Server-side JavaScript Injection(SSJI)

Server Side JavaScript injection is the ability for a user to inject code which will in turn be evaluated by the server, and therefore would allow an attacker to potentially execute arbitrary code under the context of the server and interaction with the filesystem, which may lead to the full compromise of the host.

Some JS functions can be exploited by an attacker to execute malicious JS code on the server:

- eval()
- setTimeout()
- setInterval()
- Function()

Inadequate input validation

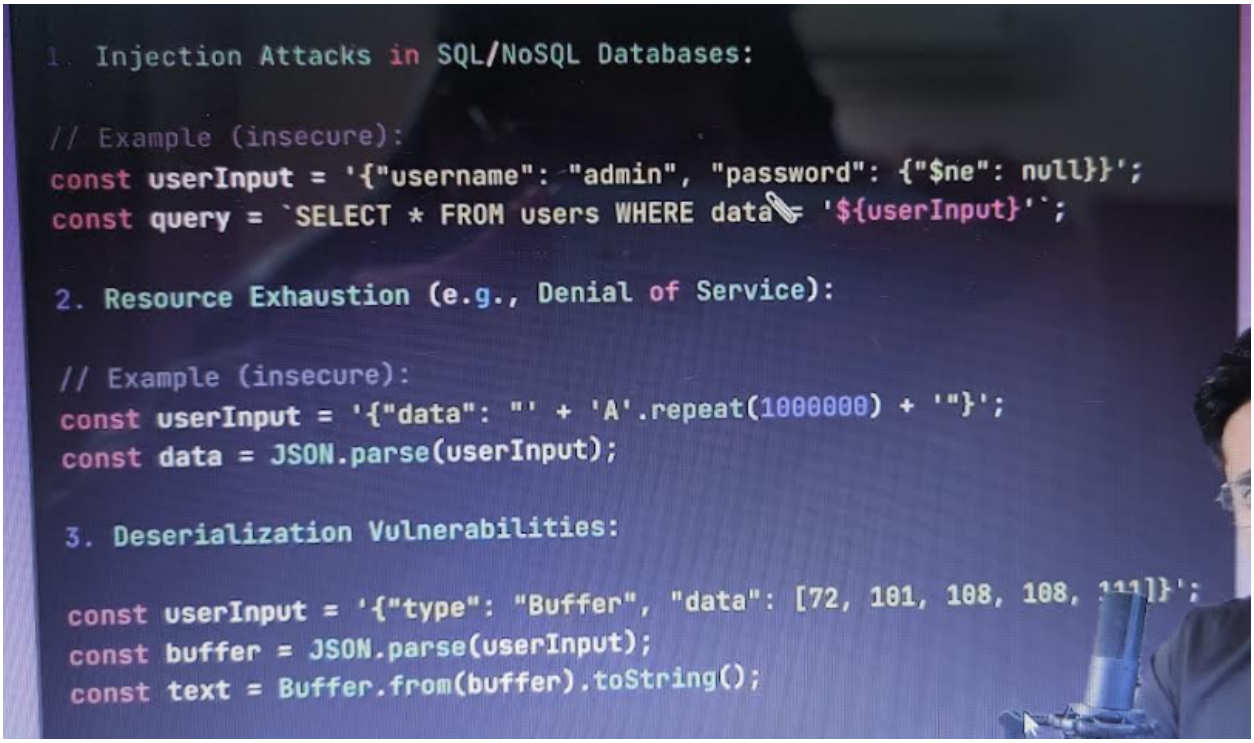
Direct execution of user provided code

Using dangerous function

Using new Function() in JS

Insecure deserialization

There is some JSON data that is coming and we do deserialization without checking

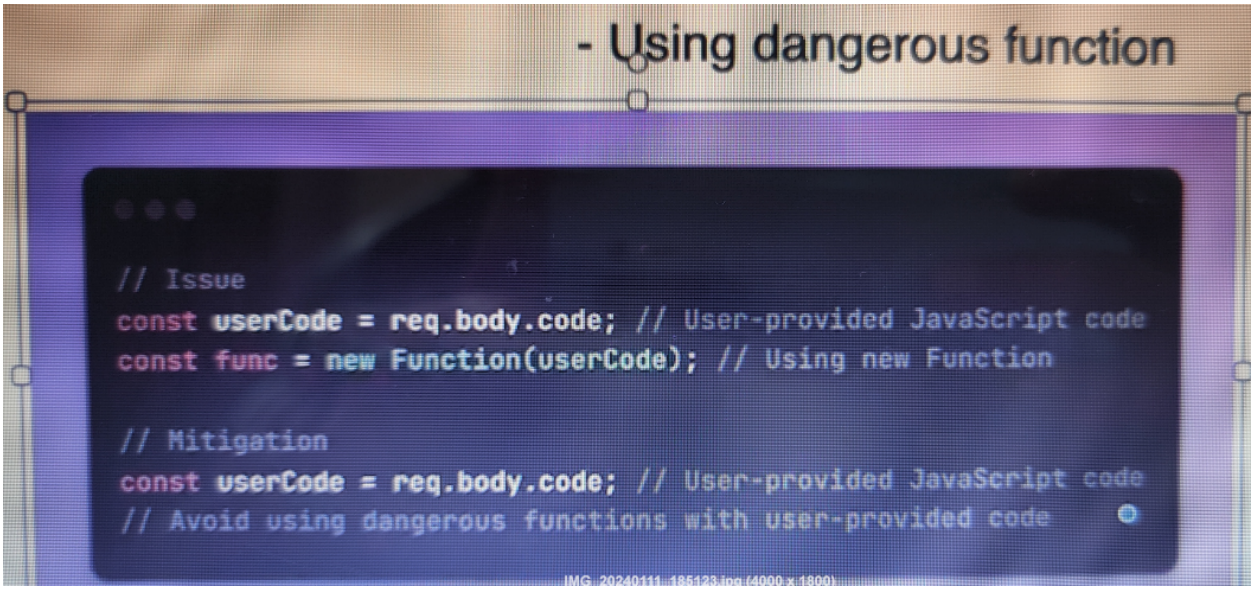

1. Injection Attacks in SQL/NoSQL Databases:
// Example (insecure):
const userInput = '{"username": "admin", "password": {"$ne": null}}';
const query = `SELECT * FROM users WHERE data = '${userInput}'`;
2. Resource Exhaustion (e.g., Denial of Service):
// Example (insecure):
const userInput = '{"data": "' + 'A'.repeat(1000000) + '"}';
const data = JSON.parse(userInput);
3. Deserialization Vulnerabilities:
const userInput = '{"type": "Buffer", "data": [72, 101, 108, 108, 111]}';
const buffer = JSON.parse(userInput);
const text = Buffer.from(buffer).toString();

- Inadequate input validation

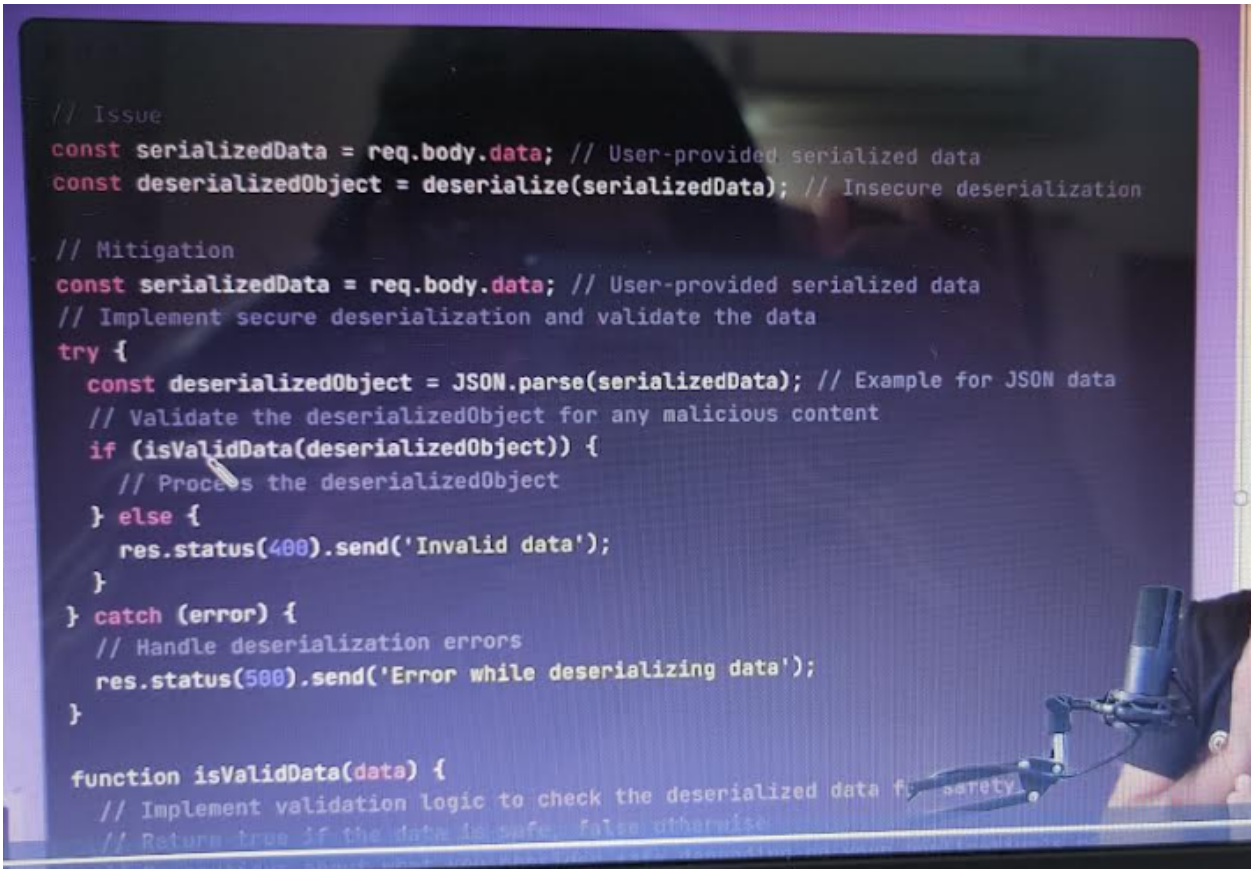
```
// Issue
const userInput = req.body.input; // User-provided input

// Mitigation
const userInput = req.body.input; // User-provided input
if (!isValidInput(user-input)) {
  // Handle the invalid input, e.g., return an error or sanitize the input
  res.status(400).send('Invalid input');
}

function isValidInput(input) {
  // Implement proper input validation logic here
  // Reject any input that contains potentially dangerous characters or patte
  const regex = /^[a-zA-Z0-9\s]+$/; // Example: Allow only letters, numbers,
  return regex.test(input);
}
```

- Direct execution of user provided code

```
// Issue
const userCode = req.body.code; // User-provided JavaScript code
eval(userCode); // Directly executing user code

// Mitigation
const userCode = req.body.code; // User-provided JavaScript co
// Do not directly execute user-provided code
```



Any issue in setTimeout/setInterval can lead to global exceptions. If global exceptions are not handled properly, it can do lot of damage.


// Issue
const serializedData = req.body.data; // User-provided serialized data
const deserializedObject = deserialize(serializedData); // Insecure deserialization
// Mitigation
const serializedData = req.body.data; // User-provided serialized data
// Implement secure deserialization and validate the data
try {
const deserializedObject = JSON.parse(serializedData); // Example for JSON data
// Validate the deserializedObject for any malicious content
if (isValidData(deserializedObject)) {
// Process the deserializedObject
} else {
res.status(400).send('Invalid data');
}
} catch (error) {
// Handle deserialization errors
res.status(500).send('Error while deserializing data');
}
function isValidData(data) {
// Implement validation logic to check the deserialized data