IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44SE/2024/4445424/
Vol. 2024 (24), Article ID 4445424

# WEB Application Security Testing: A Practical Example of Laboratory Setup*

Piotr KONTOWICZ

Poznan University of Technology, Faculty of Computing and Telecommunications,
Piotrowo 3, 60-965 Poznan, Poland

Correspondence should be addressed to: Piotr KONTOWICZ, piotr.kontowicz@put.poznan.pl

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

## Abstract

This document outlines a comprehensive framework for web application security training, using a structured, isolated environment to cover both basic and advanced vulnerability classes. Leveraging VirtualBox, it demonstrates a virtual lab setup with a tester machine and a vulnerable web application, focusing on hands-on exercises like Capture The Flag (CTF) challenges to help participants master prevalent security issues. Aligned with real-world scenarios through the inclusion of OWASP Top Ten vulnerabilities, this setup ensures practical, applicable learning. Future enhancements will introduce remote server-based environments, enabling scalable, individualized instances to prevent interference and optimize learning. Designed for academic and professional use, this framework provides a robust foundation for developing security expertise in a realistic, controlled setting.

**Keywords:** Web Application Security, Security Testing Environment, Penetration Testing

## Introduction

Nowadays, lots of utilities and programs are implemented as web services. Such services often handle and store confidential data of users. Therefore, these services need to be audited for security somewhat regularly. The popularity of implementing utilities via web services from the client point of view follows this fact: neither the installation nor any operating system configuration changes are typically required for the client of web services - namely a web browser. Large organizations will therefore not necessarily have to invest hugely in a large technical department to support such deployment and configuration of software.

Web applications help service providers, too, because there is less of a chance that somebody can steal the software or use and distribute unauthorized pirated versions of it. Still, users of these applications expect the providers to be extremely attentive to security and data privacy, which again enhances the demand for intensive security testing. Consequently, it is important not only to understand the very process of security testing, but also to reveal potential vulnerabilities at the stage of application development. An effective way to master these skills has to do with the Capture The Flag challenges: in such exercises, the participants reveal flags through exploiting intentionally embedded security flaws. The proceeding article shall outline a selection of solutions that may be used to create a controlled environment in order to learn about common security vulnerabilities in web applications, designed as part of lab exercises for classroom instruction.

## Designing a Virtual Environment for Targeted Security Assessments

To establish an isolated and comprehensive environment for practical security testing, VirtualBox[1] is utilized for managing virtual machines, though alternative virtualization software such as VMware or Hyper-V can serve

**Cite this Article as:** Piotr KONTOWICZ, Vol. 2024 (24) "WEB Application Security Testing: A Practical Example of Laboratory Setup " Communications of International Proceedings, Vol. 2024 (24), Article ID 4445424, https://doi.org/10.5171/2024.4445424

similar functions. Virtualization is selected for its ability to provide complete isolation of the test environment from the host system. Within this setup, two primary virtual machines are configured to support targeted exercises in web application security:

1. Tester Machine**:** typically, this virtual machine runs Kali Linux[2], functioning as the primary workstation for security assessments. Kali Linux, an open-source, Debian-based OS, is optimized for penetration testing through its extensive suite of pre-installed tools, including network scanners, exploit frameworks, and forensic utilities. Alternatives, such as Parrot OS or BlackArch, may also be suitable, provided they support the necessary tools. The tester machine should feature two network interfaces: one enabling internet access for software updates and tool downloads, and a second that connects to an isolated virtual network.
2. Vulnerable Web Application Machine: this machine hosts a deliberately flawed web application to enable users to explore common security vulnerabilities. Ubuntu is often employed due to its stability and broad software support, though any OS capable of running vulnerable applications can be used, offering instructional flexibility. For security purposes, this machine should operate with a single network interface exclusively linked to the internal virtual network, thereby restricting external access. Such a setup prevents risks to the host or broader network and avoids unintended data exposure by isolating the vulnerable machine from the internet. This configuration mirrors best practices in enterprise security, where vulnerable systems are isolated from production networks to limit unauthorized access.

Together, these virtual machines create a secure, isolated test bed for hands-on practice in web application security. This setup is particularly effective for Capture The Flag (CTF) challenges, which allow participants to gain practical experience by exploiting vulnerabilities to uncover hidden flags.

A primary advantage of this localized environment is the isolation of the application from testers, who lack source code access, thereby simulating real-world conditions in which attackers operate without insight into application internals. Although similar isolation could be achieved using a Virtual Private Server (VPS), localized setups offer notable benefits.

For instance, multiple testers on a shared VPS may inadvertently interfere with one another, potentially altering the application's state and leading to inconsistent behaviors. Additionally, many hosting providers impose restrictions on actions like privilege escalation, risking account suspension if tested on a VPS. By hosting the vulnerable application within a locally controlled environment, testers are able to perform comprehensive security testing without such limitations. Finally, local environments enhance testing efficiency in scenarios involving high-frequency packet transmission. Reduced latency in a local setup improves performance, making it particularly suitable for time-sensitive exercises, such as testing session management or race conditions. In summary, a local environment offers distinct advantages over a shared VPS by providing isolation, responsiveness, and compliance, establishing an ideal framework for learning foundational and advanced concepts in web application security.

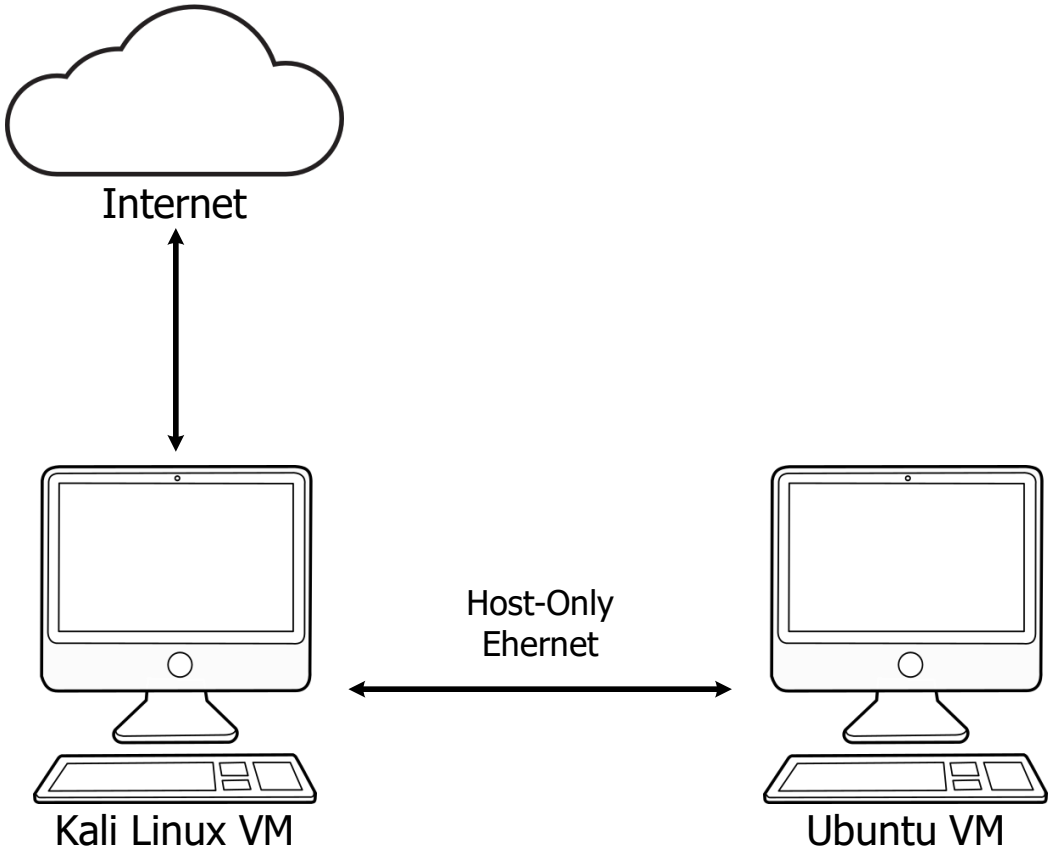


**Fig 1. Example connection setup**

## Designing Tailored Security Testing Environments

When preparing resources for security testing exercises, instructors may consider utilizing pre-configured virtual machines or applications designed by industry experts and reputable organizations, such as the Open Worldwide Application Security Project (OWASP)[3]. A prominent example is the OWASP Juice Shop[4], which offers participants a wide range of vulnerabilities to explore. This approach enables participants, especially beginners, to find solutions or seek hints online, which can serve as a valuable learning aid. For those new to security testing, accessible hints help reduce frustration and accelerate the learning process, making the experience more approachable and engaging.

To further deepen participants' understanding beyond widely available applications, it is beneficial to incorporate exercises that involve custom applications or virtual machines that are not accessible to the public. Ideally, these applications should be developed in-house, with specific security vulnerabilities intentionally embedded. Custom applications provide the added advantage of implementing unique flag generation mechanisms, which allows each participant to receive a distinct flag upon successfully completing an exercise. This method facilitates straightforward task verification.

One practical approach to unique flag generation involves incorporating a login feature where participants must enter personalized data to begin the exercise. By using participant-specific information alongside additional contextual data, unique flags can be created. For instance, common data inputs might include the participant’s name and the current date. These values are then combined into a single string, hashed using a chosen algorithm, and a designated portion of the hash output is used to generate a concise, unique flag.

This customized approach to security exercises allows participants to gain a deeper understanding of core security principles while ensuring they follow all necessary learning steps. Furthermore, the unique flags generated for each participant enable accurate assessment and verification, which enhances the overall educational impact of the hands-on learning experience.

## Typical Vulnerabilities in WEB Application

In developing effective security exercises, it is essential to focus on vulnerabilities that are both prevalent and relevant to real-world production environments. The OWASP Top Ten[3], a periodically updated list of the most common vulnerability and security error classes, serves as an excellent guide for this purpose. The latest list from 2021 includes the following key vulnerability classes:

1. **Broken Access Control**: this encompasses errors in authentication and permission configurations, potentially allowing unauthorized access, privilege escalation, and data modification or deletion within the system - making it one of the most critical vulnerability classes.
2. **Cryptographic Failures**: this category includes weaknesses in protecting data at rest or in transit. The use of outdated or unsuitable cryptographic algorithms, such as SHA1 or MD5 for password storage, also falls within this category.
3. **Injection**: injection vulnerabilities, including SQL Injection and Cross-Site Scripting (XSS), are among the most severe. XSS enables attackers to inject JavaScript into a website, potentially allowing them to read displayed information, execute user actions, or steal session data. SQL Injection targets an application's database through improperly handled input data, allowing attackers to bypass authentication or modify data.
4. **Insecure Design**: this relates to flaws introduced during the design phase of application development. While design flaws differ from implementation errors, even a well-designed architecture may suffer from implementation flaws that undermine security.
5. **Security Misconfiguration**: deploying an application requires thorough server configuration, including disabling unnecessary services, restricting open ports, and limiting user permissions. Misconfigurations like leaving redundant accounts or failing to apply the principle of least privilege pose substantial risks.
6. **Vulnerable and Outdated Components**: frequent updates to libraries and dependencies are essential to prevent vulnerabilities associated with third-party components used in software. Regular verification of these components is crucial to ensure they do not introduce new threats.
7. **Identification and Authentication Failures**: exposing session IDs in URLs or headers and failing to protect against brute-force attacks contribute to vulnerabilities in user identification and authentication processes.
8. **Software and Data Integrity Failures**: security risks arise if code from untrusted sources is integrated into applications, as attackers may compromise these sources. An example is including JavaScript from an external, potentially untrusted server.
9. **Security Logging and Monitoring Failures**: effective security management requires logging critical events, such as login attempts and system operations. Without proper monitoring, providers lack visibility into application activity, reducing their capacity to respond to potential threats.
10. **Server-Side Request Forgery (SSRF)**: SSRF vulnerabilities allow an application's server to initiate unauthorized network communication, which may target internal network resources or sensitive external assets, depending on server configuration.

For a cohesive educational program, exercises can be structured according to these vulnerability classes, allowing participants to gradually build their understanding. In initial sessions, simpler tasks can introduce fundamental vulnerabilities, enabling participants to familiarize themselves with basic concepts and build confidence. Later, exercises may involve multi-step tasks, requiring participants to combine different vulnerability classes to achieve their objectives.

By following this progression, the training ensures that participants not only learn to identify vulnerabilities but also develop the skills necessary to apply these concepts in complex, real-world scenarios.

## Future Work

Once additional resources are available, testing environments will shift to a centralized remote server. Participants will log into a shared system, where they will have the option to launch individualized instances of the application. After setting up an instance, each participant will receive a unique URL to access it directly. This solution removes the need for downloading virtual machine images and ensures that each participant works within their own isolated application. As a result, testing activities will be free from cross-interference, maintaining consistent test outcomes and stable application performance during active sessions.

## Acknowledgements

This research was funded by the Polish Ministry of Science and Higher Education (No. 0313/SBAD/1310).

## Bibliography


- VirtualBox User Manual. Oracle, 2023. Available at: https://www.virtualbox.org/manual/UserManual.html
- Kottler, Phillip, et al. Kali Linux Penetration Testing Bible. Wiley, 2021.
- OWASP Foundation. OWASP Top Ten Web Application Security Risks – 2021. Open Web Application Security Project (OWASP), 2021. Available at: https://owasp.org/www-project-top-ten/
- OWASP Foundation. OWASP Juice Shop: A Modern Web Application for Security Training. OWASP Project Documentation. Available at: https://owasp.org/www-project-juice-shop/