Started on	Monday, 17 March 2025, 3:36 PM
State	Finished
Completed on	Monday, 17 March 2025, 3:41 PM
Time taken	5 mins 5 secs
Marks	14.00/15.00
Grade	**93.33** out of 100.00

Question 1

Complete

Mark 1.00 out of 1.00

Given the following vulnerable code, what type of attack can be performed?
exec(`ping ${req.body.host}`, (error, stdout, stderr) => { ... });

a. CSRF Attack
b. Cross-Site Scripting (XSS)
c. Command Injection
d. SQL Injection

Question 2

Complete

Mark 1.00 out of 1.00

How can Broken Access Control be exploited?

a. By logging in with the wrong password
b. By making too many API requests
c. By using a strong password
d. By modifying JWT tokens or accessing restricted APIs

Question 3

Complete

Mark 1.00 out of 1.00

How can the following function be exploited?

```
app.post('/track-vehicle', (req, res) => {
    const { plateNumber } = req.body;
    exec(`echo Tracking vehicle ${plateNumber}`, (error, stdout, stderr) => { ... });
});
```

a. By using a VPN
b. By making multiple requests at the same time
c. By sending an empty request body
d. By injecting shell commands in the plateNumber field

Question 4

Complete

Mark 1.00 out of 1.00

If a user inputs `ABC123 && rm -rf /`, what will happen on a Linux server?

- a. The entire file system could be deleted
- b. The vehicle tracking system will show an error
- c. Nothing will happen
- d. The server will shut down immediately

Question 5

Complete

Mark 1.00 out of 1.00

What command could an attacker enter in the `/track-vehicle` endpoint to delete files on a Windows system?

- a. ABC123 && mv /etc/passwd /dev/null
- b. ABC123 && del C:\Windows\System32
- c. ABC123; rm -rf /
- d. ABC123 && shutdown -h now

Question 6

Complete

Mark 1.00 out of 1.00

What is the best way to prevent command injection attacks?

- a. Use eval() to process user input
- b. Use parameterized queries and sanitize input
- c. Use an insecure API to execute shell commands
- d. Allow user input directly in system commands

Question 7

Complete

Mark 1.00 out of 1.00

What is the correct way to restrict access to admin users only?

- a. if (decoded.role !== 'user') return res.status(403).json({ error: 'Forbidden' });
- b. if (decoded.role !== 'admin') return res.status(403).json({ error: 'Forbidden' });
- c. if (decoded.id === 1) return res.status(403).json({ error: 'Forbidden' });
- d. if (!decoded.role) return res.status(403).json({ error: 'Forbidden' });

Question 8

Complete

Mark 1.00 out of 1.00

What is the impact of Broken Access Control on an application?

a. Attackers can execute arbitrary commands on the server
b. The database gets automatically deleted
c. Unauthorized users can access restricted information or perform admin actions
d. It allows Cross-Site Scripting (XSS)

Question 9

Complete

Mark 1.00 out of 1.00

What is the primary cause of command injection vulnerabilities in applications?

a. Incorrect use of loops in JavaScript
b. Poor network security configuration
c. Lack of input validation when executing system commands
d. Using HTTPS instead of HTTP

Question 10

Complete

Mark 1.00 out of 1.00

What is the safest way to execute system commands in Node.js?

a. Concatenating user input into system commands
b. Using execFile() with sanitized input
c. Using eval()
d. Using exec() with user input

Question 11

Complete

Mark 1.00 out of 1.00

What security flaw exists in the following `/users` endpoint?

```
app.get('/users', (req, res) => {
    const token = req.headers.authorization;
    jwt.verify(token, SECRET_KEY, (err, decoded) => {
        db.query('SELECT id, username, role FROM users', (err, results) => {
            res.json({ users: results });
        });
    });
});
```

a. It does not return JSON data
b. It does not verify the user's role before returning data
c. It does not store passwords securely
d. It is vulnerable to SQL injection

Question 12

Complete

Mark 0.00 out of 1.00

What would happen if an attacker modified a JWT token to escalate their privileges?

a. They could access admin-only features
b. The server would detect the modification and reject the request
c. The token would expire immediately
d. They would get logged out

Question 13

Complete

Mark 1.00 out of 1.00

Which function is the most dangerous when handling user input in Node.js?

a. parseInt()
b. JSON.stringify()
c. console.log()
d. exec()

Question 14

Complete

Mark 1.00 out of 1.00

Which of the following is an effective way to prevent Broken Access Control?

- a. Store JWT tokens in Local Storage without encryption
- b. Validate user roles and permissions before processing requests
- c. Remove authentication from sensitive endpoints
- d. Allow users to modify their own JWT tokens

Question 15

Complete

Mark 1.00 out of 1.00

Why is the following endpoint a security risk?

```
app.get('/users', (req, res) => {
    db.query('SELECT id, username, role FROM users', (err, results) => {
        res.json({ users: results });
    });
});
```

- a. It uses HTTPS instead of HTTP
- b. It allows SQL Injection
- c. It exposes all users' details without authentication
- d. It is vulnerable to CSRF