Assessment Report

AdServeX - Portal for the Management and Placement of Online Advertisements

Prepared for
Mustermann GmbH

Date: February 3, 2025
Version: 1.0
Status: Final

Prepared by
Wavestone AG

Author: Gabriel Duschl
Email: gabriel.duschl@wavestone.eu

Wavestone AG
Hohlstrasse 614
8048 Zürich

Document History

Version	Status	Date	Author
0.8	Draft	January 29, 2025	Gabriel Duschl
0.9	QA	January 31, 2025	Alexander Strassheim
1.0	Final	February 3, 2025	Gabriel Duschl

Confidentiality Disclaimer

This document focuses exclusively on the technical security aspects identified during the assessment. This report is strictly confidential. It contains technical details about security vulnerabilities that could be exploited if accessed by unauthorized individuals. Therefore, access to this report must be restricted. It may not be shared, published, or distributed without the express written consent of the client.

Table of Contents

1 Management Summary

The results in this report are based on the analysis that was performed from January 20 to January 31, 2025 . This report provides an indication of the security posture of the in-scope systems as of the last day of the testing period.

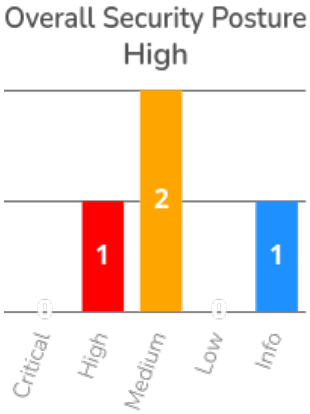
Assessment Scope

This section outlines the scope of the assessment, specifying the targets and key areas of the penetration test.

Target	Description
example.com	As part of the penetration test, the Example WebApp, a publicly accessible web application, was checked for security vulnerabilities. The application offers authentication functions, user management and API interfaces. The aim of the test was to identify and evaluate potential vulnerabilities in the web architecture, access control and data handling.

Key Observations and Recommdations

In the course of this penetration test 1 High and 2 Medium vulnerabilities were identified. These vulnerabilities could lead to a compromise of the confidentiality, integrity, and availability of the data held within the application. 1 Info issue(s) with a CVSS score of **0.0** were identified that do not pose an immediate threat. The recommended actions should still be implemented to further improve the overall security posture of the application.



The illustrated figure on the left indicate the security posture of the application. Based on the highest identified severity rating, the exposure level at the time of testing is considered High.

For further details and a more technical description of all identified observations and our recommendations, please refer to the Detailed Observations section of this report.

A summary of the key observations identified and actions to mitigate or to reduce the highest identified risk can be seen below:

Persistent Cross-Site Scripting (PXSS): Cross-Site-Scripting
Recommendation: User input should be validated

2 Observation Summary

The following table gives an overview of our observations from the security assessment. For detailed descriptions of each observation, please refer to Section 5 Detailed Observations of this document.

ID	Name	CVSS	Severity
H1	Persistent Cross-Site Scripting (PXSS)	7.2	High
M1	HTTP Strict Transport Security (HSTS)	5.4	Medium
M2	TLSv1.0 - BEAST	5.3	Medium
I1	Port Scan	0.0	Info

3 Project Dashboard

The detailed scope of the test performed from January 20 to January 31, 2025 , is as follows:

Target	Approach	Environment
example.com	Gray-Box	Staging

The following details provide additional context and considerations that were included in the scope of the penetration test, covering all relevant components and aspects of the project:

- https://example.com
- https://api.example.com

Project Timeline

The project was carried out according to the following timeline. Efforts are stated in person days (PD):

Date or Period	Description	Effort
January 20 to January 24, 2025	Web Application	5 PD
January 27 to January 29, 2025	Web Service	3 PD
January 30 to January 31, 2025	Documentation	2 PD

Contact Details

The following contact details were provided:

Role	Name	Email
Manager	Max Musterman	max.musterman@company.com
Pentester	Gabriel Duschl	gabriel.duschl@wavestone.eu

Test User Accounts

The following test users and assigned roles were provided or created within the assessment. Please make sure to disable or remove these accounts after testing:

User	Description
test-user	Test user for web application
test-admin	Test administrative user for web application

Testing Tools

The following tools were used:

Tool Name	Short Description
Burp Suite Professional	SSL Proxy
Nessus Professional	Vulnerability Scanner
Nmap Port Scanner	Network port scanner
Nikto	Check server configuration
dirsearch	Check pathes
ssh-audit	Analyse SSH configuration
sslscan	Analyse SSL configuration
testssl.sh	Analyse SSL configuration

Vulnerability Scoring and Management

CVSS: The Common Vulnerability Scoring System (CVSS) is used to determine the severity of discovered vulnerabilities. Specifically, this report provides the CVSS Base Score for each discovered issue, along with its CVSS Vector String. The numerical Base Score is also translated into a qualitative representation (such as low, medium, high, and critical) according to the CVSS specification. For further information on CVSS please consult the documentation available at https://www.first.org/cvss/

Wave Score: Wavestone utilizes the Common Vulnerability Scoring System (CVSS) as the foundation for vulnerability assessment but enhances it with the proprietary Wave Score to provide a more precise and context-aware risk evaluation. While the CVSS Base Score offers standardized ratings, the Wave Score extends this methodology by factoring in additional elements. These factors include the potential for a vulnerability to enable or exacerbate other security issues, the criticality of the affected application and the data it processes or exposes, and the broader operational impact on the organization's systems or security posture. By incorporating these dimensions, the Wave Score goes beyond numerical scoring to deliver a tailored assessment, providing a robust basis for prioritizing and addressing vulnerabilities in alignment with an organization's unique risks and requirements.

4 Best Practices for Secure Configuration

Misconfigurations are one of the most common attack vectors in cybersecurity. Even well-designed systems can become vulnerable due to improper settings or missing security measures.

This chapter provides an overview of essential security configurations. Each section includes a table summarizing common weaknesses, helping to establish a strong security foundation.

4.1 TLS Configuration

During the security test, the configuration was checked against supported protocol versions and other security-relevant settings or configurations. In the following tables, the symbol ❌ shows that there is a cryptographic issue or misconfiguration. The symbol ✔ means that no action is necessary.

	SSLv2	SSLv3	TLSv1.0	TLSv1.1	TLSv1.2	TLSv1.3	OCSP-Stapling	PFS	Certificate Trust	Expiry date
example.com:443	✔	✔	❌	✔	✔	✔	✔	✔	✔	2025-12-31

During the security test, the TLS configuration was checked against common cryptographical issues.

	Crime	Breach	Poodle	Sweet32	Freak	Drown	Logjam	Beast	Padding Oracle	RC4
example.com:443	✔	✔	✔	✔	✔	✔	✔	❌	✔	✔

4.2 Security Relevant HTTP Headers

In the course of the security test, the application was checked for missing security-relevant HTTP headers. Based on the OWASP Secure Headers Project[1], several HTTP headers should be added. In the following table, the symbol ❌ shows that the security-relevant HTTP header is missing or has potential for improvement. The symbol ✔ shows that no action is necessary.

HTTP Header	example.com	Comment
Strict-Transport-Security	❌	-
X-Frame-Options	❌	-
X-Content-Type-Options	✔	-
Content-Security-Policy	✔	-
X-Permitted-Cross-Domain-Policies	✔	-
Referrer-Policy	✔	-
Clear-Site-Data	✔	-
Cross-Origin-Embedder-Policy	✔	-
Cross-Origin-Opener-Policy	✔	-
Cross-Origin-Resource-Policy	✔	-
Cache-Control	✔	-

According to OWASP, the following HTTP headers are deprecated:

HTTP header	Comment
Feature-Policy	The HTTP header Feature-Policy is deprecated as it will be replaced by the HTTP header Permissions-Policy. As the specification of the Permissions-Policy is still under development, it is recommended to use the HTTP header Feature-Policy to restrict the use of browser features as long as the HTTP header Permissions-Policy is finished and supported by all current browsers.
Public Key Pinning Extension for HTTP	This HTTP header is currently not supported by any common browser. Additionally, for example through misconfiguration of the Header, Deep-Packet Inspection or through exchanging the TLS certificate, there is a potential to lock out users. .

1 https://owasp.org/www-project-secure-headers/

HTTP header	Comment
X-XSS-Protection	The HTTP header `X-XSS-Protection` should be regarded as deprecated by modern browsers. Furthermore, its use can introduce additional security issues on the client side. According to OWASP, it is recommended to set the header to `X-XSS-Protection: 0` in order to disable the XSS auditor. Instead, the HTTP header `Content-Security-Policy` should be used to handle JavaScript execution.
Expect-CT	This header is obsolete since June 2021. Since May 2018, new certificates are expected to support SCTs by default. Certificates issued before March 2018 were allowed to have a lifetime of 39 months and expired in June 2021.

The following output shows the responded HTTP headers for requests example.com:

```
HTTP/2 200
server: nginx
date: Tue, 04 Feb 2025 10:35:08 GMT
content-type: text/html; charset=UTF-8
x-hacker: If you're reading this, you should visit wpvip.com/careers and apply to
join the fun, mention this header.
x-powered-by: WordPress VIP <https://wpvip.com>
host-header: a9130478a60e5f9135f765b23f26593b
x-frame-options: SAMEORIGIN
content-security-policy: frame-ancestors 'self'
x-redirect-by: WordPress
location: https://example.com
x-rq: hhn1 111 254 443
cache-control: max-age=3600
x-cache: MISS
```

5 Detailed Observations

Subsequent pages of this report detail each of the identified observations, along with remediation advice.

5.1 Persistent Cross-Site Scripting (PXSS)

Target:	example.com/registration/	**Base Score:**	7.2
CVSS Vector:	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N	**Severity:**	High
Wave Details:	There is no deviation from the CVSS Score.	**Wave Score:**	7.2

Description

During the tests, we were able to identify a stored XSS vulnerability in the web application. Due to incorrect validation and encoding of data, we were able to inject malicious scripts into the web application and store them permanently.

Cross-site scripting (XSS) is a widespread web security vulnerability where malicious scripts can be injected into web applications due to insufficient validation or encoding of data. In XSS attacks, attackers embed JavaScript code into the content delivered by the vulnerable web application.

The aim of stored XSS attacks is to place script code on pages that are visited by other users. A simple visit to the affected subpage is enough for the script code to be executed in the victim's web browser.

During an attack, malicious scripts are injected into the web application by the attacker and stored and included in subsequent HTTP responses from the application. The malicious script is eventually executed in the victim's web browser and may be able to access cookies, session tokens or other sensitive information.

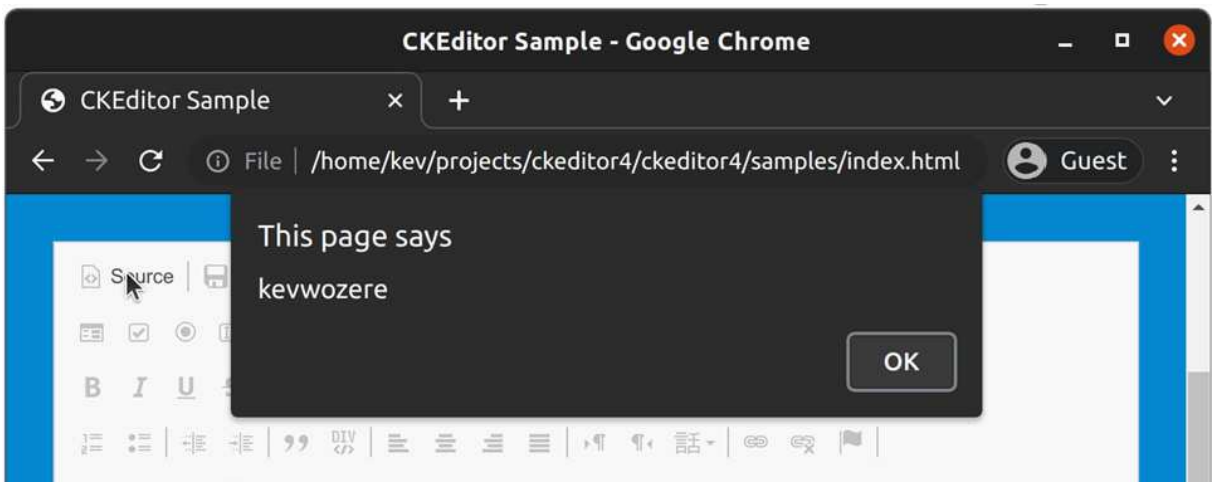


Figure 1 - Execution of JavaScript code in the browser

Impact

If the attack is successful, the attacker gains control over the functions and data of the web application in the victim's context. If the affected user has privileged access, an attacker may be able to gain complete control over the web application.

Recommendation

- Ensure that all processed data is filtered as rigorously as possible. Filtering and validation should be done based on expected and valid inputs.
- Data should be encoded before the web application includes it in HTTP responses. Encoding should be done contextually, that is, depending on where the web application inserts data in the HTML document, the appropriate encoding syntax must be considered.
- The HTTP headers `Content-Type` (e.g. `text/plain`) and `X-Content-Type-Options: nosniff` can be set for HTTP responses that do not contain HTML and JavaScript.
- We recommend to additionally use a Content Security Policy (CSP) to control which client-side scripts are allowed and which are forbidden.
- Detailed information and help on preventing XSS can be found in the linked Cross-Site Scripting Prevention Cheat Sheet from OWASP[2].

Steps to Reproduce

Within the web application, it was possible to insert Persistent Cross-Site Scripting (PXSS) in the login form, as shown in .

The following payload was used for this purpose:

```
<iframe// src=javaSCRIPT:alert('pXSS')>
```

When a user now visits the affected area, the JavaScript code is executed.

It was possible to find several places within the web application that are vulnerable to Persistent Cross-Site Scripting attacks.

[2] OWASP Cross Site Scripting Prevention Cheat Sheet

5.2 HTTP Strict Transport Security (HSTS)

Target:	example.com	**Base Score:**	5.4
CVSS Vector:	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N	**Severity:**	Medium
Wave Details:	There is no deviation from the CVSS Score.	**Wave Score:**	5.4

Description

A user who accesses the web application via the link http://example.com (i.e. unencrypted via HTTP) is not automatically redirected to the encrypted version at the URL https://example.com. This is due to the missing HTTP service on port 80/TCP and a consistently missing redirection to port 443/TCP. It is not possible to communicate with the web application via an unencrypted channel.

Wavestone supports this configuration, as man-in-the-middle attacks are made considerably more difficult by using SSL/TLS.

Impact

However, it may be possible for an attacker in a suitable network position to intercept a request to the web application via port 80/TCP and offer hosted content themselves via this service. An attacker can do this without the user ever noticing the attack.

Recommendation

HTTP Strict Transport Security (HSTS) should be implemented in order to impede such attacks. This means that the web server should send the HTTP header `Strict-Transport-Security` and a sensible `max-age` value to the browser. A suitable header could look like the following:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

When HSTS is used, the described attack can only be conducted against users that never have visited the web application before (within the time frame specified by the server). The validity time of the headers should typically be set to several months or even a few years.

Steps to Reproduce

The headers can be displayed with the following command:

```
$ curl -I -X GET https://example.com
```

5.3 TLSv1.0 - BEAST

Target:	example.com	**Base Score:**	5.3
CVSS Vector:	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:N/A:N	**Severity:**	Medium
Wave Details:	There is no deviation from the CVSS Score.	**Wave Score:**	5.3

Description

TLS version 1.0 is offered. This protocol version is deprecated. Throughout the service is susceptible to the BEAST (Browser Exploit Against SSL/TLS), for more information see (Link). BEAST relies on the way how the initialization vectors (IVs) are handled (when using CBC ciphers in combination with SSLv2, SSLv3 or TLSv1.0). This could lead to successful Chosen Plaintext Attacks (CPA) (Link).

In accordance with guidance from government agencies (e.g. NIST SP 80052r2), TLS version 1.0 should only be used if necessarily needed. For government systems it should not be used at all.

Impact

Since CBC encryption methods are mandatory in TLSv1.0 (with the exception of null ciphers), TLSv1.0 is vulnerable to BEAST and Padding Oracle attacks, which could lead to a loss of integrity.

In addition, TLS version 1.0 could be exposed to further security vulnerabilities due to misconfigurations. The use of compression could make TLSv1.0 vulnerable to CRIME and/or breach attacks. If Triple-DES is used, data transmission is affected by the Sweet32 attack[3].

Recommendation

In order to prevent BEAST and padding oracle attacks, deactivating TLSv1.0 is recommended. Although, this might cause some issues with older clients which do not support higher versions of TLS (every current browser does support TLS versions higher than 1.0).

Steps to Reproduce

Communication via TLSv1.0 can be induced via **OpenSSL**:

```
$ openssl s_client -connect example.com:443 -tls1
```

[3] https://sweet32.info

5.4 Port Scan

Target:	example.com	**Base Score:**	0.0
CVSS Vector:	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	**Severity:**	Info
Wave Details:	There is no deviation from the CVSS Score.	**Wave Score:**	0

Description

During the test, the system was checked for open ports using the Nmap port scanner. The open ports identified during the test period are listed in the following table.

Host	Offene Ports	Dienste	Banner
example.com	80/tcp	http	nginx
	443/tcp	ssl/http	nginx
	8880/tcp	http	sw-cp-server

Typically, only TCP ports 80 and 443 are required to run the web application. The port scan shows that another service is accessible.

Impact

An open port is a potential security risk as it provides a target for cyber criminals. Attackers can use open ports to gain unauthorized access to services, exploit vulnerabilities or get malware onto the system.

Recommendation

In order to run the web application, typically only the TCP-Ports 80 and 443 are needed. In order to restrict the attacking vector it is highly advisable to disable or restrict the access to any other service.

Steps to Reproduce

```
$ nmap -sS -O -v -T4 --min-rate=100 --max-rate=500 -p- example.com
$ nmap -sU -v -T4 --max-retries=3 -p- --min-rate=100 --max-rate=500 example.com
```