours = $remaining->h + ($remaining->days * 24);
    $minutes = $remaining->i;
    $errors[] = "You have reached the maximum number of {$action_label} attempts. Please try again after {$hours} hours and {$minutes} minutes.";
} else {
    // Check if the last attempt was within the rate limiting window (24 hours)
    $hours_passed = 0;
    if ($last_attempt) {
        $diff = $current_time->diff(targetObject: $last_attempt);
        $hours_passed = ($diff->days * 24) + $diff->h + (($diff->i > 0 || $diff->s > 0) ? 1 : 0);
    }

    // Determine if within rate limiting window
    if ($hours_passed < 24) {
        if ($attempts >= 3) {
            // Lock the user out for 24 hours
            $errors[] = "You have reached the maximum number of {$action_label} attempts. Please try again after 24 hours.";
            $stmt = $pdo->prepare("
                UPDATE user_attempts
                SET lock_until = :lock_until
                WHERE user_id = :user_id AND action_type = :action_type
            ");
```

Database Tables

Users

←T→				id	name	email	password	phone	is_admin	registered_at	is_verified
Edit	Copy	Delete		2	Admin	admin_lovejoy@gmail.com	$2y$10$ImGnvIneaWio9LHNArp7uuQjQJSKehFXX9kgVMCVoa8...	07712457812	1	2024-11-13 14:33:49	1
Edit	Copy	Delete		3	Bill Nye	billnye123@gmail.com	$2y$10$SjvObCRtBHo/pu9VZ5DGYelitydFnMSUMVYb96hFocl...	07748165974	0	2024-11-13 22:38:52	1
Edit	Copy	Delete		13	Owen	owenjgibson@gmail.com	$2y$10$1gs5XRp0Jf9C/O1M2WVyv.kWL4usknpIDGuO8zSg9WC...	07712345678	0	2024-12-04 15:32:28	1
Edit	Copy	Delete		15	Owen	owenmancity@gmail.com	$2y$10$kTxEc9TGc1zD4aE1liCnNO2Q.OP7rVhk86VYh0NmAvd...	03721032713	0	2024-12-04 16:37:59	0

Tokens

				id	user_id	token	type	expires_at	created_at
Edit	Copy	Delete		147	13	1de63b7f91970c1bdfe699cff01274947	password_reset	2024-12-04 18:51:33	2024-12-04 16:51:33
Edit	Copy	Delete		148	16	1b21fbdb2d2e12232e655ad15918d0e	verification	2024-12-06 10:41:42	2024-12-05 09:41:42

User Attempts

				id	user_id	ip_addre	action_type	attempts	last_attempt	lock_until
Edit	Copy	Delete		92	13	NULL	verification	3	2024-12-04 16:33:07	2024-12-05 16:33:10
Edit	Copy	Delete		102	15	NULL	verification	2	2024-12-04 17:48:19	NULL
Edit	Copy	Delete		103	13	NULL	password_reset	3	2024-12-04 17:51:33	NULL
Edit	Copy	Delete		104	15	NULL	password_reset	2	2024-12-04 17:40:30	NULL
Edit	Copy	Delete		106	15	NULL	login	1	2024-12-04 16:48:07	NULL

User Security Answers

				user_id	security_question_id	security_answer
Edit	Copy	Delete		13	4	$2y$10$efNsrPUDbfrOGzhV.dYr1OAuoD8gmVQmCSZo69wDvtF...
Edit	Copy	Delete		13	6	$2y$10$cqZzjLT2cKjNC80gLBmvA.U6XwrL7ozohdETNsRbM5c...
Edit	Copy	Delete		13	8	$2y$10$Ao4bJ101ppQnNoe/x9xxOOgTIKecrGAfF0pDJI7ZXm....
Edit	Copy	Delete		15	4	$2y$10$l62qkHy/IoMilpGS6nx9..GSXGLp5jVPTsM/BwGSCb8...
Edit	Copy	Delete		15	6	$2y$10$FS.3I6Uxpq7wfKRRR2SzUOFby5ggnRRKJ6Gm755WY0a...
Edit	Copy	Delete		15	8	$2y$10$q3c/jEILvxvEfbuNO0PYwOc/TPXc/tZ6ojYL2icpKjF...
Edit	Copy	Delete		16	1	$2y$10$BGlWWYu80upUjrREy8cKFOkP2mjQI.0WNfuLGSQRtPo...
Edit	Copy	Delete		16	9	$2y$10$x/ezLmmvFC47gSndKP.QqOhsdYciMntgfKNOD9Yr9yc...
Edit	Copy	Delete		16	10	$2y$10$okmVBM1HtYE4cnPHDclABe.hc9fj5gysgYrW1yneEvc...

Security Questions

				id	question
Edit	Copy	Delete		1	What was the name of your first pet?
Edit	Copy	Delete		2	What is your mother's maiden name?
Edit	Copy	Delete		3	What was the name of your elementary school?
Edit	Copy	Delete		4	In what city were you born?
Edit	Copy	Delete		5	What is your favorite food?
Edit	Copy	Delete		6	What is the name of your favorite book?
Edit	Copy	Delete		7	What was your childhood nickname?
Edit	Copy	Delete		8	What is the name of the street you grew up on?
Edit	Copy	Delete		9	What was the make of your first car?
Edit	Copy	Delete		10	What is your favorite movie?

User 2FA

				id	user_id	code	expires_at	attempts	last_resend	resend_count	lock_until
Edit	Copy	Delete		135	13	$2y$10$MHa3wrh09U5AAQ4NULkaA	2024-12-04 18:01:48	0	2024-12-04 16:51:50	1	NULL

Evaluation Requests

				id	user_id	details	preferred_contact	photo	request_date
Edit	Copy	Delete		19	13	This is a test	email	photo_67507c5d1b98b5.29590021.png	2024-12-04 15:59:25
Edit	Copy	Delete		20	13	This is a test	phone	photo_67507f0ea555a6.72202268.png	2024-12-04 16:10:54

Why do you think it is secure?

Same-origin, HTTPS and session fixation

- Uses the same-origin policy and HTTPS, using a self-generated certificate for HTTPS
- Regenerates session if you go onto a different page (embed in the header is the init.php file)

Sanitising Inputs and CSRF

- The inputs are put through html special chars and UTF-8
- Once the input is put through, they are sanitised against SQL Injections
- Uses CSRF token on the registration form

Password Entropy

- Forces the user to enter in a password that has 8 or more characters, lower- and upper-case letter, number and special characters
- If the password passes this but is still a vulnerable password (Pa$$word1) then it will still reject this so that it more protected against dictionary attacks
- These passwords are hashed and salted so that if a user enters in the same password as someone else then the hash will be different, protecting against rainbow table attack

Email

- Checks if the email is already used by a user
- Have to verify your email before you can login, so users can’t use a fake email or create loads of accounts without real emails

Security Questions

- The user has to select unique security questions
- These security answers are hashed using the same hashing algorithm (BCRYPT) as the password

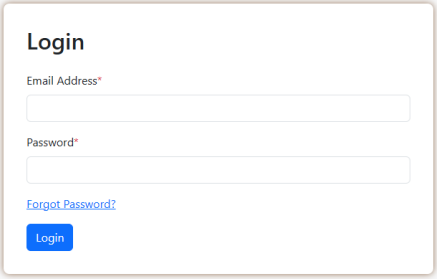
Verification Email

- Sends a verification email as soon as the user has registered an account
- The previous email is invalid if the user asks for another email
- This email link expires after 24 hours
- Can’t send more than 5 emails in 24 hours
- Sanitises the token so that the user can’t enter in a fake one for an SQL Injection

Task 2 – Develop a secure login feature

Login features and code screenshots

Login form



CRSF, XSS and SQL Injection protection

```php
function processLoginForm(PDO $pdo): array {
    $errors = [];

    // CSRF token validation
    if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
        $errors[] = "Invalid CSRF token.";
        return $errors;
    }

    // Retrieve and sanitize inputs
    $email = trim(string: $_POST['email'] ?? '');
    $password = $_POST['password'] ?? '';
```

```php
<!-- CSRF Token -->
<input type="hidden" name="csrf_token" value="<?php echo escape(html: generateCsrfToken()); ?>">

<!-- Email Field -->
<div class="mb-3">
    <label for="email" class="form-label">Email Address<span class="text-danger">*</span></label>
    <input type="email" class="form-control" id="email" name="email" required
        value="<?php echo isset($_POST['email']) ? escape(html: $_POST['email']) : ''; ?>">
</div>
```

Validate user inputs

```php
// Validate email
if (empty($email)) {
    $errors[] = "Email is required.";
} elseif (!filter_var(value: $email, filter: FILTER_VALIDATE_EMAIL)) {
    $errors[] = "Invalid email format.";
}

// Validate password
if (empty($password)) {
    $errors[] = "Password is required.";
}
```

```php
// Fetch user details if email is provided
if (!empty($email)) {
    $stmt = $pdo->prepare(query: "SELECT id, password, is_verified, is_admin, name FROM users WHERE email = :email LIMIT 1");
    $stmt->execute(params: [':email' => $email]);
    $user = $stmt->fetch(mode: PDO::FETCH_ASSOC);
    if ($user) {
        $user_id = $user['id'];
        $user_name = $user['name'];
    }
}
```

User attempts (Brute force)

```php
// Get client IP
$client_ip = $_SERVER['REMOTE_ADDR'];
```

```php
/**
 * Increment login attempts for a user or IP address
 */
function incrementLoginAttempts(PDO $pdo, ?int $user_id, string $ip_address, string $action_type = 'login'): void {
    try {
        if ($user_id !== null) {
            handleLoginAttempt(pdo: $pdo, type: 'user', identifier: $user_id, action_type: $action_type, thresho…5, "User ID {$user_id}");
        } else {
            handleLoginAttempt(pdo: $pdo, type: 'ip', identifier: $ip_address, action_type: $action_type, thresho…10, "IP Address {$ip_address}");
        }
    } catch (PDOException $e) {
        error_log(message: "Database Error in incrementLoginAttempts: " . $e->getMessage());
    }
}
```

```php
// Check if account or IP is locked
if ($attempt_record && $attempt_record['lock_until']) {
    $current_time = new DateTime();
    $lock_until = new DateTime(datetime: $attempt_record['lock_until']);

    if ($current_time < $lock_until) {
        $remaining = $lock_until->diff(targetObject: $current_time);
        $hours = $remaining->h;
        $minutes = $remaining->i;
        $seconds = $remaining->s;
        $errors[] = "Your account is locked due to multiple failed login attempts. Please try again after {$hours}h {$minutes}m {$seconds}s.";
        return $errors;
    }
}
```

Locking Account or IP for 30 minutes (Max attempts 7)

```php
if ($record) {
    // Increment attempts
    $new_attempts = $record['attempts'] + 1;
    $update_stmt = $pdo->prepare(query: "UPDATE user_attempts SET attempts = :attempts, last_attempt = NOW() WHERE id = :id");
    $update_stmt->execute(params: [
        ':attempts' => $new_attempts,
        ':id' => $record['id']
    ]);

    // Lock account or IP if attempts exceed threshold
    if ($new_attempts >= $threshold) {
        $lock_duration = '+30 minutes'; // Lock for 30 minutes
        $lock_until = date(format: 'Y-m-d H:i:s', timestamp: strtotime(datetime: $lock_duration));
        $lock_stmt = $pdo->prepare(query: "UPDATE user_attempts SET lock_until = :lock_until WHERE id = :id");
        $lock_stmt->execute(params: [
            ':lock_until' => $lock_until,
            ':id' => $record['id']
        ]);
```

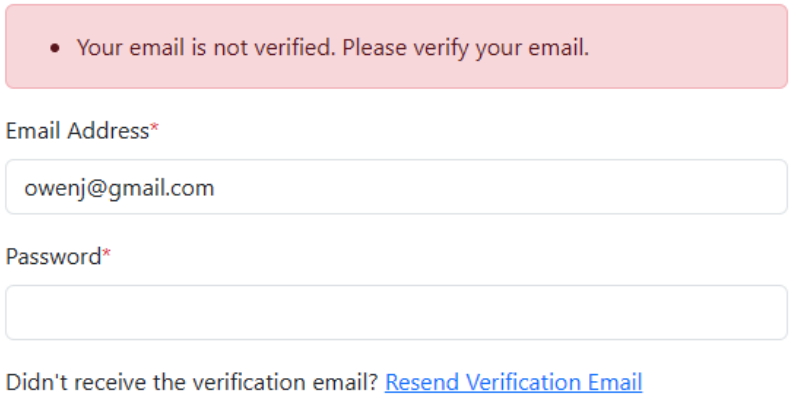
CAPTCHA to protect against Botnet attack (after 3 attempts)

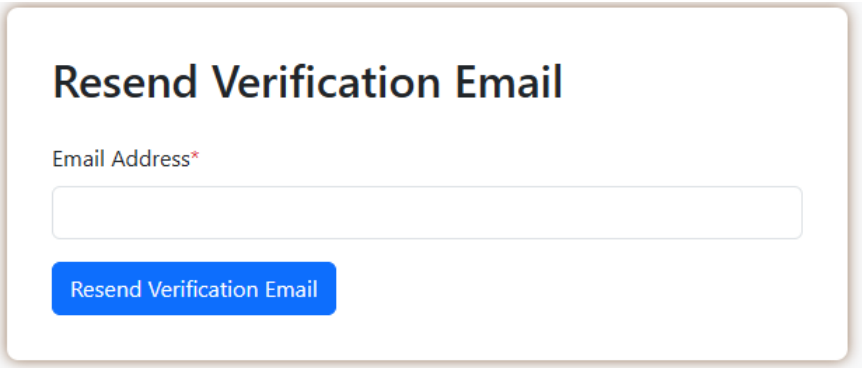
```php
// Determine if CAPTCHA should be shown
if ($attempt_record && $attempt_record['attempts'] >= 3 && $attempt_record['attempts'] < 7) {
    $show_captcha = true;
}
```

```php
// If CAPTCHA is required, verify it
if ($show_captcha) {
    if (empty($_POST['g-recaptcha-response'])) {
        $errors[] = "Please complete the CAPTCHA.";
    } else {
        // Verify CAPTCHA with Google
        $recaptcha_secret = RECAPTCHA_SECRET_KEY;
        $recaptcha_response = $_POST['g-recaptcha-response'];

        $verify_response = file_get_contents(filename: "https://www.google.com/recaptcha/api/siteverify?secret={$recaptcha_secret}&response={$recaptcha_response}");
        $response_data = json_decode(json: $verify_response);

        if (!$response_data->success) {
            $errors[] = "CAPTCHA verification failed. Please try again.";
        }
    }
}
```

If not verified disallows login and adds an attempt to the login attempt, uses the same policy of maximum 5 attempts and lock for 24 hours (shown in registration)

```php
// Proceed to credential verification
if ($user_id && isset($user['password'])) {
    if (password_verify(password: $password, hash: $user['password'])) {
        if ($user['is_verified'] != 1) {
            $errors[] = "Your email is not verified. Please verify your email.";
            incrementLoginAttempts(pdo: $pdo, user_id: $user_id, ip_address: $client_ip, action_type: 'login');
            return $errors;
```





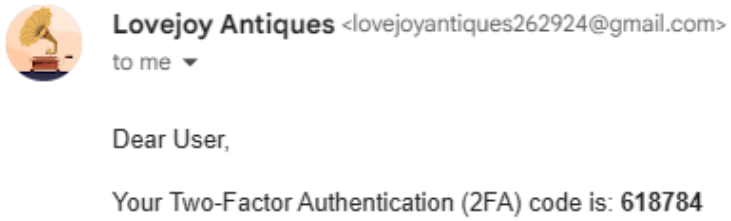
Can’t access the resend verification page if you are logged in

```php
if (isLoggedIn()) {
    header(header: 'Location: index.php');
    exit();
}
```

Authenticate User

```php
/**
 * Authenticate user with email and password
 */
function authenticateUser($pdo, $email, $password): mixed {
    try {
        $stmt = $pdo->prepare("SELECT id, name, email, password, is_admin, is_verified FROM users WHERE email = :email");
        $stmt->execute([':email' => $email]);
        $user = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($user && password_verify(password: $password, hash: $user['password'])) {
            return $user;
        }
    } catch (PDOException $e) {
        error_log(message: "Database Error: " . $e->getMessage());
    }
    return false;
}
```

Once logged in sends a 2FA 6-digit code to email, using the same PHPMailer settings as Task 1




```php
// Validate CSRF token
if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
    $errors[] = "Invalid CSRF token.";
    return $errors;
}

//Santise code
$entered_code = trim(string: $_POST['2fa_code'] ?? '');
```

```php
<!-- CSRF Token -->
<input type="hidden" name="csrf_token" value="<?php echo escape(html: generateCsrfToken()); ?>">

<!-- 2FA Code Field -->
<div class="mb-3">
    <label for="two_fa_code" class="form-label">Enter the 2FA Code sent to your email <span class="text-danger">*</span></label>
    <input
        type="text"
        class="form-control"
        id="two_fa_code"
        name="2fa_code"
        required
        value="<?php echo isset($_POST['2fa_code']) ? escape(html: $_POST['2fa_code']) : ''; ?>"
        pattern="\d{6}"
        maxlength="6">
    <div class="form-text">6-digit code.</div>
</div>
```

Stores the 2FA code hashed using BCRYPT and it expires within 10 minutes

```php
// Generate a random 6-digit code using a cryptographically secure method
$code = str_pad(string: random_int(min: 0, max: 999999), length: 6, pad_string: '0', pad_t…STR_PAD_LEFT);

$hash_code = password_hash(password: $code, algo: PASSWORD_BCRYPT);
// Set expiration time (e.g., 10 minutes from now)
$expires_at = date(format: 'Y-m-d H:i:s', timestamp: strtotime(datetime: '+10 minutes'));
```

```php
try {
    $resend_count = getResendCount(pdo: $pdo, user_id: $user_id);

    // Delete any existing 2FA codes for the user to ensure a single active code
    deleteExisting2FACodes(pdo: $pdo, user_id: $user_id);

    // Insert the new 2FA code into the database
    insert2FACode(pdo: $pdo, user_id: $user_id, code: $hash_code, expires_at: $expires_at, resend_coun…$resend_count);
```

Can resend the 2FA Code up to 5 times with a new code each time

```php
// Define limits
$max_resends = 5; // Maximum number of resends allowed
$resend_cooldown_minutes = 2; // Cooldown period in minutes

// Fetch the latest 2FA record for the user
$stmt = $pdo->prepare(query: "SELECT resend_count, last_resend FROM user_2fa WHERE user_id = :user_id ORDER BY id DESC LIMIT 1");
$stmt->execute(params: [':user_id' => $user_id]);
$record = $stmt->fetch(mode: PDO::FETCH_ASSOC);

if ($record) {
    $current_time = new DateTime();
    $last_resend = $record['last_resend'] ? new DateTime(datetime: $record['last_resend']) : null;

    // Check if the cooldown period has passed
    if ($last_resend) {
        $diff = $current_time->getTimestamp() - $last_resend->getTimestamp();
        if ($diff < ($resend_cooldown_minutes * 60)) {
            $remaining_seconds = ($resend_cooldown_minutes * 60) - $diff;
            $remaining_time = gmdate(format: "i:s", timestamp: $remaining_seconds);
            return [
                'can_resend' => false,
                'message' => "Please wait {$remaining_time} minutes before requesting another 2FA code."
            ];
```

Can try to enter in an incorrect 2FA Code up to 5 time before it locks (checks the hashes)

```php
// Check if the code matches and is not expired
if (password_verify(password: $code, hash: $record['code']) && new DateTime() <= new DateTime(datetime: $record['expires_at'])) {
    // Successful verification, delete the 2FA code
    delete2FACodeById(pdo: $pdo, id: $record['id']);

    return true;
} else {
    // Increment the attempt count
    increment2FAAttempts(pdo: $pdo, id: $record['id']);

    return "Invalid or expired 2FA code.";
}
```

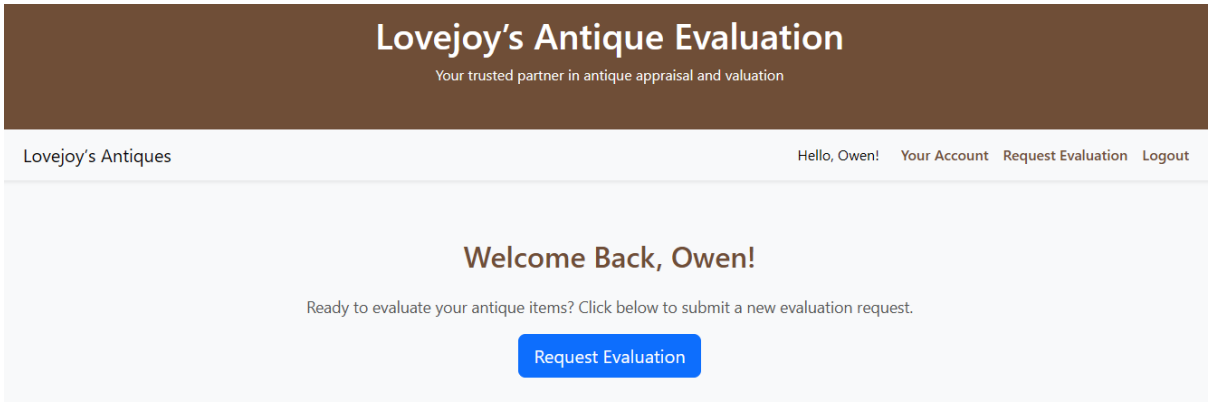
```php
/**
 * Increments the attempt count for a specific 2FA record.
 */
function increment2FAAttempts(PDO $pdo, int $id): void {
    $update_stmt = $pdo->prepare(query: "UPDATE user_2fa SET attempts = attempts + 1 WHERE id = :id");
    $update_stmt->execute(params: [':id' => $id]);
}
```

```php
/**
 * Locks the 2FA process for a user for a specified number of minutes.
 */
function lock2FA(PDO $pdo, int $user_id, int $minutes): void {
    $stmt = $pdo->prepare(query: "
        UPDATE user_2fa
        SET lock_until = DATE_ADD(NOW(), INTERVAL :minutes MINUTE)
        WHERE user_id = :user_id
    ");
    $stmt->execute(params: [
        ':minutes' => $minutes,
        ':user_id' => $user_id
    ]);
}
```

```php
// If a record exists and lock_until is set
if ($record && !empty($record['lock_until'])) {
    $current_time = new DateTime(datetime: 'now', timezone: new DateTimeZone(timezone: 'UTC'));
    $lock_until_time = new DateTime(datetime: $record['lock_until'], timezone: new DateTimeZone(timezone: 'UTC'));

    // Compare current time with lock_until
    if ($current_time > $lock_until_time) {
        // Lock period has expired, reset resend_count and lock_until
        reset2FALock(pdo: $pdo, user_id: $user_id);
        return 0;

    } else {
        // Lock is still active
        $remaining_interval = $current_time->diff(targetObject: $lock_until_time);
```

Once verified it will go the index page

```php
if ($user) {
    // Set session variables
    $_SESSION['user_id'] = $user_id;
    $_SESSION['user_name'] = escape(html: $user['name']);

    // Reset login attempts
    resetLoginAttempts(pdo: $pdo, user_id: $user_id);

    // Unset temporary session variables
    unset($_SESSION['2fa_user_id']);
    unset($_SESSION['temp_user_name']);

    // Regenerate session ID to prevent session fixation
    session_regenerate_id(delete_old_session: true);

    // Set redirect URL
    $redirect = 'index.php';
```



Can't access the 2FA form unless you are logged in

```php
checkAccess(requiredRole: 'user');
```

Forgot Password shown in Task 3

User 2FA Database shown in Task 1

Why do you think it is secure?

Same-origin, HTTPS and session fixation

- Same as in Task 1

Sanitising inputs and CSRF Token

- Uses XSS and sanitises on all inputs in the 2FA, email, password and forgot password
- Has a CSRF Token for all forms

Protection against Botnet and brute force (Login)

- Only allows up to 7 login attempts for an email and then locks out account
- Only allows up to 7 attempts for an IP
- Shows a CAPTCHA to complete after 3 attempts

2FA Code

- Only allows 5 2FA code attempts and then locks out
- Can only send up to 5 2FA Codes
- Previous code is invalid after new sent
- Code expires after 10 minutes
- The Code is hashed and salted using BCRYPT to be stored in the database

Verification

- Only allow verified users to be able to login

Forgot Password

- Allow user to send a reset password link to their email if forgotten password (with a limit of 5)

Task 3 – Implement password strength and password recovery

Password entropy, recovery code screenshots

- Password Entropy (8 characters, special character, capital letter, lower case letter and number)

```php
// Check if the password is provided
if (empty($password)) {
    $errors[] = "New password is required.";
} elseif (strlen(string: $password) < 8) {
    $errors[] = "Password must be at least 8 characters long.";
}

// Check complexity rules
if (!preg_match(pattern: '/[A-Z]/', subject: $password)) {
    $errors[] = "Password must contain at least one uppercase letter.";
}
if (!preg_match(pattern: '/[a-z]/', subject: $password)) {
    $errors[] = "Password must contain at least one lowercase letter.";
}
if (!preg_match(pattern: '/[0-9]/', subject: $password)) {
    $errors[] = "Password must contain at least one number.";
}
if (!preg_match(pattern: '/[\W]/', subject: $password)) { // \W matches any non-word character (special characters)
    $errors[] = "Password must contain at least one special character.";
}
```

- Checks against 10 million common passwords to protect against dictionary attack

```php
/**
 * Check the password is not in the common password list
 */
function isCommonPassword(string $password, string $file_path): bool {
    $handle = fopen(filename: $file_path, mode: 'r');
    if (!$handle) {
        die("Failed to open weak password file.");
    }

    while (($line = fgets(stream: $handle)) !== false) {
        if (trim(string: $line) === $password) {
            fclose(stream: $handle); // Close the file before returning
            return true;
        }
    }

    fclose(stream: $handle); // Ensure the file is closed
    return false;
}
```

```php
// Check if the password is too common
if ($isWeakPassword) {
    $errors[] = "Password is too common, try another one.";
}
```

Snippet of file, can be found in the scripts folder under 10 million passwords

```
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
123123
baseball
abc123
football
monkey
letmein
```

- Forgot password in login form

Check CSRF and sanitise input

```php
// CSRF token validation
if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
    $errors[] = "Invalid CSRF token.";
    return ['errors' => $errors, 'success' => $success];
}

// Retrieve and sanitize input
$email = trim(string: $_POST['email'] ?? '');
```

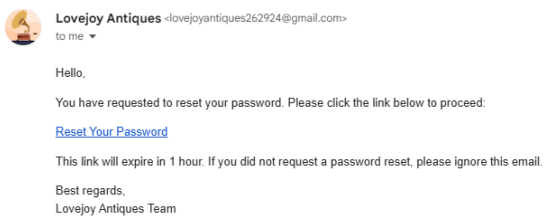
```html
<!-- CSRF Token -->
<input type="hidden" name="csrf_token" value="<?php echo escape(html: generateCsrfToken()); ?>">

<!-- Email Field -->
<div class="mb-3">
    <label for="email" class="form-label">Email Address<span class="text-danger">*</span></label>
    <input type="email" class="form-control" id="email" name="email" required
        value="<?php echo isset($_POST['email']) ? escape(html: $_POST['email']) : ''; ?>">
</div>
```

Shows that an email is sent even if the email doesn't exist, so that the user can't get database information based on if an email is sent

```php
$email_message_sent = "If an account with that email exists, a password reset link has been sent.";
```

```php
// For password reset, inform user that email has been sent regardless of account existence
if(empty($errors)){
    $success = $email_message_sent;
}
```

Send a link to the email entered in using the PHPMailer settings from Task 1 and rate limit so that it can only send a limit of 3 within 24 hours



Lovejoy Antiques <lovejoyantiques262924@gmail.com>
to me

Hello,

You have requested to reset your password. Please click the link below to proceed:

Reset Your Password

This link will expire in 1 hour. If you did not request a password reset, please ignore this email.

Best regards,
Lovejoy Antiques Team

```php
// Check if user is currently locked out
if ($lock_until && $current_time < $lock_until) {
    $remaining = $current_time->diff(targetObject: $lock_until);
    $hours = $remaining->h + ($remaining->days * 24);
    $minutes = $remaining->i;
    $errors[] = "You have reached the maximum number of {$action_label} attempts. Please try again after {$hours} hours and {$minutes} minutes.";
} else {
    // Check if the last attempt was within the rate limiting window (24 hours)
    $hours_passed = 0;
    if ($last_attempt) {
        $diff = $current_time->diff(targetObject: $last_attempt);
        $hours_passed = ($diff->days * 24) + $diff->h + (($diff->i > 0 || $diff->s > 0) ? 1 : 0);
    }

    // Determine if within rate limiting window
    if ($hours_passed < 24) {
        if ($attempts >= 3) {
            // Lock the user out for 24 hours
            $errors[] = "You have reached the maximum number of {$action_label} attempts. Please try again after 24 hours.";
            $stmt = $pdo->prepare("
                UPDATE user_attempts
                SET lock_until = :lock_until
                WHERE user_id = :user_id AND action_type = :action_type
            ");
            $lock_until_time = (clone $current_time)->modify(modifier: '+24 hours')->format(format: 'Y-m-d H:i:s');
            $stmt->execute([
                ':lock_until' => $lock_until_time,
                ':user_id' => $user['id'],
                ':action_type' => $action_type
            ]);
        }
    }
```

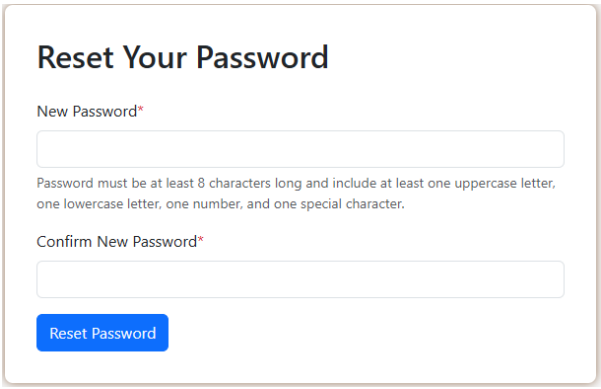
Much like the verification, the previous email expires once you request another email.

The token expires after 1 hour

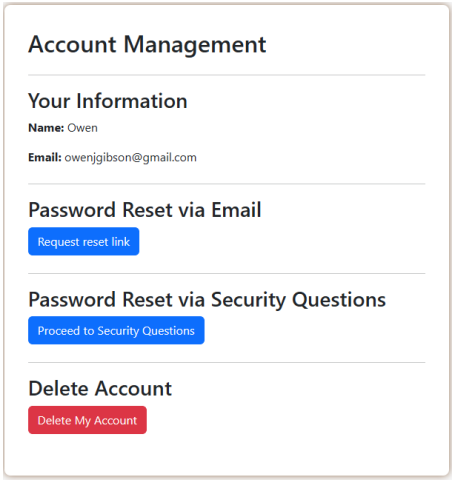
```php
$action_type = 'password_reset';
$action_label = 'password reset';
$email_message_sent = "If an account with that email exists, a password reset link has been sent.";
$email_send_function = 'sendPasswordResetEmail';
$token_type = 'password_reset';
$token_length = 32;
$token_expiry = '+1 hour';
```

```php
try {
    $stmt = $pdo->prepare(query: "
        SELECT user_id, expires_at
        FROM tokens
        WHERE token = :token AND type = 'password_reset'
    ");
    $stmt->execute(params: [':token' => $token]);
    $token_data = $stmt->fetch(mode: PDO::FETCH_ASSOC);

    if ($token_data) {
        $current_time = new DateTime();
        $expires_at = new DateTime(datetime: $token_data['expires_at']);

        if ($current_time > $expires_at) {
            $errors[] = "This password reset link has expired.";
        } else {
            $can_display_form = true;
            $_SESSION['reset_token'] = $token;
            $user_id = $token_data['user_id'];
        }
    } else {
        $errors[] = "Invalid password reset token.";
    }
```

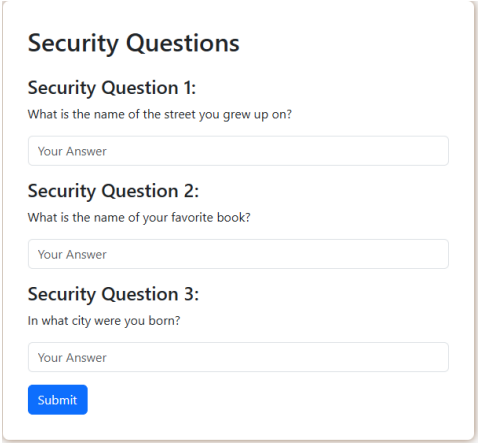
The password reset form, using the same password entropy and common password check as register



Once logged in you can reset your password using a email link (the same as in forgot password so if you are locked out from that you will be locked out from this one)



Additionally, you can reset your password using your Security questions



XSS, CSRF and SQL Injection protection

```
<!-- CSRF Token -->
<input type="hidden" name="csrf_token" value="<?php echo escape(html: generateCsrfToken()); ?>">

<div class="mb-3">
    <h4>Security Question 1:</h4>
    <p><?php echo escape(html: $security_questions[0]['question']); ?></p>
    <!-- Hidden input to store the question ID -->
    <input type="hidden" name="security_question_1" value="<?php echo escape(html: $security_questions[0]['id']); ?>">
    <!-- Input for the user's answer -->
    <input type="text" name="security_answer_1" class="form-control mt-2" placeholder="Your Answer" required
        value="<?php echo isset($_POST['security_answer_1']) ? escape(html: $_POST['security_answer_1']) : ''; ?>">
</div>
```

```php
// Validate CSRF token
if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
    $errors[] = "Invalid CSRF token.";
    return ['errors' => $errors, 'success' => $success];
}

// Fetch stored security answers
$stored_answers = fetchUserSecurityAnswers(pdo: $pdo, user_id: $user_id);

// Retrieve and sanitize user-provided answers
$provided_answers = [
    strtolower(string: trim(string: $_POST['security_answer_1'] ?? '')),
    strtolower(string: trim(string: $_POST['security_answer_2'] ?? '')),
    strtolower(string: trim(string: $_POST['security_answer_3'] ?? ''))
];
```

Checks each input isn't empty and that it matches the hash in the database

```php
// Validate that all answers are provided
foreach ($provided_answers as $index => $answer) {
    if (empty($answer)) {
        $errors[] = "Answer to Security Question " . ($index + 1) . " is required.";
    }
}

if(empty($errors)){
    // Verify each provided answer against the stored hashed answer
    foreach ($stored_answers as $index => $stored) {
        if (!password_verify(password: $provided_answers[$index], hash: $stored['hashed_answer'])) {
            $errors[] = "Incorrect answer to the security question/s.";
            return ['errors' => $errors, 'success' => $success];
        }
    }
}
```

If all correct then it will go to the reset password form with a tag of security_questions=1 tag, this is checked by against the $_session token['can_reset_password'] (all session tokens are reset if you go to a different page)

```php
if(empty($errors)){
    // Verify each provided answer against the stored hashed answer
    foreach ($stored_answers as $index => $stored) {
        if (!password_verify(password: $provided_answers[$index], hash: $stored['hashed_answer'])) {
            $errors[] = "Incorrect answer to the security question/s.";
            return ['errors' => $errors, 'success' => $success];
        }
    }
}

// All answers match
if(empty($errors)){
    $_SESSION['can_reset_password'] = true;
    $success = true;
}
```

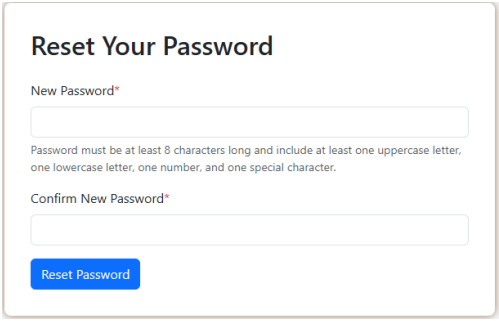
```php
// Validate security questions access
if (isset($_SESSION['can_reset_password']) && $_SESSION['can_reset_password'] === true) {
    $is_security_questions = true;
    $can_display_form = true;
} else {
    $errors[] = "Unauthorized access to password reset.";
}
```

Form validation using CSRF, sanitising, checking password validation and removing the token data

```php
// Validate CSRF token
if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
    $errors[] = "Invalid CSRF token.";
    return ['success' => $success, 'errors' => $errors];
}

// Retrieve and sanitize inputs
$token = $_POST['token'] ?? '';
$new_password = $_POST['new_password'] ?? '';
$confirm_new_password = $_POST['confirm_new_password'] ?? '';

// Determine reset method
if ($is_security_questions) {
    if (!isset($user_id)) {
        $errors[] = "User identification error.";
        return ['success' => $success, 'errors' => $errors];
    }
} elseif (!empty($token)) {
    // Token-based reset
} else {
    $errors[] = "Invalid password reset access method.";
    return ['success' => $success, 'errors' => $errors];
}
```

```php
if ($is_security_questions) {
    // Update password for authenticated user
    updateUserPassword(pdo: $pdo, user_id: $user_id, new_password: $new_password);
} else {
    // Token-based password reset
    $token_data = getTokenData(pdo: $pdo, token: $token, type: 'password_reset');

    if ($token_data) {
        if (isTokenExpired(expires_at: $token_data['expires_at'])) {
            $errors[] = "This password reset link has expired.";
            $pdo->rollBack();
            return ['success' => $success, 'errors' => $errors];
        }

        $user_id = $token_data['user_id'];
        updateUserPassword(pdo: $pdo, user_id: $user_id, new_password: $new_password);

        // Delete the used token to prevent reuse
        deleteToken(pdo: $pdo, token: $token, type: 'password_reset');
```

Password reset form (after security questions) uses the same password entropy and common password check



Can't access these forms unless you are logged in or through a valid password reset link

```php
//Check if user is logged in
checkAccess(requiredRole: 'user');
```

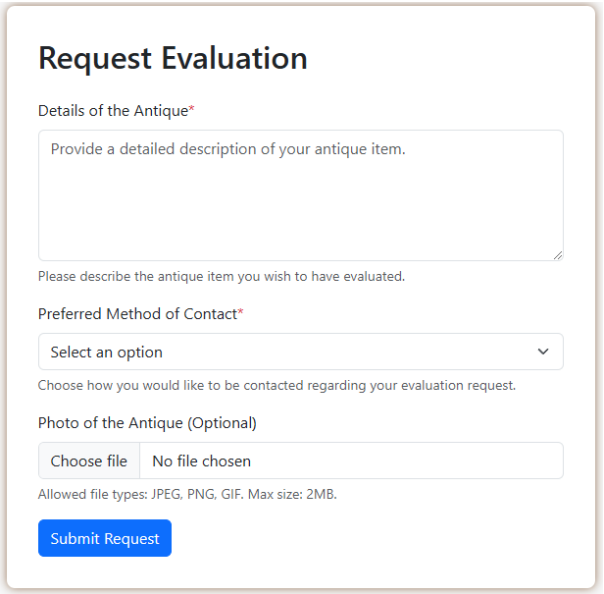
All forms use same-origin, HTTPS and session fixation as shown in task 1.

The user attempts table can be found in task 1.

Task 4 – Implement a “Evaluation Request” web page

Request Evaluation features and code screenshots

Request evaluation form



Users get a choice between email or phone as a method of contact

XSS, SQL Injection and CSRF validation

```php
<!-- CSRF Token -->
<input type="hidden" name="csrf_token" value="<?php echo escape(html: generateCsrfToken()); ?>">

<!-- Details of the Antique -->
<div class="mb-3">
    <label for="details" class="form-label">Details of the Antique<span class="text-danger">*</span></label>
    <textarea class="form-control" id="details" name="details" rows="5" required maxlength="1000"
        placeholder="Provide a detailed description of your antique item."><?php echo isset($_POST['details']) ? $_POST['details'] : ''; ?></textarea>
    <div class="form-text">Please describe the antique item you wish to have evaluated.</div>
</div>
```

```php
// CSRF token validation
if (!isset($_POST['csrf_token']) || !verifyCsrfToken(token: $_POST['csrf_token'])) {
    $errors[] = "Invalid CSRF token.";
}

// Retrieve and sanitize inputs
$details = trim(string: $_POST['details'] ?? '');
$preferred_contact = $_POST['preferred_contact'] ?? '';
```

Validate details and preferred contact

```php
// Validate 'details'
if (empty($details)) {
    $errors[] = "Details of the antique are required.";
} elseif (strlen(string: $details) > 1000) {
    $errors[] = "Details must not exceed 1000 characters.";
}

// Validate 'preferred_contact'
if (empty($preferred_contact)) {
    $errors[] = "Preferred method of contact is required.";
} elseif (!in_array(needle: $preferred_contact, haystack: ALLOWED_CONTACT_METHODS)) {
    $errors[] = "Invalid preferred contact method.";
}
```

Validate file upload

```php
function handleFileUpload($file, &$errors): string|null {
    $photo_filename = null;

    // Check if a file was uploaded
    if ($file['error'] === UPLOAD_ERR_NO_FILE) {
        // No file uploaded
        return $photo_filename;
    }

    //Check for upload errors
    if (!checkUploadError(file: $file, errors: &$errors)) {
        return $photo_filename;
    }

    // Validate MIME type
    if (!validateMimeType(filePath: $file['tmp_name'], errors: &$errors)) {
        return $photo_filename;
    }

    //Validate file size
    if (!validateFileSize(fileSize: $file['size'], errors: &$errors)) {
        return $photo_filename;
    }

    //Validate image
    if (!validateImage(filePath: $file['tmp_name'], errors: &$errors)) {
        return $photo_filename;
    }

    //Read file content
    $file_content = readFileContent(filePath: $file['tmp_name'], errors: &$errors);
    if ($file_content === false) {
        return $photo_filename;
    }
```

Check MIME file

```php
/**
 * Validate MIME type using finfo
 */
function validateMimeType($filePath, &$errors): bool {
    $finfo = finfo_open(flags: FILEINFO_MIME_TYPE);
    if (!$finfo) {
        $errors[] = "Failed to open fileinfo.";
        return false;
    }

    $detected_type = finfo_file(finfo: $finfo, filename: $filePath);
    finfo_close(finfo: $finfo);

    if (!in_array(needle: $detected_type, haystack: ALLOWED_FILE_TYPES)) {
        $errors[] = "Invalid file type. Only JPG, PNG, and GIF are allowed.";
        return false;
    }

    return true;
}
```

Check valid image (JPEG, PNG ,GIF)

```php
/**
 * Validate image using getimagesize
 */
function validateImage($filePath, &$errors): bool {
    $image_info = getimagesize(filename: $filePath);
    if ($image_info === false) {
        $errors[] = "Uploaded file is not a valid image.";
        return false;
    }
    return true;
}
```

Check valid size

```php
/**
 * Validate file size
 */
function validateFileSize($fileSize, &$errors): bool {
    if ($fileSize > MAX_FILE_SIZE) {
        $errors[] = "File size exceeds the 2MB limit.";
        return false;
    }
    return true;
}
```

If passed, encrypt the photo using my encryption key in my config file

```php
define(constant_name: 'RECAPTCHA_SITE_KEY', value: '6Leu6oMqAAAAALCJy7q8XQwHFrmRFR5PQtCKofa3');
define(constant_name: 'RECAPTCHA_SECRET_KEY', value: '6Leu6oMqAAAAAKt1tqDcRna5ETihV9es-WksQC3e');
define(constant_name: 'ENCRYPTION_KEY', value: 'f049815dc7a59b9d47da9ab0a7d45c69facab4ecfee7a8bc851454a366b5a532');
define(constant_name: 'HOST', value: 'localhost');
define(constant_name: 'DB', value: 'lovejoy_antiques');
define(constant_name: 'USER', value: 'root');
define(constant_name: 'PASS', value: '');
```

```php
/**
 * Encrypt data using AES-256-CBC
 */
function encryptData($data): string {
    $iv_length = openssl_cipher_iv_length(cipher_algo: 'AES-256-CBC');
    $iv = openssl_random_pseudo_bytes(length: $iv_length);
    $encrypted = openssl_encrypt(data: $data, cipher_algo: 'AES-256-CBC', passphrase: ENCRYPTION_KEY, options: 0, iv:…$iv);
    // Store the IV with the encrypted data for decryption
    return base64_encode(string: $iv . $encrypted);
}
```

Generate a unique file name

```php
/**
 * Generate a unique filename
 */
function generateUniqueFilename($originalName): string {
    $ext = strtolower(string: pathinfo(path: $originalName, flags: PATHINFO_EXTENSION));
    return uniqid(prefix: 'photo_', more_entropy: true) . '.' . $ext;
}
```

Save the encrypted file to uploads

```php
/**
 * Save encrypted file to destination
 */
function saveEncryptedFile($encrypted_content, $destination, &$errors): bool {
    if (file_put_contents(filename: $destination, data: $encrypted_content) === false) {
        $errors[] = "Failed to save the encrypted file.";
        return false;
    }
    return true;
}
```

Set strict file permissions (0600 means gives the owner of a file full read and write access, while preventing other users from accessing the file)

```php
//Set file permissions
if (!setFilePermissions(filePath: $destination, errors: &$errors)) {
    // If setting permissions fails, delete the file for security
    unlink(filename: $destination);
    $photo_filename = null;
    return $photo_filename;
}
```

```php
/**
 * Set strict file permissions
 */
function setFilePermissions($filePath, &$errors): bool {
    if (!chmod(filename: $filePath, permissions: 0600)) {
        $errors[] = "Failed to set file permissions.";
        return false;
    }
    return true;
}
```

Inputting this into the SQL table

```php
function submitEvaluationRequest($pdo, $userId, $details, $preferred_contact, $photo_filename): bool|string {
    try {
        $stmt = $pdo->prepare("
            INSERT INTO evaluation_requests (user_id, details, preferred_contact, photo)
            VALUES (:user_id, :details, :preferred_contact, :photo)
        ");

        $stmt->execute([
            ':user_id' => $userId,
            ':details' => $details,
            ':preferred_contact' => $preferred_contact,
            ':photo' => $photo_filename
        ]);

        return $stmt->rowCount() ? true : "Failed to submit your request. Please try again.";
    } catch (PDOException $e) {
        error_log(message: "Database Insertion Error: " . $e->getMessage());
        return "An error occurred while submitting your request. Please try again later.";
    }
}
```

Can't access form unless logged in

```php
//Check if user is logged in
checkAccess(requiredRole: 'user');
```

```php
/**
 * Access control based on user role
 */
function checkAccess($requiredRole = 'user'): void {
    if (!isLoggedIn()) {
        header(header: 'Location: index.php');
        exit();
    }
    if ($requiredRole === 'admin' && !isAdmin()) {
        header(header: 'Location: index.php');
        exit();
    }
}
```

The evaluation_requests database can be found in task 1

Why do you think it is secure?

Same-origin, HTTPS and session fixation

- Same as in Task 1
- Can't access form unless logged in

Sanitising inputs and CSRF Token

- Uses XSS and sanitises on the details input
- Has a CSRF Token for all forms

Details

- Details have to be inputted and are sanitised
- Details have to be below 1000 characters

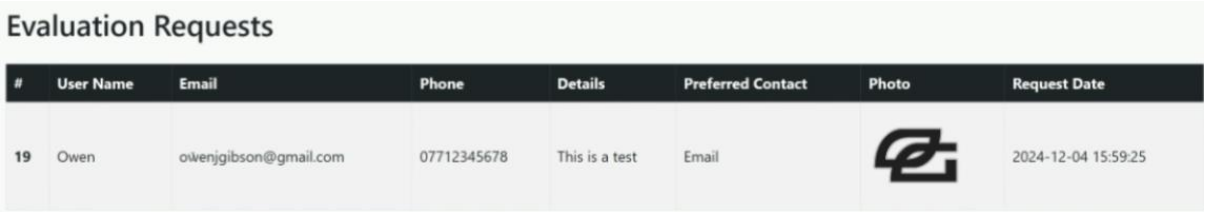
File Upload

- The photo gets check that is either a PNG, JPEG or a GIF
- The size of the image is below 2mb
- Check the image for MIME file so you can't spoof a file type
- Set the permissions for the file as chmod 600 which gives the owner of a file full read and write access, while preventing other users from accessing the file
- The file is encrypted in the storage and is only decrypted when an admin requests for the requests

Task 5 – Request Listing Page

Request Listing features and code screenshots

The evaluation page



Evaluation Requests

#	User Name	Email	Phone	Details	Preferred Contact	Photo	Request Date
19	Owen	owenjgibson@gmail.com	07712345678	This is a test	Email		2024-12-04 15:59:25

XSS

```
<th scope="row"><?php echo escape(html: $request['id']); ?></th>
<td><?php echo escape(html: $request['name']); ?></td>
<td><?php echo escape(html: $request['email']); ?></td>
<td><?php echo escape(html: $request['phone']); ?></td>
<td><?php echo nl2br(string: escape(html: $request['details'])); ?></td>
<td><?php echo ucfirst(string: escape(html: $request['preferred_contact'])); ?></td>
```

Get the requests form the database

```
try {
    $stmt = $pdo->prepare("
        SELECT er.id, er.details, er.preferred_contact, er.photo, er.request_date,
               u.name, u.email, u.phone
        FROM evaluation_requests er
        JOIN users u ON er.user_id = u.id
        ORDER BY er.request_date DESC
    ");
    $stmt->execute();
    $requests = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Gets the file path and check that it exists and if so, it gets the content

```
if (!empty($request['photo'])) {
    $filepath = '../uploads/' . basename(path: $request['photo']);
    if (file_exists(filename: $filepath)) {
        $encrypted_content = file_get_contents(filename: $filepath);
```

Decrypt the photo

```php
        if ($encrypted_content !== false) {
            $decrypted_content = decryptData(data: $encrypted_content);
            if ($decrypted_content !== false) {
                // Encode decrypted content to base64
                $request['decrypted_photo'] = base64_encode(string: $decrypted_content);
            } else {
                $request['decrypted_photo'] = null;
            }
        } else {
            $request['decrypted_photo'] = null;
        }
    } else {
        $request['decrypted_photo'] = null;
    }
} else {
    $request['decrypted_photo'] = null;
}
```

```php
/**
 * Decrypt data using AES-256-CBC
 */
function decryptData($data): bool|string {
    $data = base64_decode(string: $data);
    $iv_length = openssl_cipher_iv_length(cipher_algo: 'AES-256-CBC');
    $iv = substr(string: $data, offset: 0, length: $iv_length);
    $encrypted = substr(string: $data, offset: $iv_length);
    return openssl_decrypt(data: $encrypted, cipher_algo: 'AES-256-CBC', passphrase: ENCRYPTION_KEY, options: 0, iv:…$iv);
}
```

Checks that the user accessing the page is an admin

```php
// Ensure the user has 'admin' access level
checkAccess(requiredRole: 'admin');
```

```php
/**
 * Access control based on user role
 */
function checkAccess($requiredRole = 'user'): void {
    if (!isLoggedIn()) {
        header(header: 'Location: index.php');
        exit();
    }
    if ($requiredRole === 'admin' && !isAdmin()) {
        header(header: 'Location: index.php');
        exit();
    }
}
```

```php
/**
 * Check if user is an admin
 */
function isAdmin(): bool {
    return isset($_SESSION['is_admin']) && $_SESSION['is_admin'] === 1;
}
```

The encrypted photos can be found in the uploads file

Why do you think it is secure?

Same-origin, HTTPS and session fixation

- Same as in Task 1
- Can't access form unless admin

XSS

- The details are all sanitised using html special chars

Photo

- The photos are encrypted until you try and access the requests and then they are decrypted and can only be seen on that page

Task 6 – AWS Virtual Private Cloud settings screenshots

Instances Running

Name	Instance ID	Instance state
CompSec - ApacheMySQLPHP	i-0b2cb232c0a2afe2c	Running
CompSec - WS2016	i-0ab4f237ed7d8992a	Running

Instance summary for i-0b2cb232c0a2afe2c (CompSec - ApacheMySQLPHP) Info

Updated 3 minutes ago

Connect | Instance state | Actions

Instance ID i-0b2cb232c0a2afe2c	Public IPv4 address 100.27.32.21 | open address	Private IPv4 addresses 10.0.4.143
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-100-27-32-21.compute-1.amazonaws.com | open address
Hostname type IP name: ip-10-0-4-143.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-4-143.ec2.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 100.27.32.21 [Public IP]	VPC ID vpc-0dbbe7d63ee81a3c3 (CompSec-vpc)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-0d81aea6002186276 (CompSec-subnet-public3)	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:880190903685:instance/i-0b2cb232c0a2afe2c	

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

▼ Security details

IAM Role –	Owner ID 880190903685	Launch time Mon Nov 18 2024 15:09:12 GMT+0000 (Greenwich Mean Time)
Security groups sg-0fd80b31591b66603 (Public Subnet 3 - Security Group)		

Instance summary for i-0ab4f237ed7d8992a (CompSec - WS2016) Info

Updated less than a minute ago

Connect | Instance state | Actions

Instance ID i-0ab4f237ed7d8992a	Public IPv4 address 34.230.64.11 | open address	Private IPv4 addresses 10.0.2.48
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-34-230-64-11.compute-1.amazonaws.com | open address
Hostname type IP name: ip-10-0-2-48.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-2-48.ec2.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 34.230.64.11 [Public IP]	VPC ID vpc-0dbbe7d63ee81a3c3 (CompSec-vpc)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-08c639bf46caa052f (CompSec-subnet-public2)	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:880190903685:instance/i-0ab4f237ed7d8992a	

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

▼ Security details

IAM Role –	Owner ID 880190903685	Launch time Mon Nov 18 2024 15:05:21 GMT+0000 (Greenwich Mean Time)
Security groups sg-0147628cad993c89d (Public Subnet 2 -Security Group)		

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
–	sgr-0a5723573d7d904a8	443	TCP	0.0.0.0/0	Public Subnet 3 - Security Group	–
–	sgr-042894a4f0bfb88cd	80	TCP	0.0.0.0/0	Public Subnet 3 - Security Group	–
–	sgr-0e31048a5a290233b	3306	TCP	10.0.2.0/24	Public Subnet 3 - Security Group	–

Outbound rules

Name	Security group rule ID	Port range	Protocol	Destination	Security groups	Description
–	sgr-0167f1cbe22238a6f	All	All	0.0.0.0/0	Public Subnet 3 - Security Group	–

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
–	sgr-053fa20d1d13a3e34	3389	TCP	0.0.0.0/0	Public Subnet 2 -Security Group	–

Outbound rules

Name	Security group rule ID	Port range	Protocol	Destination	Security groups	Description
–	sgr-0e26dac87098a24c6	All	All	0.0.0.0/0	Public Subnet 2 -Security Group	–

Security groups rules

Apache MySQL and PHP

Inbound rules (3)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0a5723573d7d90...	IPv4	HTTPS	TCP	443	0.0.0.0/0	–
–	sgr-042894a4f0bfb88cd	IPv4	HTTP	TCP	80	0.0.0.0/0	–
–	sgr-0e31048a5a29023...	IPv4	MYSQL/Aurora	TCP	3306	10.0.2.0/24	–

Outbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
–	sgr-0e26dac87098a24c6	IPv4	All traffic	All	All	0.0.0.0/0	–

Windows server 2016

Inbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-053fa20d1d13a3e34	IPv4	RDP	TCP	3389	0.0.0.0/0	–

Outbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
–	sgr-0167f1cbe22238a6f	IPv4	All traffic	All	All	0.0.0.0/0	–

Routing tables

rtb-0bd230ed1a7433fcd / CompSec-rtb-private

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

Explicit subnet associations (2)

Edit subnet associations

Find subnet association

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
CompSec-subnet-private2	subnet-062ebd5daffcff753	10.0.3.0/24	–
CompSec-subnet-private1	subnet-09359423892872c05	10.0.1.0/24	–

rtb-08be7f7154d9addef / CompSec-rtb-public

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

Explicit subnet associations (3)

Edit subnet associations

Find subnet association

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
CompSec-subnet-public3	subnet-0d81aea6002186276	10.0.4.0/24	–
CompSec-subnet-public1	subnet-028769f81a7744e34	10.0.0.0/24	–
CompSec-subnet-public2	subnet-08c639bf46caa052f	10.0.2.0/24	–

All subnets

CompSec-subnet-public3	subnet-0d81aea6002186276	Available	vpc-0dbbe7d63ee81a3c3 | Com...	10.0.4.0/24
CompSec-subnet-public2	subnet-08c639bf46caa052f	Available	vpc-0dbbe7d63ee81a3c3 | Com...	10.0.2.0/24
CompSec-subnet-public1	subnet-028769f81a7744e34	Available	vpc-0dbbe7d63ee81a3c3 | Com...	10.0.0.0/24
CompSec-subnet-private2	subnet-062ebd5daffcff753	Available	vpc-0dbbe7d63ee81a3c3 | Com...	10.0.3.0/24
CompSec-subnet-private1	subnet-09359423892872c05	Available	vpc-0dbbe7d63ee81a3c3 | Com...	10.0.1.0/24

subnet-028769f81a7744e34 / CompSec-subnet-public1

Actions

Details

Subnet ID: subnet-028769f81a7744e34
Available IPv4 addresses: 250
Availability Zone ID: use1-az4
Network ACL: acl-045f7b8e3c15c0c43
Auto-assign customer-owned IPv4 address: No
IPv6 CIDR reservations: –
Resource name DNS AAAA record: Disabled

Subnet ARN: arn:aws:ec2:us-east-1:880190903685:subnet/subnet-028769f81a7744e34
IPv6 CIDR: –
Network border group: us-east-1
Default subnet: No
Customer-owned IPv4 pool: –
IPv6-only: No
DNS64: Disabled

State: Available
IPv6 CIDR association ID: –
VPC: vpc-0dbbe7d63ee81a3c3 | CompSec-vpc
Auto-assign public IPv4 address: No
Outpost ID: –
Hostname type: IP name
Owner: 880190903685

IPv4 CIDR: 10.0.0.0/24
Availability Zone: us-east-1a
Route table: rtb-08be7f7154d9addef | CompSec-rtb-public
Auto-assign IPv6 address: No
IPv4 CIDR reservations: –
Resource name DNS A record: Disabled

subnet-08c639bf46caa052f / CompSec-subnet-public2

Actions

Details

Subnet ID: subnet-08c639bf46caa052f
Available IPv4 addresses: 250
Availability Zone ID: use1-az6
Network ACL: acl-045f7b8e3c15c0c43
Auto-assign customer-owned IPv4 address: No
IPv6 CIDR reservations: –
Resource name DNS AAAA record: Disabled

Subnet ARN: arn:aws:ec2:us-east-1:880190903685:subnet/subnet-08c639bf46caa052f
IPv6 CIDR: –
Network border group: us-east-1
Default subnet: No
Customer-owned IPv4 pool: –
IPv6-only: No
DNS64: Disabled

State: Available
IPv6 CIDR association ID: –
VPC: vpc-0dbbe7d63ee81a3c3 | CompSec-vpc
Auto-assign public IPv4 address: No
Outpost ID: –
Hostname type: IP name
Owner: 880190903685

IPv4 CIDR: 10.0.2.0/24
Availability Zone: us-east-1b
Route table: rtb-08be7f7154d9addef | CompSec-rtb-public
Auto-assign IPv6 address: No
IPv4 CIDR reservations: –
Resource name DNS A record: Disabled

subnet-0d81aea6002186276 / CompSec-subnet-public3

Actions

Details

Subnet ID subnet-0d81aea6002186276	Subnet ARN arn:aws:ec2:us-east-1:880190903685:subnet/subnet-0d81aea6002186276	State Available	IPv4 CIDR 10.0.4.0/24
Available IPv4 addresses 250	IPv6 CIDR –	IPv6 CIDR association ID –	Availability Zone us-east-1c
Availability Zone ID use1-az1	Network border group us-east-1	VPC vpc-0dbbe7d63ee81a3c3 | CompSec-vpc	Route table rtb-08be7f7154d9addef | CompSec-rtb-public
Network ACL acl-045f7b8e3c15c0c43	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool –	Outpost ID –	IPv4 CIDR reservations –
IPv6 CIDR reservations –	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 880190903685	

subnet-09359423892872c05 / CompSec-subnet-private1

Actions

Details

Subnet ID subnet-09359423892872c05	Subnet ARN arn:aws:ec2:us-east-1:880190903685:subnet/subnet-09359423892872c05	State Available	IPv4 CIDR 10.0.1.0/24
Available IPv4 addresses 251	IPv6 CIDR –	IPv6 CIDR association ID –	Availability Zone us-east-1a
Availability Zone ID use1-az4	Network border group us-east-1	VPC vpc-0dbbe7d63ee81a3c3 | CompSec-vpc	Route table rtb-0bd230ed1a7433fcd | CompSec-rtb-private
Network ACL acl-045f7b8e3c15c0c43	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool –	Outpost ID –	IPv4 CIDR reservations –
IPv6 CIDR reservations –	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 880190903685	

subnet-062ebd5daffcff753 / CompSec-subnet-private2

Actions

Details

Subnet ID subnet-062ebd5daffcff753	Subnet ARN arn:aws:ec2:us-east-1:880190903685:subnet/subnet-062ebd5daffcff753	State Available	IPv4 CIDR 10.0.3.0/24
Available IPv4 addresses 251	IPv6 CIDR –	IPv6 CIDR association ID –	Availability Zone us-east-1b
Availability Zone ID use1-az6	Network border group us-east-1	VPC vpc-0dbbe7d63ee81a3c3 | CompSec-vpc	Route table rtb-0bd230ed1a7433fcd | CompSec-rtb-private
Network ACL acl-045f7b8e3c15c0c43	Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No
Auto-assign customer-owned IPv4 address No	Customer-owned IPv4 pool –	Outpost ID –	IPv4 CIDR reservations –
IPv6 CIDR reservations –	IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled
Resource name DNS AAAA record Disabled	DNS64 Disabled	Owner 880190903685	

vpc-0dbbe7d63ee81a3c3 / CompSec-vpc

Actions

Details Info

VPC ID vpc-0dbbe7d63ee81a3c3	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0e3a020b9005c9637	Main route table rtb-05d7161b32f4d583a	Main network ACL acl-045f7b8e3c15c0c43
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –	IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 880190903685	

sg-0147628cad993c89d - Public Subnet 2 -Security Group

Actions

Details

Security group name Public Subnet 2 -Security Group	Security group ID sg-0147628cad993c89d	Description Windows Server 2016	VPC ID vpc-0dbbe7d63ee81a3c3
Owner 880190903685	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

sg-0fd80b31591b66603 - Public Subnet 3 - Security Group

Actions

Details

Security group name Public Subnet 3 - Security Group	Security group ID sg-0fd80b31591b66603	Description Apache, MySQL and PHP	VPC ID vpc-0dbbe7d63ee81a3c3
Owner 880190903685	Inbound rules count 3 Permission entries	Outbound rules count 1 Permission entry	

Resource Map