Penetration Test Report

Diary for travelers

Table of contents

Attack narrative

Mapping the application

Enumerating Content and Functionality

Browsing the entire content of the web application with Burp and OWASP ZAP as a local proxies.

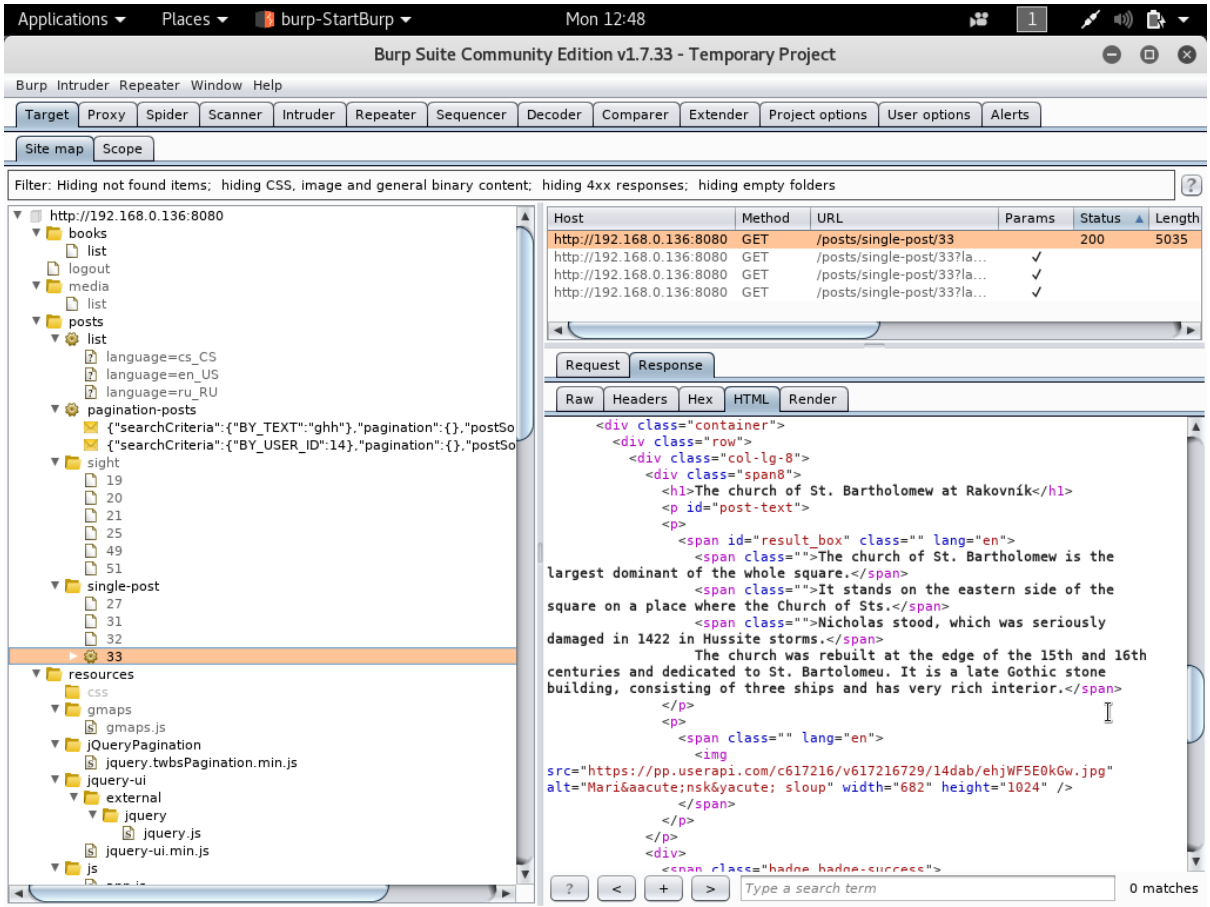

Figure 1: Basic structure of web application Diary for travelers

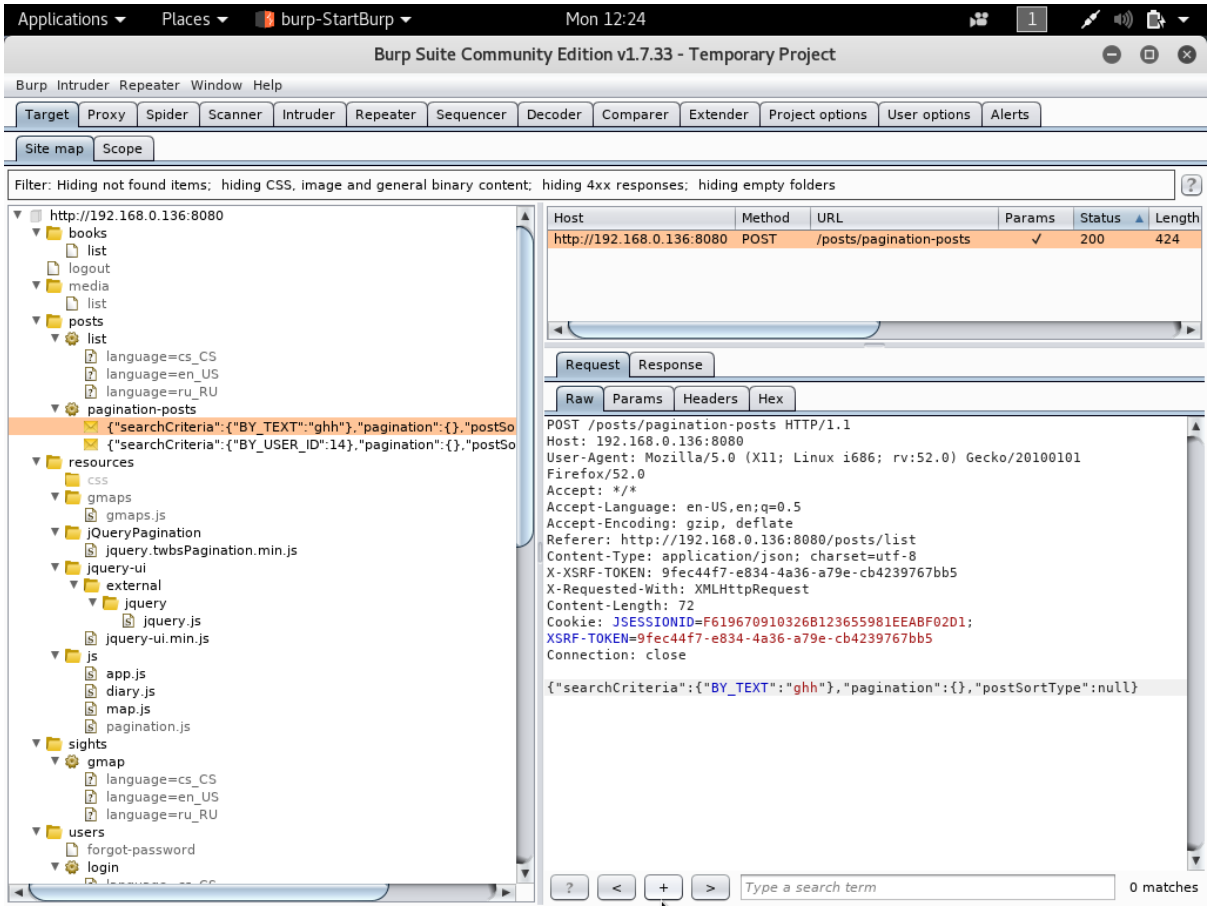

Figure 2: Discovering functional paths by Burp

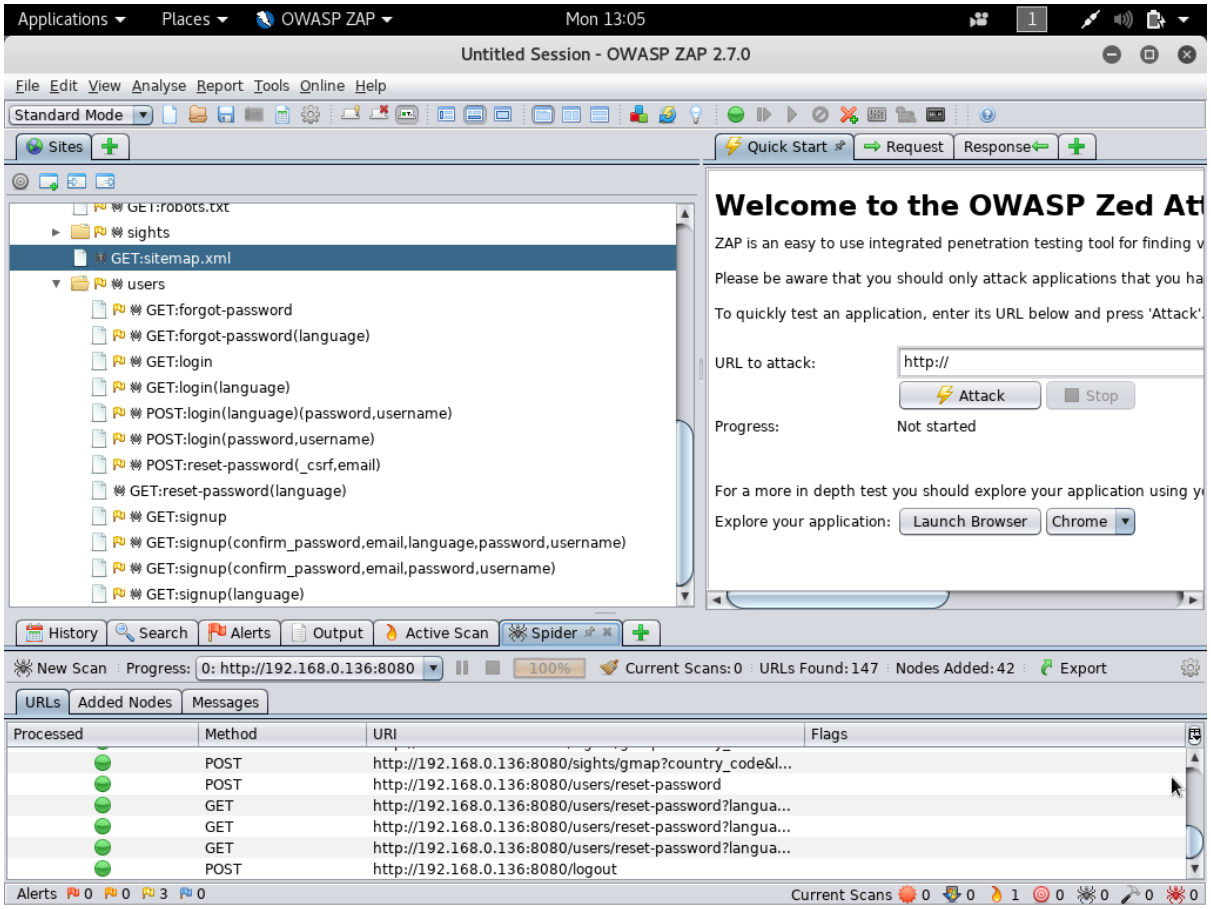

Figure 3: Discovering user's functionality by OWASP ZAP

Discovering the Web Server

Vulnerabilities may exist at the web server layer that enable you to discover content and functionality that are not linked within the web application itself [1].

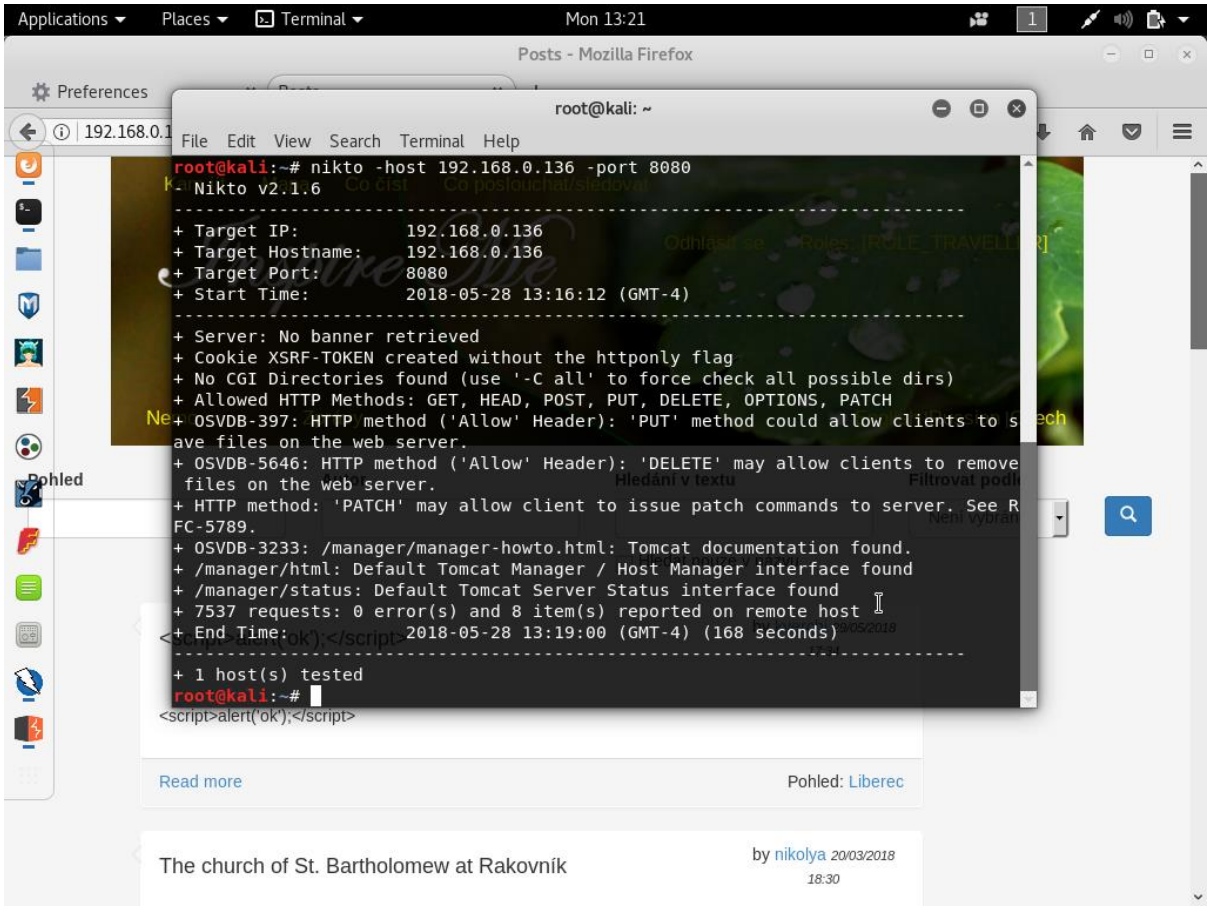

Figure 4: Using Nikto tool for scanning application's web server

The Diary for travelers is running on Tomcat server. Together with JSESSIONID cookies from web application scanning it is possible to make a conclusion that Diary for travelers is created by Java and deployed on Tomcat.
NMap scanning
Database, application file structure, hidden content, web server version

Discovering Hidden Content

Attack plan

After getting basic understanding of the structure, functionality and technologies of the web application Diary for travelers it's possible to create an attack plan:

- Client-side validation: if checks are replicated on the server side
- SQL injection
- Cross-site scripting
- Login functionality: username enumeration, weak passwords, ability to use brute force login credentials
- Session: predictable tokens, insecure handling of tokens, session hijacking, capture of sensitive data
- Informative error messages

- Weaknesses of application components: known vulnerabilities and configuration weaknesses

Attacking Authentication

Bad passwords

The authentication mechanism of web application Diary for travelers requires at least 6 symbols for password. But it allows users to use weak passwords, such as common dictionary words or names and the same as the username, as illustrated in Figure 1.

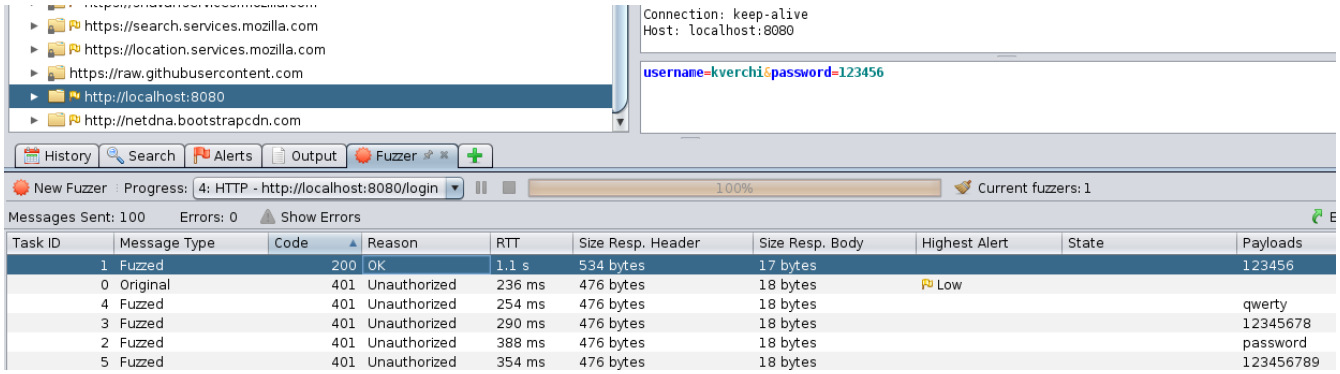

Figure 5: An account with weak password

Solving bad passwords issue

The application need to use strong credentials. Password must contain:

- alphabetic, numeric, and typographic characters
- the appearance of both uppercase and lowercase characters
- the avoidance of dictionary words, names, and other common passwords
- preventing a password from being set to the username
- preventing a similarity or match with previously set passwords [1]

Brute-Forcible Login

As it's shown in Figure 2, application allows an attacker to make repeated login attempts with different passwords until he guesses the correct one.

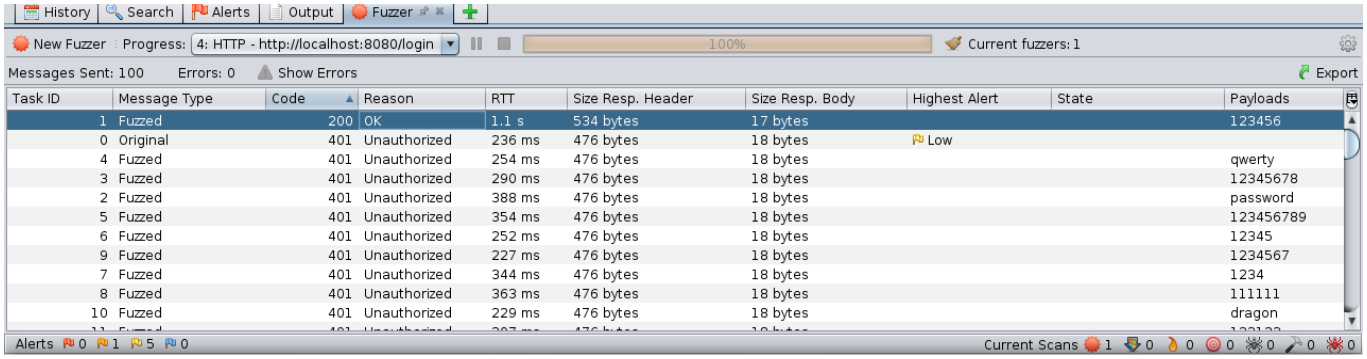

Figure 6: Unlimited attempts to guess a password

Solving Brute-Forcible Login issue

Web application must implement policy of temporary account suspension to prevent brute-force attacks. In addition, an application can be protected through the use of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) challenges on every page that may be a target for brute-force attacks [1].

Conclusion

Risk rating

Recommendations