Started on	Monday, 2 June 2025, 4:40 PM
State	Finished
Completed on	Monday, 2 June 2025, 4:47 PM
Time taken	7 mins 2 secs
Marks	14.00/16.00
Grade	**87.50** out of 100.00

Question 1

Complete

Mark 1.00 out of 1.00

How can you prevent JWT replay attacks in sensitive RBAC-based applications?

- a. Use longer expiration time
- b. Use only the frontend to validate roles
- c. Store tokens in localStorage
- d. Implement rotating refresh tokens

Question 2

Complete

Mark 1.00 out of 1.00

If a user's role is updated from "editor" to "admin", but their JWT hasn't expired yet, what is a potential risk?

- a. Signature gets mismatched
- b. Token becomes invalid immediately
- c. Token size increases
- d. Role update may not reflect until re-login

Question 3

Complete

Mark 1.00 out of 1.00

In a RBAC model, which principle is crucial for minimizing access privileges?

- a. Least privilege
- b. Token obfuscation
- c. Role inheritance
- d. Time-based access

Question 4

Complete

Mark 1.00 out of 1.00

In a secure RBAC system, where should the logic for role-based route protection ideally reside?

- a. JWT header
- b. Middleware or backend route handlers
- c. Frontend only
- d. Database triggers

Question 5

Complete

Mark 1.00 out of 1.00

What change should be made to the following JWT-based login handler to add RBAC? const token = jwt.sign({ id: user.id }, 'mysecret');

- a. Add user email to the payload
- b. Add role: user.role to payload
- c. Use HS512 algorithm
- d. Encrypt the token

Question 6

Complete

Mark 1.00 out of 1.00

What is a secure way to refresh a short-lived JWT without asking the user to log in again?

- a. Store token in sessionStorage
- b. Use a secure refresh token mechanism
- c. Use a cookie-stored access token
- d. Use the same JWT for 1 year

Question 7

Complete

Mark 0.00 out of 1.00

What is the primary purpose of the JWT signature?

- a. Encrypts the token data
- b. Validates the integrity and authenticity of the token
- c. Prevents cross-site scripting attacks
- d. Stores expiration timestamp

Question 8

Complete

Mark 1.00 out of 1.00

What is the problem with the following code if used in production? const token = jwt.sign({ userId: 1 }, '123', { expiresIn: '2h' });

- a. It uses numeric user ID
- b. Token will never expire
- c. Nothing, it's secure
- d. The secret is weak and predictable

Question 9

Complete

Mark 1.00 out of 1.00

What will happen if the secret key used to sign a JWT is leaked?

- a. Signature verification will be stricter
- b. Any user can generate valid tokens
- c. JWTs will auto-expire
- d. Token will become unreadable

Question 10

Complete

Mark 0.00 out of 1.00

Which claim in a JWT helps enforce token expiration?

- a. iat
- b. sub
- c. aud
- d. exp

Question 11

Complete

Mark 1.00 out of 1.00

Which part of a JWT is typically used to store user roles for implementing RBAC?

- a. Payload
- b. Header
- c. Signature
- d. Token Expiry

Question 12

Complete

Mark 1.00 out of 1.00

Why is storing a JWT in localStorage considered risky in web applications?

- a. It's vulnerable to XSS attacks
- b. It expires too quickly
- c. It increases backend load
- d. It cannot be read by JavaScript

Question 13

Complete

Mark 1.00 out of 1.00

Given the following code, which statement is true?

```
const MyComponent = React.memo(({ onClick }) => {
  console.log("Rendered");
  return <button onClick={onClick}>Click</button>;
});
```

What must be true for React.memo to prevent re-renders when parent re-renders?

- a. onClick must be stable across renders (e.g., memoized using useCallback)
- b. onClick must be memoized using useMemo
- c. React.memo always skips rendering regardless of prop types
- d. onClick must be declared outside the parent component

Question 14

Complete

Mark 1.00 out of 1.00

In which of the following scenarios is useMemo most beneficial?

- a. To prevent unnecessary re-renders of pure components
- b. To optimize expensive computations based on stable inputs
- c. To store global constants across modules
- d. To memoize functions used as event handlers

Question 15

Complete

Mark 1.00 out of 1.00

Consider the following component:

```
const List = React.memo(({ items }) => {
  return items.map(item => <div key={item.id}>{item.name}</div>);
});
```

If the parent re-renders but passes the same array reference for items, what happens?

- a. React.memo skips rendering because the array reference is unchanged
- b. React.memo skips rendering only if keys are stable
- c. React.memo deep compares array values
- d. React.memo causes List to re-render

Question 16

Complete

Mark 1.00 out of 1.00

Why might excessive use of useMemo lead to performance degradation rather than improvement?

- a. useMemo causes stale closures
- b. Creating memoized values and comparing dependencies has computational cost
- c. useMemo increases memory usage permanently
- d. React re-renders the component regardless of useMemo