**INCIDENT REPORT: BN-19003-Download-Secured-Documents**

**Date: 6/28/2024**

**Executive Summary:**

I was tasked with exploiting a null byte injection vulnerability in OWASP Juice Shop to download a hidden backup JSON file (package.json.bak) from the server.

**Results:**

* Successfully downloaded package.json.bak (a backup configuration file) using a null byte injection.
* Downloaded eastere.gg file, which contained a Base64 encoded message that, when decoded and deciphered using ROT13, revealed a clue.

(The owl is in front of the door to the right of the guard of the gate)

**Application Details:**

* Burp Suite Community Edition: v2024.4.4.5
* OWASP Juice Shop: Latest version (running on Docker)

**Attack Narrative:**

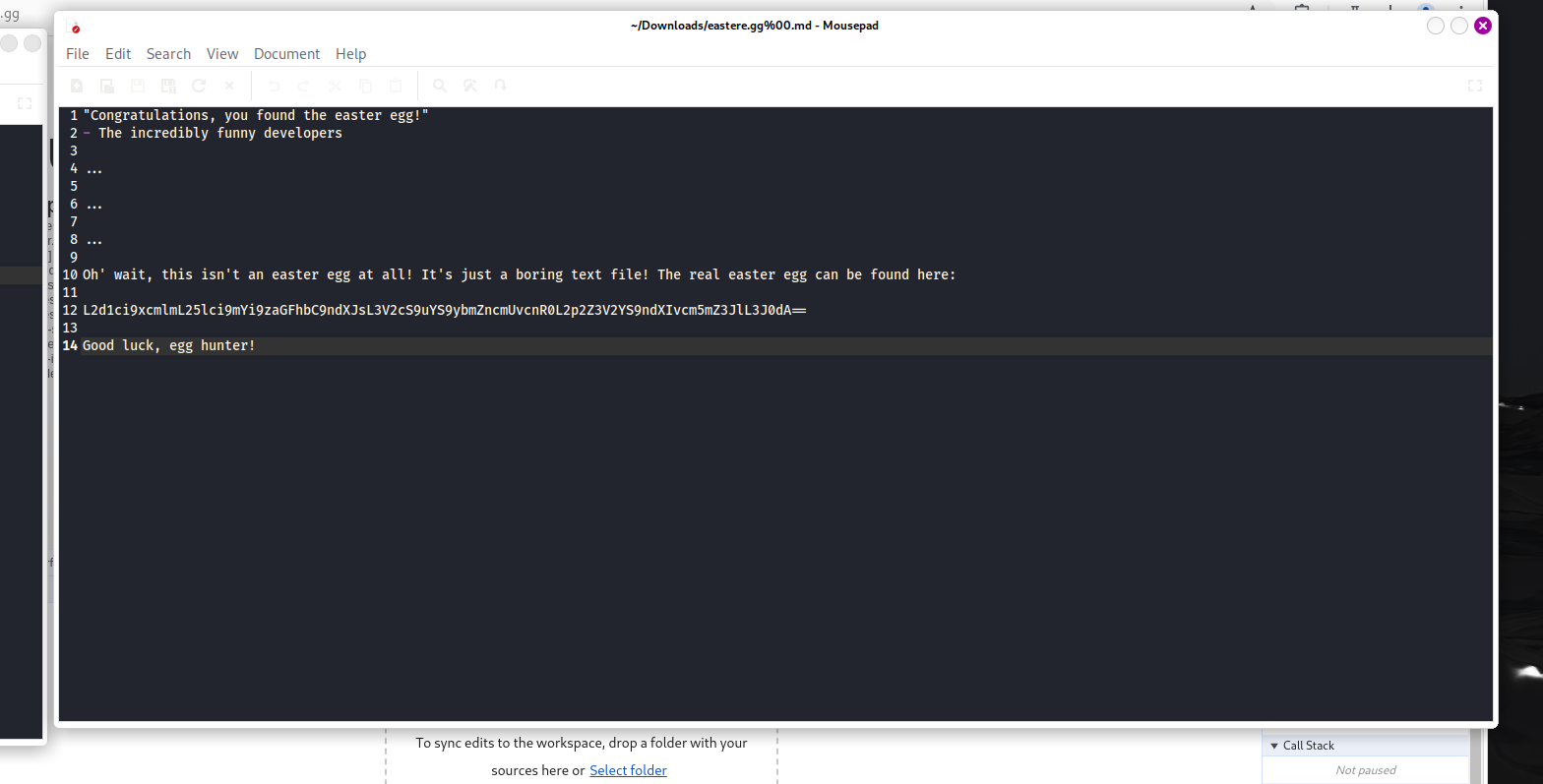
1. **Identified Vulnerable Endpoint:** While interacting with the Juice Shop application, I found a functionality that allowed downloading files (e.g., product images, user manuals).
2. **Initial Attempt:** I attempted to manipulate the file download URL by appending %00.md to the filename. This resulted in an error, likely due to URL encoding filtering.
3. **Successful Payload:** Recognizing the need for URL encoding, I modified the payload to %2500.md. This bypassed the encoding and allowed me to download package.json.bak from the /ftp/ directory. I used the same technique to download eastere.gg, which contained a hidden message within the application.

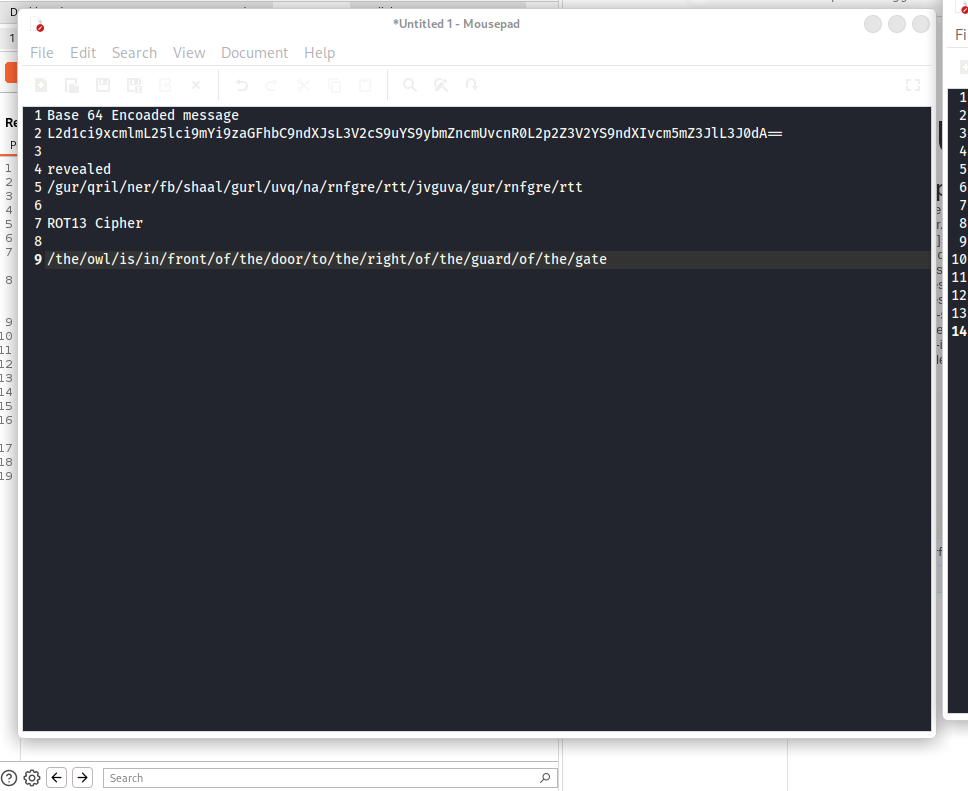
**Conclusion:**

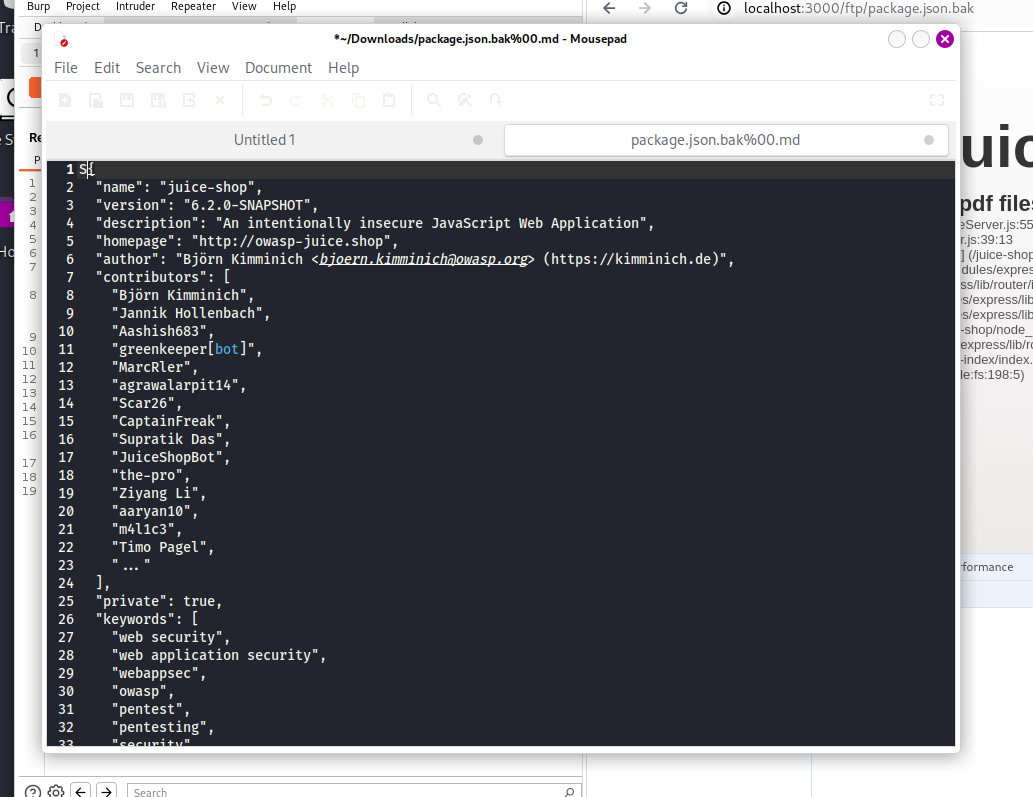
This exercise demonstrates how null byte injection vulnerabilities can be exploited to access and download files that are not intended to be publicly accessible. In this case, the vulnerability allowed me to retrieve the package.json.bak file, which could contain sensitive configuration data, as well as a hidden easter egg. This highlights the importance of proper input validation, secure file access controls, and understanding the potential impact of null byte injection attacks on web applications.

**To prevent null byte injection vulnerabilities:**

* **Input Validation:** Rigorously validate and sanitize user input, especially file paths.
* **Parameterization (Prepared Statements):** Use parameterized queries or prepared statements when interacting with databases.
* **Proper String Handling:** In C-based languages, ensure functions are aware of null bytes and don't rely solely on null byte termination for string validation.
* **Web Application Firewalls (WAFs):** Deploy a WAF to help detect and block potential null byte injection attacks.

**(eastere.gg file)**

**(All decoding and decryption)**



**(Package.json.bak file)**