[LOCK] Token Security Lab

Setup Guide - Building Your Vulnerable Testing Environment

[DOC] Document Purpose: This guide walks you through creating a deliberately vulnerable Node.js web application for security testing and education. This is Part 1 of 2 - the companion Challenge Guide contains the pentesting exercises.

[BOOK] Introduction

What You're Building

You will construct a fully functional web application with intentional security vulnerabilities related to token-based authentication. This controlled environment allows you to:

- **Practice offensive security techniques** in a safe, legal environment
- **Understand real-world vulnerabilities** found in production applications
- **Learn secure coding practices** by first experiencing the flaws
- **Develop penetration testing skills** without ethical or legal concerns

Learning Objectives

By completing this lab setup, you will:

1. Understand the architecture of modern token-based authentication systems
2. Learn how Node.js/Express applications handle authentication flows
3. Recognize the difference between secure and insecure storage mechanisms
4. Build foundational knowledge of HTTP vs HTTPS security implications
5. Gain hands-on experience with modern web development tools

> **!? Legal & Ethical Notice:** This lab is for educational purposes ONLY. The vulnerabilities demonstrated here should NEVER be implemented in production systems. Only perform security testing on systems you own or have explicit written permission to test.

[TARGET] Target Audience

This lab is designed for:

- **Security Analysts** learning web application penetration testing
- **Developers** wanting to understand authentication vulnerabilities
- **Students** in cybersecurity or computer science programs
- **IT Professionals** responsible for security compliance and auditing
- **Bug Bounty Hunters** practicing vulnerability discovery techniques

Prerequisites

Requirement	Level	Details
Programming Knowledge	Basic	Understanding of JavaScript fundamentals
Web Concepts	Basic	Knowledge of HTTP, HTML, and browser DevTools
Command Line	Basic	Ability to navigate directories and run commands
Security Knowledge	None Required	Lab teaches from fundamentals

[TOOL]? System Requirements

Software Requirements

Software	Minimum Version	Purpose
Node.js	14.x or higher	JavaScript runtime environment
npm	6.x or higher	Package manager (included with Node.js)
Text Editor	Any	VS Code, Sublime, Notepad++ recommended
Web Browser	Modern	Chrome, Edge, or Firefox with DevTools

Hardware Requirements

- **RAM:** 4GB minimum, 8GB recommended
- **Storage:** 500MB free space
- **CPU:** Any modern processor (dual-core minimum)
- **Network:** Internet connection for initial package downloads

? Installing Node.js and npm

Windows Installation

1. Visit **https://nodejs.org**
2. Download the **LTS (Long Term Support)** version
3. Run the installer (.msi file)
4. Follow the installation wizard (accept default options)
5. Verify installation by opening PowerShell or Command Prompt and running:

```
node --version && npm --version
```

Expected output: Version numbers like `v18.17.0` and `9.6.7`

macOS Installation

1. **Using Homebrew (recommended):**

```
brew install node
```

1. **Using Official Installer:**
2. Download from **https://nodejs.org**
3. Install the .pkg file
4. Verify with: `node --version && npm --version`

Linux Installation

Ubuntu/Debian:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
sudo apt-get install -y nodejs
node --version && npm --version
```

Fedora/RHEL:

```
sudo dnf install nodejs
node --version && npm --version
```

?? Project Structure Overview

Before creating files, understand the architecture:

```
TokenLab/
?
??? package.json          <- Project configuration & dependencies
??? server.js             <- Backend API (Express server)
?
??? public/               <- Frontend files (served to browser)
    ??? index.html        <- User interface
    ??? app.js            <- Client-side authentication logic
```

Component Roles

File	Technology	Responsibility
package.json	JSON	Defines project metadata and dependencies
server.js	Node.js/ Express	Handles login, token generation, API endpoints
index.html	HTML/CSS	Login form and admin panel UI
app.js	JavaScript	Manages client-side token storage & API calls

? Step-by-Step Setup

1 Create Project Directory

Windows (PowerShell/Command Prompt):

```
mkdir TokenLab
cd TokenLab
```

macOS/Linux (Terminal):

```
mkdir TokenLab
cd TokenLab
```

> **[TIP] Tip:** You can create this folder anywhere on your system. Desktop or Documents folder works well for easy access.

2 Create package.json

This file tells npm what packages to install and how to run the project.

[PAGE] File: package.json

```
{
  "name": "token-lab",
  "version": "1.0.0",
  "description": "Vulnerable web app for token security training",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "node server.js"
  },
  "keywords": ["security", "pentesting", "tokens", "jwt"],
  "author": "Security Training Lab",
  "license": "MIT",
  "dependencies": {
    "express": "^4.18.2",
    "cors": "^2.8.5",
    "body-parser": "^1.20.2"
  }
}
```

Understanding Dependencies

- **express:** Web framework for building the API server
- **cors:** Allows cross-origin requests (frontend to backend communication)
- **body-parser:** Parses JSON data from HTTP requests

❸ Create server.js (Backend)

This is the heart of your vulnerable application. It contains intentional security flaws.

[PAGE] File: server.js

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const app = express();

// Middleware Configuration
app.use(cors());                    // Enable CORS for all routes
app.use(bodyParser.json());         // Parse JSON request bodies
app.use(express.static('public'));  // Serve static files from 'public' folder

// ========================================
// MOCK DATABASE & CONFIGURATION
// ========================================
const USERS = {
    'admin': 'admin123',
    'user': 'password123'
};

const SECRET_DATA = "FLAG{Network_Sniffing_Master_882}";

// ========================================
// VULNERABLE ENDPOINT #1: LOGIN
// ========================================
// VULNERABILITY: Returns token in response body (Finding 4)
// VULNERABILITY: Served over HTTP (Finding 5)
app.post('/login', (req, res) => {
    const { username, password } = req.body;

    if (USERS[username] && USERS[username] === password) {
        // Generate mock token (in reality, this would be a signed JWT)
        // Format: Base64-encoded JSON object
        const rawPayload = JSON.stringify({
            user: username,
            role: 'superuser',
            id: 999,
            issued: Date.now()
        });

        const token = Buffer.from(rawPayload).toString('base64');

        console.log(`[LOG] User ${username} logged in at ${new
Date().toISOString()}`);
        console.log(`[LOG] Token: ${token}`);

        // INSECURE: Token sent in response body
        res.json({
            success: true,
            token: token,
```

```
            message: 'Authentication successful'
        });
    } else {
        console.log(`[FAILED] Login attempt for username: ${username}`);
        res.status(401).json({
            success: false,
            error: 'Invalid credentials'
        });
    }
});

// ========================================
// VULNERABLE ENDPOINT #2: CLASSIFIED DATA
// ========================================
// VULNERABILITY: Accepts token from Authorization header
// VULNERABILITY: Transmitted over unencrypted HTTP
app.get('/api/classified', (req, res) => {
    const authHeader = req.headers.authorization;

    if (authHeader && authHeader.startsWith('Bearer ')) {
        const token = authHeader.split(' ')[1];

        try {
            // Decode mock token (simulating JWT verification)
            const payload = Buffer.from(token, 'base64').toString('ascii');
            console.log(`[ACCESS] Classified data requested with payload: $
{payload}`);

            if (payload.includes('superuser')) {
                res.json({
                    message: "Access Granted to Classified Information",
                    secret_flag: SECRET_DATA,
                    timestamp: new Date().toISOString()
                });
            } else {
                res.status(403).json({
                    error: "Insufficient permissions - Superuser role
required"
                });
            }
        } catch (err) {
            res.status(400).json({
                error: "Invalid token format"
            });
        }
    } else {
        res.status(401).json({
            error: "Missing Authorization Header - Format: Bearer "
        });
```

```
    }
});

// ========================================
// SERVER STARTUP
// ========================================
const PORT = 3000;
app.listen(PORT, () => {
    console.log('================================================');
    console.log('[TARGET] VULNERABLE TARGET RANGE ACTIVE');
    console.log('================================================');
    console.log(`[SIGNAL] Server URL: http://localhost:${PORT}`);
    console.log(`[UNLOCK] Credentials: admin / admin123`);
    console.log(`!?  WARNING: Contains intentional vulnerabilities`);
    console.log(`[READ] Use only for authorized security training`);
    console.log('================================================');
});
```

[ALERT] Security Note: This server runs on HTTP (port 3000) and contains multiple vulnerabilities:

- Tokens sent in JSON response (client must store them)
- No HTTPS encryption
- Weak token validation
- No token expiration enforcement

These are intentional for learning purposes.

4 Create public Folder

This folder holds the frontend files that users interact with.

Command:

```
mkdir public
```

❺ Create index.html (Frontend UI)

Navigate into the `public` folder and create this file.

[PAGE] File: public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Insecure Token Lab - Target System</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Courier New', monospace;
            background: #0a0a0a;
            color: #00ff00;
            padding: 20px;
            min-height: 100vh;
            display: flex;
            align-items: center;
            justify-content: center;
        }

        .container {
            max-width: 800px;
            width: 100%;
        }

        h1 {
            text-align: center;
            color: #00ff00;
            text-shadow: 0 0 10px #00ff00;
            margin-bottom: 30px;
            font-size: 2em;
            letter-spacing: 3px;
        }

        .panel {
            border: 2px solid #00ff00;
            padding: 30px;
            margin-bottom: 20px;
            background: rgba(0, 255, 0, 0.05);
            border-radius: 5px;
            box-shadow: 0 0 20px rgba(0, 255, 0, 0.3);
        }

        h3 {
```

```
            color: #00ff00;
            margin-bottom: 20px;
            font-size: 1.3em;
        }

        input {
            background: #1a1a1a;
            color: #00ff00;
            border: 2px solid #00ff00;
            padding: 12px;
            width: 100%;
            margin: 10px 0;
            box-sizing: border-box;
            font-family: 'Courier New', monospace;
            font-size: 14px;
        }

        input:focus {
            outline: none;
            box-shadow: 0 0 10px #00ff00;
        }

        button {
            background: #00ff00;
            color: #0a0a0a;
            border: none;
            padding: 12px 25px;
            font-weight: bold;
            cursor: pointer;
            margin-top: 15px;
            font-family: 'Courier New', monospace;
            font-size: 14px;
            transition: all 0.3s;
            text-transform: uppercase;
            letter-spacing: 1px;
        }

        button:hover {
            background: #00cc00;
            box-shadow: 0 0 15px #00ff00;
            transform: translateY(-2px);
        }

        button:active {
            transform: translateY(0);
        }

        #output {
            border-top: 1px dashed #00ff00;
```

```
            margin-top: 15px;
            padding-top: 15px;
            display: none;
            white-space: pre-wrap;
            word-wrap: break-word;
        }

        .badge {
            background: #ff0000;
            color: white;
            padding: 3px 8px;
            font-size: 0.7em;
            border-radius: 3px;
            margin-left: 10px;
            animation: pulse 2s infinite;
        }

        @keyframes pulse {
            0%, 100% { opacity: 1; }
            50% { opacity: 0.6; }
        }

        .status-line {
            margin: 10px 0;
            font-size: 0.9em;
        }

        .status-line strong {
            color: #ffffff;
        }

        .logout-btn {
            background: #333333;
            color: #00ff00;
            border: 1px solid #00ff00;
        }

        .logout-btn:hover {
            background: #444444;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>!? TARGET_SYSTEM_v1.0 !?</h1>

        <!-- Login Panel -->
        <div id="login-panel" class="panel">
            <h3>:: AUTHENTICATION REQUIRED ::</h3>
```

```
            <input type="text" id="username" placeholder="Username"
value="admin">
            <input type="password" id="password" placeholder="Password"
value="admin123">
            <button onclick="login()">? INITIATE SESSION</button>
        </div>

        <!-- Admin Panel (Hidden until login) -->
        <div id="admin-panel" class="panel" style="display:none;">
            <h3>:: ADMIN CONSOLE ::</h3>
            <div class="status-line">
                <strong>Status:</strong> <span style="color:#00ff00">v
AUTHENTICATED</span>
            </div>
            <div class="status-line">
                <strong>Storage:</strong>
                <span class="badge">INSECURE (localStorage)</span>
            </div>
            <div class="status-line">
                <strong>Protocol:</strong>
                <span class="badge">UNENCRYPTED (HTTP)</span>
            </div>

            <button onclick="getClassified()">[UNLOCK] RETRIEVE CLASSIFIED
DATA</button>
            <button onclick="logout()" class="logout-btn">[POWER] TERMINATE
SESSION</button>

            <div id="output"></div>
        </div>
    </div>

    <script src="app.js"></script>
</body>
</html>
```

6 Create app.js (Frontend Logic)

This file contains the vulnerable client-side JavaScript that manages authentication.

[PAGE] File: public/app.js

```
// API Configuration
const API = 'http://localhost:3000';

// ========================================
// LOGIN FUNCTION
// ========================================
function login() {
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    console.log('[CLIENT] Attempting login...');

    fetch(`${API}/login`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            username: username,
            password: password
        })
    })
    .then(response => response.json())
    .then(data => {
        if(data.success) {
            console.log('[CLIENT] Login successful!');
            console.log('[CLIENT] Token received:', data.token);

            // VULNERABILITY: Storing token in localStorage
            // This makes it accessible to any JavaScript code
            localStorage.setItem('session_token', data.token);

            showPanel(true);

            // Display success message
            alert('v Authentication Successful');
        } else {
            console.error('[CLIENT] Login failed:', data.error);
            alert('? Access Denied: ' + data.error);
        }
    })
    .catch(error => {
        console.error('[CLIENT] Network error:', error);
        alert('Network error - Is the server running?');
    });
}

// ========================================
```

```
// GET CLASSIFIED DATA FUNCTION
// ========================================
function getClassified() {
    // VULNERABILITY: Reading token from localStorage
    const token = localStorage.getItem('session_token');

    if (!token) {
        alert('No token found - Please login first');
        return;
    }

    console.log('[CLIENT] Requesting classified data...');
    console.log('[CLIENT] Using token:', token);

    fetch(`${API}/api/classified`, {
        headers: {
            // VULNERABILITY: Token transmitted over unencrypted HTTP
            'Authorization': 'Bearer ' + token
        }
    })
    .then(response => response.json())
    .then(data => {
        console.log('[CLIENT] Response received:', data);

        const output = document.getElementById('output');
        output.style.display = 'block';
        output.innerHTML = '[TARGET] SERVER RESPONSE:\n\n' +
JSON.stringify(data, null, 2);
    })
    .catch(error => {
        console.error('[CLIENT] Request failed:', error);
        alert('Failed to retrieve data');
    });
}

// ========================================
// UI MANAGEMENT FUNCTIONS
// ========================================
function showPanel(isAuthenticated) {
    document.getElementById('login-panel').style.display =
        isAuthenticated ? 'none' : 'block';
    document.getElementById('admin-panel').style.display =
        isAuthenticated ? 'block' : 'none';
}

function logout() {
    console.log('[CLIENT] Logging out...');
    localStorage.removeItem('session_token');
    location.reload();
```

```
}

// ========================================
// AUTO-LOGIN CHECK (On page load)
// ========================================
window.addEventListener('DOMContentLoaded', () => {
    const existingToken = localStorage.getItem('session_token');
    if (existingToken) {
        console.log('[CLIENT] Existing session found');
        showPanel(true);
    }
});
```

!? Vulnerability Summary in app.js:

- **Line 32:** Token stored in `localStorage` (vulnerable to XSS)
- **Line 50:** Token read from `localStorage`
- **Line 60:** Token sent over HTTP in Authorization header (network sniffing risk)

? Installation & Launch

7 Install Dependencies

Open your terminal/command prompt in the `TokenLab` folder and run:

```
npm install
```

You should see output similar to:

```
added 57 packages, and audited 58 packages in 3s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

> **? What just happened?** npm read your `package.json` file and downloaded Express, CORS, and body-parser libraries into a new `node_modules` folder.

8 Start the Server

Run the following command:

```
npm start
```

You should see:

```
==============================================
[TARGET] VULNERABLE TARGET RANGE ACTIVE
==============================================
[SIGNAL] Server URL: http://localhost:3000
[UNLOCK] Credentials: admin / admin123
!?  WARNING: Contains intentional vulnerabilities
[READ] Use only for authorized security training
==============================================
```

[OK] Success! Your vulnerable lab environment is now running.

9 Access the Application

1. Open your web browser (Chrome or Edge recommended)
2. Navigate to: **http://localhost:3000**
3. You should see a terminal-style login page with green text on black background

[TIP] What is localhost:3000?

- **localhost:** Refers to your own computer (127.0.0.1)
- **:3000:** The port number where the server is listening
- This means the server is only accessible from your machine (safe for testing)

[OK] Verification Checklist

Ensure everything is working before proceeding to the Challenge Guide:

Step	Expected Result	Status
Navigate to http://localhost:3000	Login page loads with green terminal theme	?
Enter credentials: admin / admin123	Click "INITIATE SESSION" button	?
After login	Admin console appears with "AUTHENTICATED" status	?
Click "RETRIEVE CLASSIFIED DATA"	JSON response with secret_flag appears	?
Open DevTools (F12)	Console shows client/server logs	?
Check Application tab -> Local Storage	session_token visible	?

? Troubleshooting

Problem: "npm: command not found"

Solution: Node.js is not installed or not in your system PATH. Reinstall Node.js from nodejs.org and restart your terminal.

Problem: "Error: Cannot find module 'express'"

Solution: Dependencies not installed. Run `npm install` in the TokenLab directory.

Problem: "Port 3000 already in use"

Solution: Another application is using port 3000. Either:

- Close the other application
- Change port in server.js: `const PORT = 3001;`
- Change port in app.js: `const API = 'http://localhost:3001';`

Problem: "Cannot GET /"

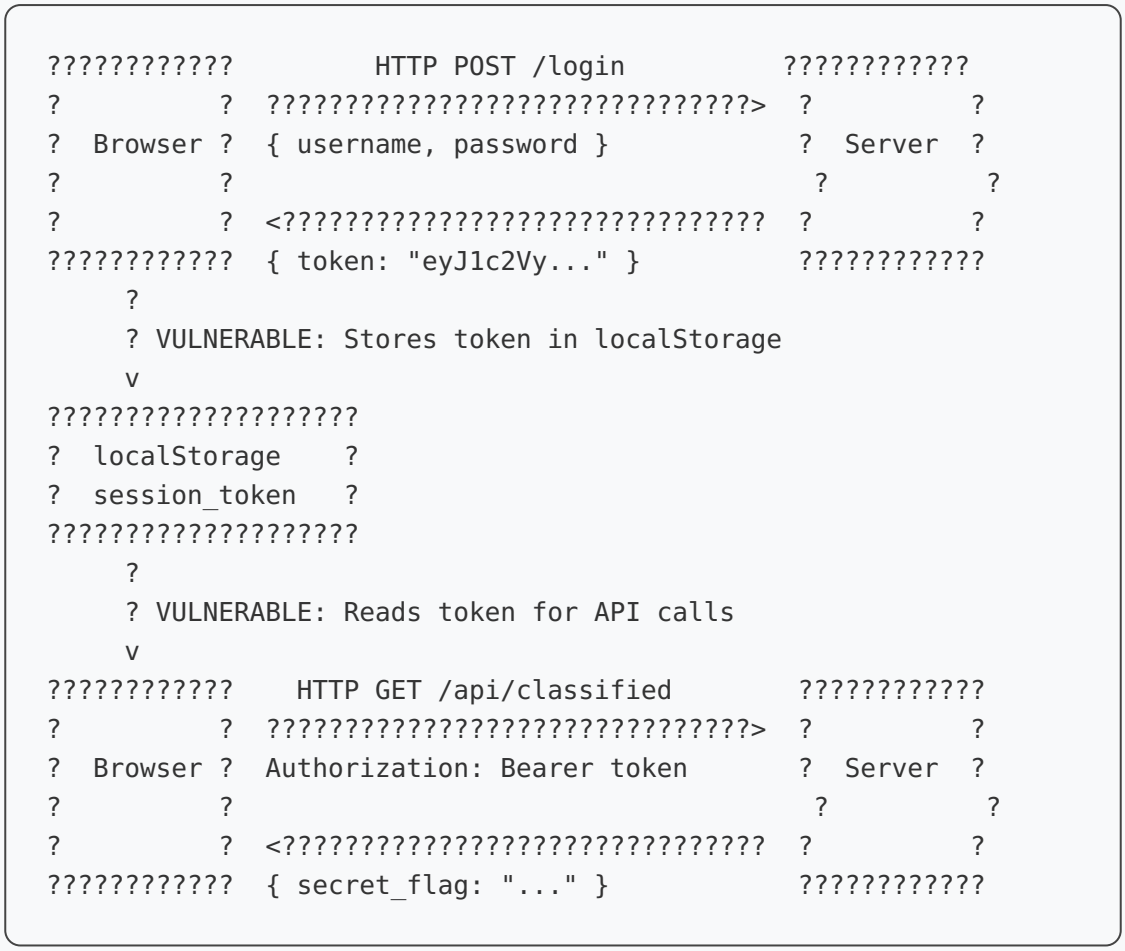
Solution: The public folder or index.html is missing. Verify folder structure matches the architecture diagram.

Problem: Browser shows "This site can't be reached"

Solution: Server is not running. Open terminal, navigate to TokenLab folder, and run `npm start`.

[READ] Understanding the Architecture

Authentication Flow



Why These Vulnerabilities Exist

Vulnerability	Developer Reasoning	Why It's Dangerous
localStorage storage	"It's simple and persists across sessions"	Accessible to any JavaScript (XSS attacks)
HTTP (not HTTPS)	"HTTPS certificates are complicated"	Tokens transmitted in plain text (network sniffing)
Token in response body	"Frontend needs to store it somewhere"	Forces insecure client-side storage
No token expiration	"Users don't want to re-login constantly"	Stolen tokens valid indefinitely

[GRAD] Next Steps

[TARGET] Setup Complete! Your vulnerable lab environment is ready for pentesting.

Now proceed to the companion document:

? Token Security Masterclass - Challenge Guide

The Challenge Guide contains:

- 4 Capture-the-Flag (CTF) style challenges
- Step-by-step exploitation techniques
- Deep-dive foundational knowledge on tokens and authentication
- Secure remediation code examples
- Answer key with detailed explanations

!? Remember: Keep the server running (`npm start`) while working through the challenges. You can stop it anytime with `Ctrl+C` in the terminal.

[BOOK] Additional Resources

Learning Resources

- **OWASP Top 10:** https://owasp.org/www-project-top-ten/
- **JWT.io:** https://jwt.io/ (Learn about JSON Web Tokens)
- **PortSwigger Web Security Academy:** https://portswigger.net/web-security
- **MDN Web Docs (localStorage):** https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

Tools for Further Exploration

- **Burp Suite:** Professional web vulnerability scanner
- **OWASP ZAP:** Free security testing tool
- **Wireshark:** Network protocol analyzer for packet sniffing
- **Chrome DevTools:** Built-in browser debugging (press F12)

Document Version: 1.0 | **Last Updated:** 2024

Disclaimer: The techniques demonstrated in this lab should only be used in authorized testing environments. Unauthorized access to computer systems is illegal.