Assignment 5

Exploring Security Testing Tools

Atharva Vaidya
121942024

Contents

Wapiti

Wapiti allows you to audit the security of your web applications. It performs "black-box" scans; i.e., it does not study the source code of the application but will scans the webpages of the deployed webapp, looking for scripts and forms where it can inject data. Once it gets this list, Wapiti acts like a fuzzer, injecting payloads to see if a script is vulnerable.

```
wapiti -u http://www.moodle.coep.org.in/moodle/ -d 2 -v 2 -o ./moodle
```

Options:

- **-n**: Define a limit of urls to read with the same pattern (prevent endless loops), here limit to 10.
- **-b**: Set the scope of the scan, here we analyze all the links to the pages which are in the same domain as the URL passed.
- **-u**: Use color to highlight vulnerables parameters in output.
- **-v**: Define verbosity level, here we print each url.
- **-f**: Define report type, here we choose HTML format.
- **-o**: Define report destination, in our case it must be a directory because we choose HTML format.

Attack modules used by WAPITI

- **backup**: This module search backup of scripts on the server.
- **blindsql**: Time-based blind sql scanner.
- **crlf**: Search for CR/LF injection in HTTP headers.
- **exec**: Module used to detect command execution vulnerabilities.
- **file**: Search for include()/fread() and other file handling vulns.
- **htaccess**: Try to bypass weak htaccess configurations.
- **nikto**: Use a Nikto database to search for potentially dangerous files.
- **permanentxss**: Look for permanent XSS.
- **sql**: Standard error-based SQL injection scanner.
- **xss**: Module for XSS detection.
- **buster**: Module for a file and directory buster attack - checking for "bad" files.

- **shellshock**: Module for Shellshock bug detection.

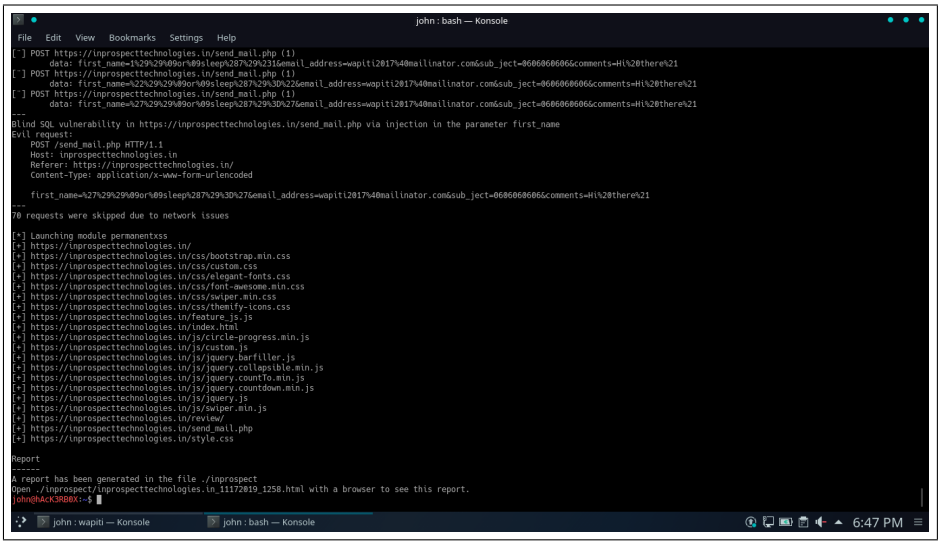


Figure 1: Target 1 : Inprospect Technologies

Figure 2: Vulnerability Report for Inprospect Technologies

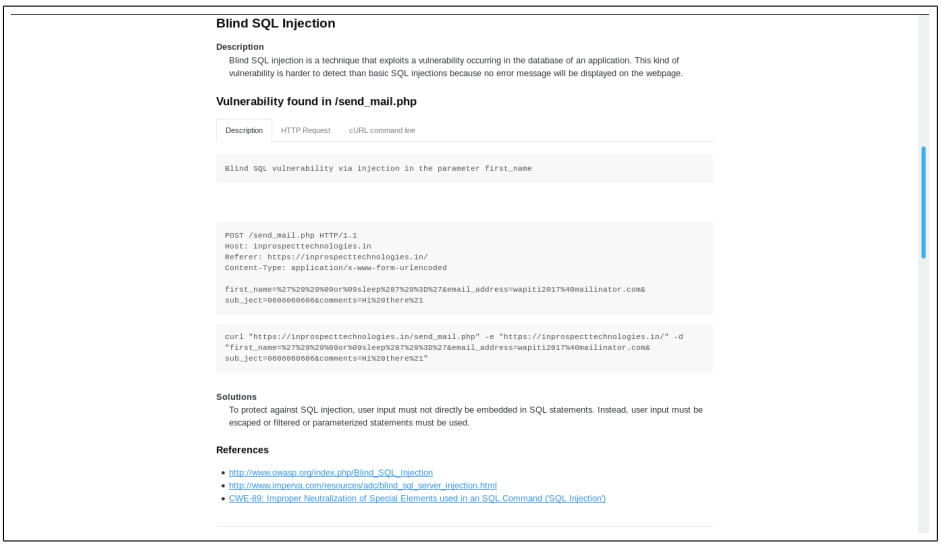


Figure 3: Vulnerability 1 : Blind SQL Injection

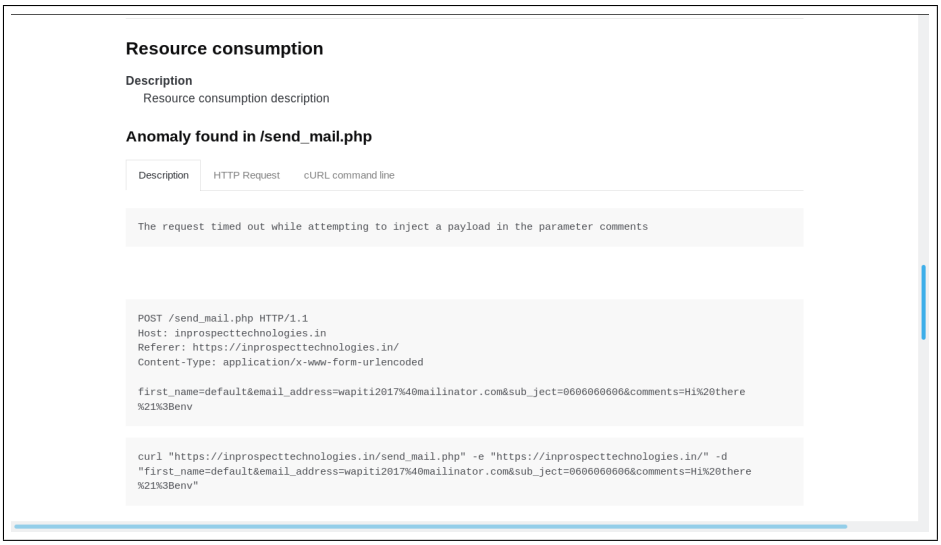


Figure 4: Vulnerability 2 : Anomaly - Request Timeout

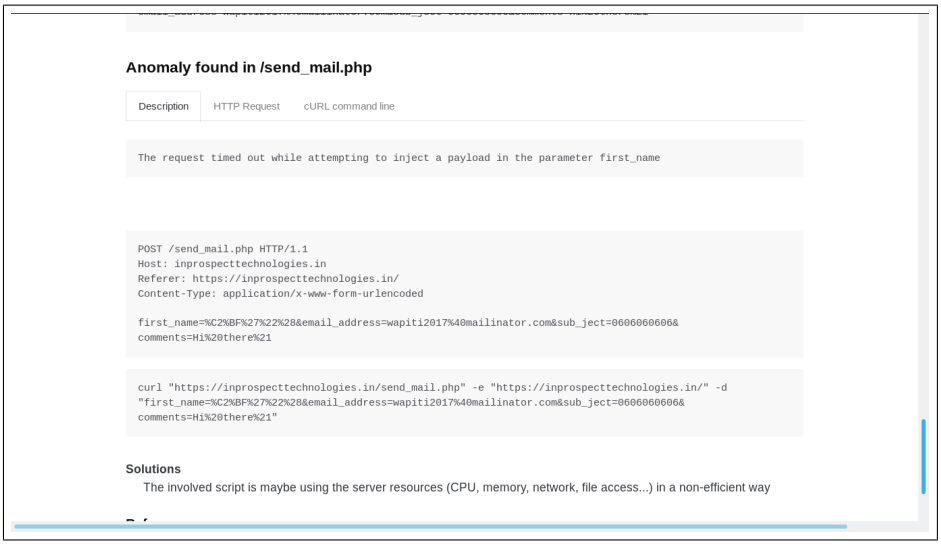


Figure 5: Vulnerability 3 : Anomaly - Request Timeout

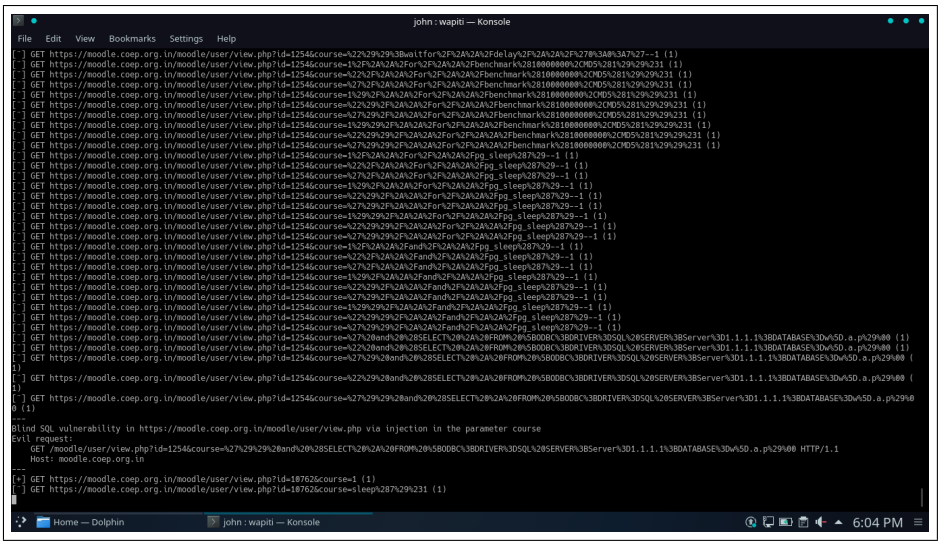
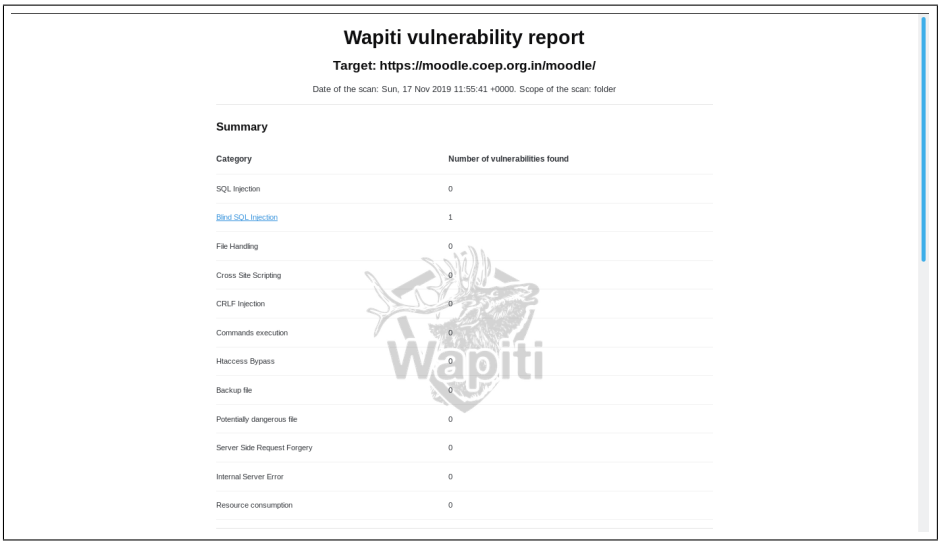


Figure 6: Target 2 : COEP Moodle



Wapiti vulnerability report

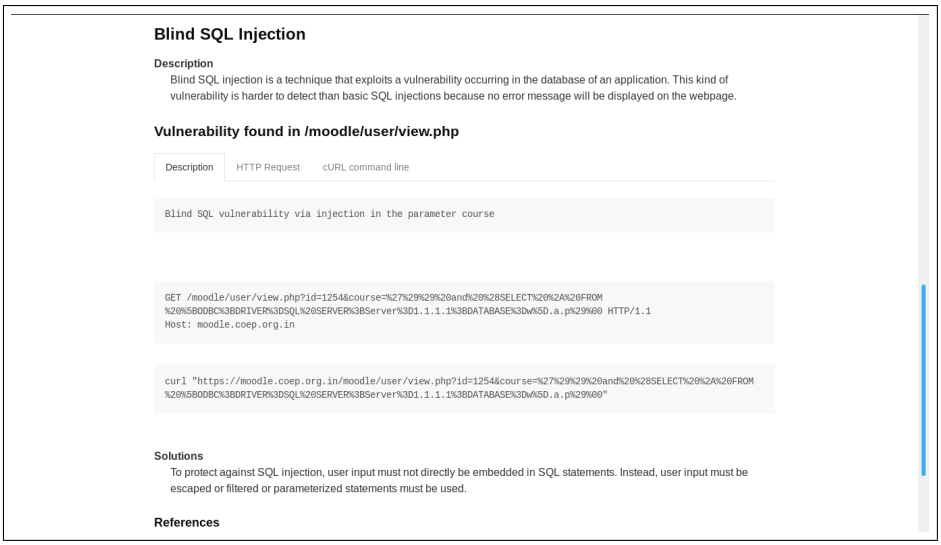
Target: https://moodle.coep.org.in/moodle/

Date of the scan: Sun, 17 Nov 2019 11:55:41 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
SQL Injection	0
Blind SQL Injection	1
File Handling	0
Cross Site Scripting	0
CRLF Injection	0
Commands execution	0
Htaccess Bypass	0
Backup file	0
Potentially dangerous file	0
Server Side Request Forgery	0
Internal Server Error	0
Resource consumption	0

Figure 7: Vulnerability Report for COEP Moodle



Blind SQL Injection

Description

Blind SQL injection is a technique that exploits a vulnerability occurring in the database of an application. This kind of vulnerability is harder to detect than basic SQL injections because no error message will be displayed on the webpage.

Vulnerability found in /moodle/user/view.php

Description | HTTP Request | cURL command line

```
Blind SQL vulnerability via injection in the parameter course
```

```
GET /moodle/user/view.php?id=1254&course=%27%29%29%20and%20%28SELECT%20%2A%20FROM
%20%5BODBC%3BDRIVER%3DSQL%20SERVER%3BServer%3D1.1.1.1%3BDATABASE%3Dw%5D.a.p%29%00 HTTP/1.1
Host: moodle.coep.org.in
```

```
curl "https://moodle.coep.org.in/moodle/user/view.php?id=1254&course=%27%29%29%20and%20%28SELECT%20%2A%20FROM
%20%5BODBC%3BDRIVER%3DSQL%20SERVER%3BServer%3D1.1.1.1%3BDATABASE%3Dw%5D.a.p%29%00"
```

Solutions

To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

References

Figure 8: Vulnerablitiy 1: Blind SQL Injection

Zed Attack Proxy

Zed Attack Proxy (ZAP) is a free, open-source penetration testing tool being maintained under the umbrella of the Open Web Application Security Project (OWASP). ZAP is designed specifically for testing web applications and is both flexible and extensible. At its core, ZAP is what is known as a "man-in-the-middle proxy." It stands between the tester's browser and the web application so that it can intercept and inspect messages sent between browser and web application, modify the contents if needed, and then forward those packets on to the destination. It can be used as a stand-alone application, and as a daemon process.

If there is another network proxy already in use, as in many corporate environments, ZAP can be configured to connect to that proxy.

ZAP provides functionality for a range of skill levels – from developers, to testers new to security testing, to security testing specialists. ZAP has versions for each major OS and Docker, so you are not tied to a single OS. Additional functionality is freely available from a variety of add-ons in the ZAP Marketplace, accessible from within the ZAP client.

Installation

Go to https://github.com/zaproxy/zaproxy/wiki/Downloads , download the installer and run the installer

Wfuzz

Wfuzz has been created to facilitate the task in web applications assessments and it is based on a simple concept: it replaces any reference to the FUZZ keyword by the value of a given payload.

A payload in Wfuzz is a source of data.

This simple concept allows any input to be injected in any field of an HTTP request, allowing to perform complex web security attacks in different web application components such as: parameters, authentication, forms, directories/files, headers, etc.

Wfuzz is more than a web content scanner:

- Wfuzz could help you to secure your web applications by finding and exploiting web application vulnerabilities. Wfuzz's web application vulnerability scanner is supported by plugins.
- Wfuzz is a completely modular framework and makes it easy for even the newest of Python developers to contribute. Building plugins is simple and takes little more than a few minutes.
- Wfuzz exposes a simple language interface to the previous HTTP requests/responses performed using Wfuzz or other tools, such as Burp. This allows you to perform manual and semi-automatic tests with full context and understanding of your actions, without relying on a web application scanner underlying implementation.

Installation

To install, simply type:

```
pip install wfuzz
```

It was created to facilitate the task in web applications assessments, it's a tool by pentesters for pentesters ;)

W3af

w3af - Web Application Attack and Audit Framework

w3af is an open source web application security scanner which helps developers and penetration testers identify and exploit vulnerabilities in their web applications.

It is fully written in Python.

The scanner is able to identify 200+ vulnerabilities, including Cross-Site Scripting, SQL injection and OS commanding.

The w3af framework has both a graphical and console user interface, in less than 5 clicks and using the predefined profiles it is possible to audit the security of your web application.

Installation

Download from source, move to the w3af Directory, and run the executable.

```
git clone https://github.com/andresriancho/w3af.git

cd w3af

./w3af_gui
```

SQLMap

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester, and a broad range of switches including database fingerprinting, over data fetching from the database, accessing the underlying file system, and executing commands on the operating system via out-of-band connections.

Installation

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

```
python sqlmap.py -h
```

SonarQube

SonarQube collects and analyzes source code, measuring quality, detects security vulnerabilities and providing reports for your projects. It supports more than 20 programming languages. It combines static and dynamic analysis tools and enables quality to be measured continuously over time. Everything that affects our code base, from minor styling details to critical design errors, is inspected and evaluated by SonarQube, thereby enabling developers to access and track code analysis data ranging from styling errors, potential bugs, and code defects to design inefficiencies, code duplication, lack of test coverage, and excess complexity. The Sonar platform analyzes source code from different aspects and hence it drills down to your code layer by layer, moving from the module level down to the class level. At each level, SonarQube produces metric values and statistics, revealing problematic areas in the source that require inspection or improvement.

Nogotofail

Nogotofail is a network security testing tool designed to help developers and security researchers spot and fix weak TLS/SSL connections and sensitive cleartext traffic on devices and applications in a flexible, scalable, powerful way. It includes testing for common SSL certificate verification issues, HTTPS and TLS/SSL library bugs, SSL and STARTTLS stripping issues, cleartext issues, and more.

Iron Wasp

IronWASP (Iron Web Application Advanced Security testing Platform) is an open source tool used for web application vulnerability testing. It is designed in such a way that users having the right knowledge can create their own scanners using this as a framework. IronWASP is built using Python and Ruby and users having knowledge of them would be able to make full use of the platform. However, IronWASP provides with a lot of features are simple to understand.

Few features of IronWASP are as follow:

- Its GUI based
- Supports recording Login sequence
- Can generate reports in HTML as well as RTF formats
- Scans for over 25 different web vulnerabilities
- False positive detection support
- False negative detection support
- Built-in scripting engine that supports Python and Ruby
- Extensible via plug-ins or modules in Python, Ruby, or VB .NET
- It is bundled with a number of modules built by independent security researchers such as WiHawk, IronSAP, CSRF PoC Generator, XMLChor, etc.

Grabber

Grabber is a black box web application vulnerability scanner that looks for SQL Injection, Blind SQL injection, XSS vulnerability and File include injection.

The tool aims to be quite generic, and can work with any kind of web application regardless of the server side programming language. The tool is designed to be a simple, efficient way to detect vulnerabilities.

The tool uses a modular and extensible design having individual module for each kind of vulnerability detection. You can also extend the attack list by adding them to the XML files in specified format.

Grabber is written entirely in Python and requires additional python modules as dependencies such as BeautifulSoup and PyXML for web scrapping and parsing XML.

How it works

Grabber first runs the spider to scan the given web application for all the pages. It parses the HTML content using Beautiful soup to identify links in href, form actions, javascript etc. and builds an XML index file inside local/spidersite.xml.

Once it has scanned for all the links it makes a GET call to every link in the index to identify all the variable parameters in GET queries, HTML form fields and generates a database of all the the variable parameters found against the specific URL along with request methods available.

Based on the given attack parameters passed, Grabber then runs the investigation for each of the attacks by manipulating the parameter values using attack instances mentioned in the XML files. It then performs GET or POST request on the URLs in the database. The returned HTML output from the server is scanned for Keywords that indicate injection was a successful or failure.

Grabber generates a live and interactive HTML report that displays the current status of the tool, all the scanned URLs and vulnerabilities detected in an intuitive manner.

Installation

Download from source, move to the Grabber Directory, and run the executable.

```
git clone https://github.com/avaneeshd/grabber
```

```
cd Grabber

python grabber.py --url <APPLICATION_URL>
```

Arachni

Arachni is a feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of web applications.

It is smart, it trains itself by monitoring and learning from the web application's behavior during the scan process and is able to perform meta-analysis using a number of factors in order to correctly assess the trustworthiness of results and intelligently identify (or avoid) false-positives.

Moreover, due to its integrated browser environment, it can also audit and inspect client-side code, as well as support highly complicated web applications which make heavy use of technologies such as JavaScript, HTML5, DOM manipulation and AJAX.

Finally, it is versatile enough to cover a great deal of use cases, ranging from a simple command line scanner utility, to a global high performance grid of scanners, to a Ruby library allowing for scripted audits, to a multi-user multi-scan web collaboration platform.