Sensitive Data Exposure and Poison Null Byte Attack

By: Ryan, Pradeep, and Joey

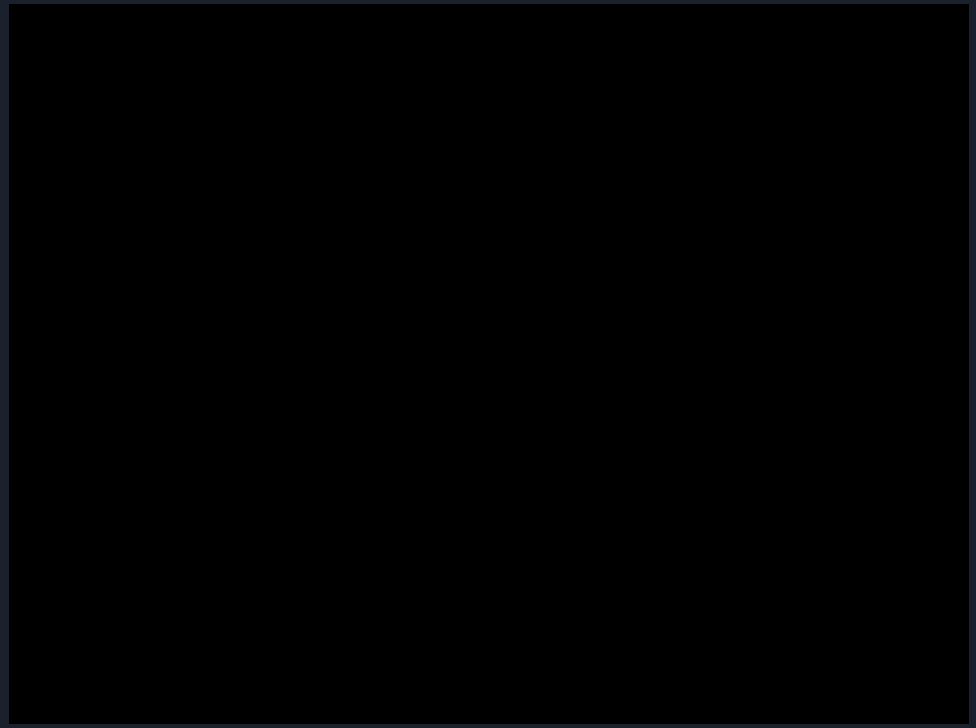
Technical Background

- Demonstrate web app vulnerabilities
- Security concepts
  - Sensitive data exposure
  - Exposed directories
  - URL parameter manipulation
  - URL encoding (poison null byte)
- Networking concepts
  - OWASP Juice Shop web app
  - TryHackMe learning platform/VM
- OWASP Juice Shop is open source
- TryHackMe has many free scenarios
- Expand on existing concepts covered in class

Demonstration Preview

1. Access web app
2. Navigate to page with exposed directory
3. Manipulate URL to access directory
4. Try to download sensitive files
5. Use poison null byte to access restricted filetype

Demonstration



Impact & Mitigation - Exposed Directories

- Exposed directories in web application can allow attackers access potentially sensitive data in the web server's filesystem that isn't normally available through the web application
  - Impact
    - Sensitive Data Exposure
    - Future Attacks
  - Mitigation
    - Single Page Application (SPA) framework (eg. React, Angular)
    - Web server modules (eg. mod_rewrite, URLRewrite)
    - Index file requirement on critical directories

Impact & Mitigation - Poison Null Bytes

- Poison null byte injections can bypass mandatory appended file extensions, allowing access to restricted or sensitive data
  - Impact
    - Information about dependencies revealed
    - Application vulnerability exposure
  - Mitigation
    - Server-side input sanitization or validation
    - Web application firewall (WAF)
    - Null byte filter
    - Content Security Policy (CSP)
    - Web application vulnerability scanner (Nessus, Burp Scanner)

Demonstration Summary

- Navigated to the “About Us” section of the Juice Shop (web app) and noticed a directory was revealed to us through a hyperlink
- Manipulated the URL to bring us to the directory listed in the hyperlink
- Downloaded a sensitive file (acquisitions.md) that contained confidential information regarding future acquisitions
- Tried downloading a second file (package.json.bak) but was stopped by a 403 error
- Used a character bypass called “Poison Null Byte” to circumvent the 403 error by encoding it into the URL
- We did this by adding “%2500” after “.bak” and then adding “.md” to the very end, an allowed file type per server response (10.10.151.131/ftp/package.json.bak%2500.md)

Real World Example

- Microsoft Teams Vulnerability: Exploiting Null Bytes for Code Execution
  - August 2022
  - Microsoft Teams
  - Execute arbitrary code
  - Poison null byte
  - CVE-2022-34591
  - Significant security risk

References

- https://www.tryhackme.com
- https://defendtheweb.net/article/common-php-attacks-poison-null-byte
- http://hakipedia.com/index.php/Poison_Null_Byte
- https://stackoverflow.com/questions/3278359/filter-null-byte-in-request
- https://owasp.org/Top10/
- https://stackoverflow.com/questions/3278359/filter-null-byte-in-request
- https://superuser.com/questions/1139676/why-can-i-access-a-file-from-a-web-server-even-though-the-url-doesnt-exist
- https://www.reddit.com/r/tryhackme/comments/tqdjyk/owasp_juice_shop_question_about_poison_null_byte/
- https://cve.report/CVE-2021-34591