

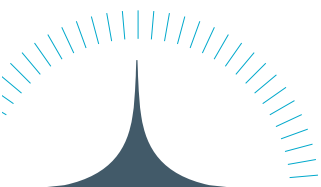


Penetration test test-voorbeeld.nl

Example B.V.

Version: 1.0

Date: 1 Januari 2018



COLOPHON

NORTHWAVE BV

Office address:

Marconibaan 49, 3439 MR Nieuwegein

Postal address:

Postbus 1305, 3430 BH Nieuwegein

E-mail:

info@northwave.nl

Phone number office:

+31 (0) 30 303 1240

Phone number NW-CERT (24*7):

0800 2255 2747

Website:

www.northwave.nl

VERSIONING AND AUTHORS

Version	Date	Author	Role	Status	Comment
0.1	01-12-2018	Frank de Korte	Security Consultant	Draft	First version
0.2	01-12-2018	Patrick de Brouwer	Security Consultant	Review	Review first version
0.3	02-12-2018	Frank de Korte	Security Consultant	Draft	Second version

APPROVAL

Version	Date	Name	Role	Status	Signature
1.0	01-12-2018	Bart Roos	Principal Consultant	Final	

CONDITIONS

Northwave commits to keeping all information confidential. The information provided by the client will only be used in the scope of the assignment. Northwave will ensure that all employees or third parties involved in the execution of the assignment are legally bound by this obligation. All information, trademarks, names, logos, portraits, tables or other matters where property rights apply and are used in this document are the property of their respective owner.

Table of contents

1	Management summary.....	5
1.1	Assignment.....	5
1.2	Results and conclusion.....	5
1.2.1	Research question answer	5
2	Introduction.....	6
2.1	Assignment.....	6
2.2	Scope	6
2.2.1	Test scope	6
2.2.2	Technical scope	6
2.2.3	Timeboxing.....	6
2.3	Tooling.....	6
2.4	Framework	7
2.5	Methodology	7
2.6	Vulnerability classification.....	8
2.6.1	CVSS 3.0 scoring system	8
2.6.2	CVSS 3.0 score and risk classification	9
2.7	Logs	10
3	Results.....	11
3.1	Critical risk.....	11
3.1.1	SQL injection	11
3.2	High Risk.....	13
3.2.1	Stored cross-site scripting (XSS)	13
3.3	Medium Risk.....	16
3.3.1	Unauthenticated access to sensitive information.....	16
3.3.2	Weak TLS cipher-suite.....	17
3.3.3	HTTP response header Strict-Transport-Security is missing.....	19
3.4	Low Risk.....	21
3.4.1	Cookie without Secure Flag.....	21
3.4.2	Stack trace is enabled.....	22
3.5	No Risk.....	24
3.5.1	HTTP response header Content-Security-Policy is missing	24
3.5.2	HTTP response header X-Frame-Options is missing.....	25
3.5.3	HTTP response header contains version information	27
4	Conclusion and recommendations	29
4.1	Conclusion	29
4.2	Findings	29
4.3	Recommendations.....	30

5	Appendix: CVSS 3.0 scoring system	31
5.1	<i>Cvss 3.0 Score model</i>	31
5.1.1	Base Metric	31
5.1.2	Temporal Metric.....	32
5.1.3	Environmental Metric	32
5.2	<i>Cvss 3.0 Vector values</i>	33
6	Appendix: Web Application Security Checklist	35

1 Management summary

1.1 ASSIGNMENT

Northwave received the assignment from Example B.V. to perform a black-box and a grey-box penetration tests on the test-voorbeeld.nl web application. The customer would like to know if the web application contains vulnerabilities that a malicious user, with or without credentials, could exploit. Specifically, Example B.V. would like to have the following research question answered:

- Can a malicious user obtain or manipulate customer data?

1.2 RESULTS AND CONCLUSION

A total of 11 findings have been identified during the penetration test. For each of these findings, a concrete recommendation has been made. Of these findings, 1 have been classified as a 'critical' risk, 1 as a 'high' risk and 3 as a 'medium' risk. We advise to remedy these issues as soon as possible. This concerns the following findings:

- SQL injection
- Stored cross-site scripting (XSS)
- Unauthenticated access to sensitive information
- Weak TLS cipher-suite
- HTTP response header Strict-Transport-Security is missing

All other findings have been classified as risk level 'low' or 'none'.

1.2.1 Research question answer

Northwave deems the security of the web application to be extremely weak, as the web application www.test-voorbeeld.nl is vulnerable to several attacks. Most notably, an SQL-injection vulnerability has been discovered, leading to the complete compromise of the database. This is a serious flaw that needs to be addressed immediately. Furthermore, a stored Cross-Site Scripting vulnerability exists within the application that puts your users at risk to several attacks. Northwave recommends to resolve these issues as soon as possible. A malicious user is able to obtain and manipulate user data via the SQL injection vulnerability. The same is true for all other types of data stored in the database.

2 Introduction

2.1 ASSIGNMENT

Northwave received the assignment from Example B.V. to perform a black-box and a grey-box penetration tests on the test-voorbeeld.nl web application. The customer would like to know if the web application contains vulnerabilities that a malicious user, with or without credentials, could exploit. Specifically, Example B.V. would like to have the following research question answered:

- Can a malicious user obtain or manipulate customer data?

2.2 SCOPE

In a penetration test the technical scope, the testing methods and the available timebox to perform the penetration test are relevant factors when assessing the penetration test results. For this test the scope of the assignment has been defined as described below.

2.2.1 Test scope

The test consists of a black-box and a grey-box penetration test. In the black-box scenario the website will be attacked without prior knowledge and without credentials. In the grey-box scenario a user account Test01 was provided for testing.

2.2.2 Technical scope

Below URL is in scope for this test:

- <https://www.test-voorbeeld.nl>

2.2.3 Timeboxing

Time is an important factor in every technical penetration test. Especially in complex environments, it is not possible to achieve a high level of permeation in a limited time. It is therefore important to consider the allotted time when rating the test results. In general, more time for testing will result in more findings. For performing this test, a total of 46 hours of testing time was available, which includes composing the report.

2.3 TOOLING

For performing the test, Northwave makes use of the following tools:

- **Kali Linux (rolling or 2.0 edition):** Kali Linux is a Linux operating system in which a number of open-source and other freely available tools have been incorporated. An overview of all tools that are available can be found on the Kali website¹.
- **Nessus Professional:** A commercial vulnerability scanner for determining known vulnerabilities within infrastructures².
- **Burp Suite Professional:** A commercial platform for testing web applications, in both a manual and automated manner³.
- **Acunetix Web Vulnerability Scanner:** A commercial vulnerability scanner aimed at determining security issues in web applications⁴.

2.4 FRAMEWORK

The report of this penetration test is executed based on the Certified Secure Web Application Security Test checklist. This checklist contains the executed tests for the most common vulnerabilities found in web applications, including those found in the OWASP top 10. The checklist provides an overview of all checks that have been performed and for which of those checks vulnerabilities have been discovered.

The test is based on the Certified Secure Web Application Security Test checklist⁵. The checklist is extended with additional checks based on practical experience of the Northwave Red Team with web application penetration testing. The checklist contains a list of tests for the most prevalent vulnerabilities in web applications, among which those of the OWASP top 10. The checklist provides insight in which checks have been performed and whether vulnerabilities have been discovered. An overview of the checklist is shown in the appendix 'Web Application Security Checklist'.

Besides this, Northwave also uses other tactics for finding technical vulnerabilities in this test, as taught in trainings such as GIAC Penetration Tester (GPEN) from SANS, Offensive Security Certified Professional (OSCP) and Offensive Security Certified Expert (OSCE) from Offensive Security.

2.5 METHODOLOGY

The penetration test is based on a widely-accepted methodology in the field. Depending on the scope of the assignment, this means that the following process is followed:

- **Information gathering:** With minimal knowledge of the systems, the tester simulates a possible malicious user without special permissions or account information to gather information about the environment. This information forms the base for the next steps of the test.

¹ <http://tools.kali.org/tools-listing>

² <https://www.tenable.com/products/nessus/nessus-professional>

³ <https://portswigger.net/burp/>

⁴ <http://www.acunetix.com/vulnerability-scanner/>

⁵ <https://www.certifiedsecure.com/checklists/cs-web-application-security-test.pdf>

- **Vulnerability scan:** Using automated tools, the tester examines which vulnerabilities are present in the infrastructure or applications. The scan results give an overview of vulnerabilities that are present and form the basis of the input for the black box, grey box and code review testing steps.
- **Black box test:** With the knowledge that is gained from the information gathering phase and the vulnerability scan, the tester will manually examine the environment for additional vulnerabilities. The tester has no special (user) privileges to the environment. The attack scenario that fits this is a malicious outsider who gains access to the environment over the Internet.
- **Grey box test:** With knowledge acquired during the information gathering phase and information provided by the owner, the tester tries to identify vulnerabilities in the environment. A scenario is simulated in which a malicious insider is attacking the target environment.
- **Code review:** During the code review phase, the tester will look for vulnerabilities by analysing the source code of an application. This test can be performed in parallel with the other testing steps. The tester validates the results of the other testing tests according to the code review.

2.6 VULNERABILITY CLASSIFICATION

Classification of vulnerabilities is done on the basis of the Common Vulnerability Scoring System (CVSS) version 3.0⁶. CVSS 3.0 offers a method to combine several characteristics of vulnerabilities according to a mathematical model, resulting in one total score. From this CVSS 3.0 score, a qualitative classification of the vulnerability is determined (low, medium, high or critical) to aid the owner of the tested system in judging and prioritising the vulnerabilities that have been found.

Northwave has chosen to use the CVSS 3.0 model because it offers complete transparency in how the score has been calculated. In addition to this, the result of the method can be completely traced by using the freely available CVSS 3.0 calculator⁷. Finally, the different vulnerabilities can be more easily compared because of the structure and consistency of the scoring system.

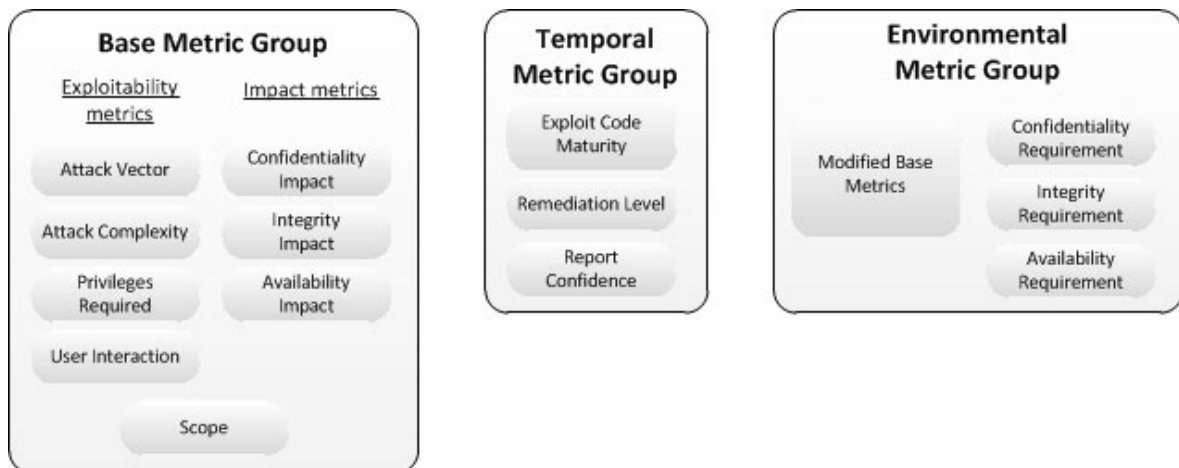
Naturally, the owner of the tested system remains responsible for making their own assessment of the risk associated with the findings. The classification used by the tester serves merely as a guide in the decision-making process regarding the measures that should be taken by the owner.

2.6.1 CVSS 3.0 scoring system

The CVSS 3.0 scoring system consists of three separate scoring groups: the base metric, temporal metric and environmental metric. These separate scores are composed based on several different criteria as depicted below:

⁶ <https://www.first.org/cvss/specification-document>

⁷ <https://www.first.org/cvss/calculator/3.0>



For a more elaborate explanation of the different criteria out of which these scores are composed, see the appendix 'CVSS 3.0 scoring system'.

2.6.2 CVSS 3.0 score and risk classification

Within the CVSS 3.0 model, only the use of the base score is required. Northwave also uses the previously mentioned additional scoring groups for calculating the eventual CVSS 3.0 score when they are relevant.

After all characteristics of a vulnerability are rated, it is possible to calculate a score with the CVSS 3.0 model, with which the severity of the vulnerability is indicated on scale from 0 to 10. Depending on the score, a qualitative risk classification (low, medium, high or critical) is assigned to the vulnerability per the table below:

Risk classification	CVSS 3.0 score	Description
(C) Critical	9.0 - 10.0	Critical vulnerabilities have a very high probability of being found by malicious user and are likely to be abused. The existence of a critical vulnerability stands in the way of deploying the environment and should be mitigated as soon as possible. Consider removing the vulnerable system from production if it is in use considering the possible high damage.
(H) High	7.0 - 8.9	Vulnerabilities that are classified as high should be mitigated as soon as possible. It is strongly advised not to use a system that contains such vulnerabilities.
(M) Medium	4.0 - 6.9	Vulnerabilities classified as medium should be resolved, but do not require direct action. It is recommended to set a term in which to solve the vulnerability and to consider taking mitigating measures in case the environment is already in use.

(L) Low	0.1 - 3.9	The vulnerabilities classified as low generally do not require direct action. However, it is recommended to remedy the vulnerabilities.
(N) None	0.0	Vulnerabilities with no risk classification do not pose a risk to security. Examples of this are vulnerabilities that have been solved during the test or security best practices.

2.7 LOGS

During the penetration test, extensive logs are generated containing the commands used by the tester and corresponding results. This information forms the basis of this report and ensures a process that can be reproduced and checked. Naturally, the stored information is available and can be requested up to three months after completion of the report. As the amount of data is generally quite large, the data is not sent together with the report by default and can be removed from Northwave systems after three months.

3 Results

3.1 CRITICAL RISK

3.1.1 SQL injection

The web application uses a database to store information. Data is retrieved and stored by the web application by using SQL queries. In an SQL injection attack, manipulation of the query used by the web application is possible. This may lead to the unauthorized access of data from the database. This is often caused by the improper input validation controls on user supplied data. In these cases a malicious user can alter the parameters used by the SQL query to mount a variety of attacks. In some cases SQL injection can lead to the loss of data, loss of availability and loss of data integrity. In a few cases, an SQL injection can even lead to remote code execution. Performing proper input validation on user supplied data is one way to prevent SQL injection attacks. Care must be taken to use the correct encoding of the characters that have a special meaning in the SQL language. Often this is done by using prepared statements⁸.

More information about this vulnerability can be found on the website of OWASP⁹.

<https://www.test-voorbeeld.nl>

The page test.php is vulnerable to an SQL-injection. It is possible for a malicious user by adding a specific string of characters (also known as a payload) to the ID-parameter to execute SQL commands on the Microsoft SQL server.

The payload below was used to get information from the database:

```
http://www.test-voorbeeld.nl/test.php?id=1 and 1=(select+table_name%2b'::'%2bcolumn_name as  
t+from+information_schema.columns FOR XML PATH(''))--
```

This payload tries to convert the entire database contents to an integer. As this is not possible, the database throws an error and displays the entire contents of the database in the resulting error message:

⁸ <https://secure.php.net/manual/en/mysqli.prepare.php>

⁹ https://www.owasp.org/index.php/SQL_Injection



Server Error in '/' Application.

Conversion failed when converting the varchar value 'a' to data type int.
 Microsoft SQL Server 2012 (SP1) - 11.0.3000.0 (X64)

Copyright (c) Microsoft Corporation
 Web Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601: Service Pack 1)
 Database : [REDACTED]

Table : Employee
 Columns : : Id : EmployeeName : Email : DateOfJoining : Username : Password : IsAdmin : IsActive
 Table : DownloadCategory
 Columns : : Id : DCategoryName : Desc : IsActive
 Table : Downloads
 Columns : : DownloadId : DCategoryId : Name : ApplicationPath : Description : DownloadFile : IsActive
 Table : News
 Columns : : Id : NewsTitle : NewsDetail : Description : IsActive
 Table : Member
 Columns : : UserId : OrderId : FullName : Email : Phone : Mobile : Address : Instructions : DateofRegister :
 Table : OrderFinal
 Columns : : Id : SrNo : OrderId : ProductId : ProductName : Qty : Price : TotalPrice : TotAmount : TotalAmoun
 Table : Services
 Columns : : ServiceId : ServiceName : ServiceDesc

Above screenshot shows that it is possible to read the entire contents of the database.

Finding 1: SQL injection	
CVSS Score	9.1 - critical
Finding	The web application is vulnerable to SQL injection.
Risk	A malicious user has unauthorized access to data from the database. This can lead to loss of data confidentiality, integrity and availability and sometimes remote code execution.
Recommendation	Perform proper input validation on all user supplied data.
Reference	CS-WEB-6.1
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

3.2 HIGH RISK

3.2.1 Stored cross-site scripting (XSS)

Users can enter data within the application that the browser sends as parameters within an HTTP request¹⁰ to the web server. This data is stored in the application and requested at a later time with an HTTP request. The application then generates an HTTP-response using the stored data that is processed by the browser to display a web page to the user.

A Stored Cross-Site Scripting (XSS)¹¹ occurs when an attacker can inject JavaScript code into one of the parameters that is stored within the application. This vulnerability happens because the application does not sufficiently check user input for malicious input. The JavaScript code is sent by the application in the HTTP response and the code is then executed by the user's browser. The injection is stored in the application and is effective every time any user loads the injected page.

To abuse a Stored XSS vulnerability, a malicious user must inject JavaScript code into the application. When successful, a malicious user can for example attempt to steal cookies and gain access to a user session or otherwise manipulate the user session from the browser's perspective (exfiltration or modification of data on the page).

To prevent this vulnerability, all input and output data should be validated and properly encoded to prevent any manipulation in the operation of the page. A cheat sheet is available on the OWASP website in which the preventive measures are described¹².

<https://www.test-voorbeeld.nl>

The web application allows users to leave a message on the visitors.php page. This message is stored in the database and shown to each visitor that requests the page. The application does not properly validate the input and output of the message. This makes it possible for a malicious users to enter Javascript code that is executed every time a visitor requests the page.

By adding below code to the page, a popup-window is shown in the browser upon visiting the page:

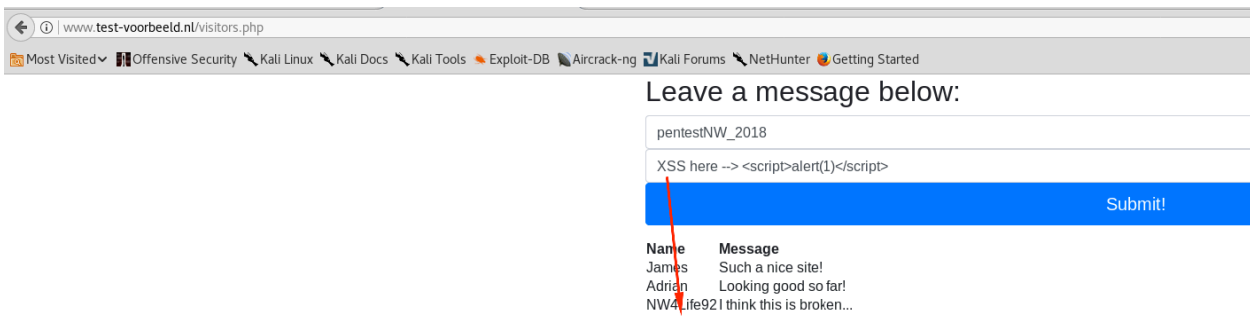
```
<script>alert(1)</script>
```

The above payload is entered into the message field.

¹⁰ https://www.tutorialspoint.com/http/http_requests

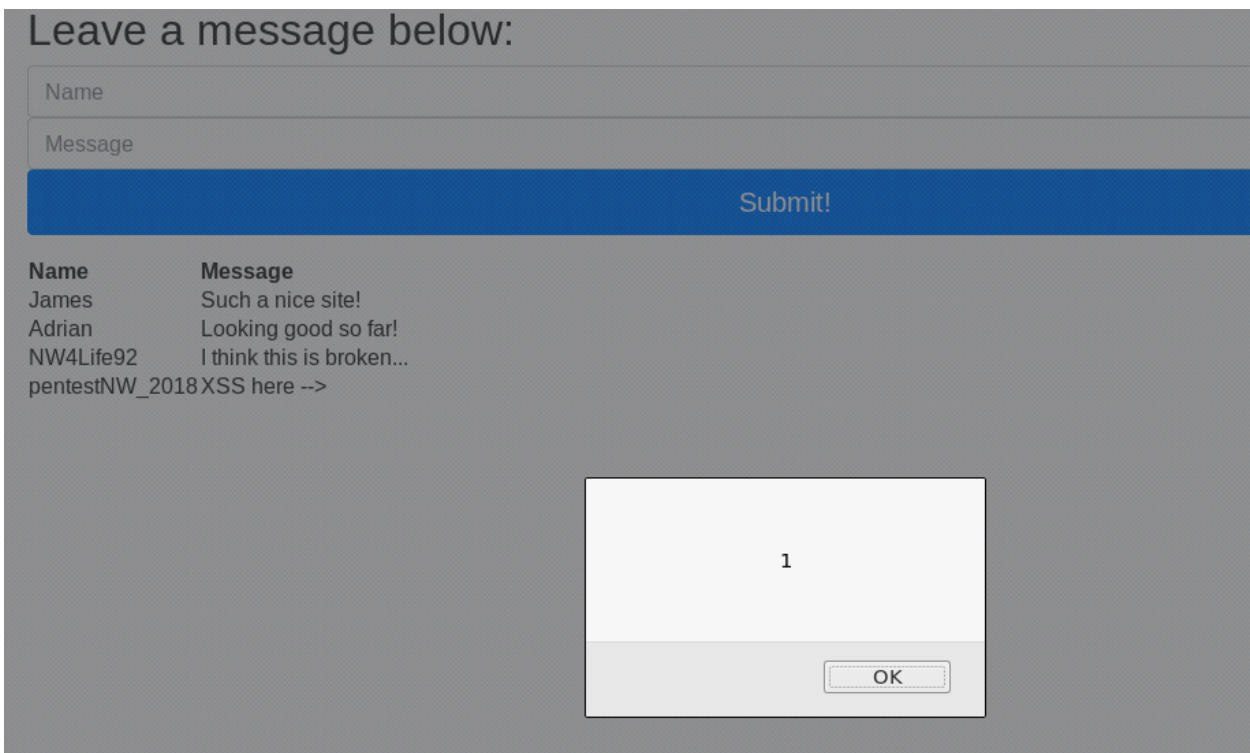
¹¹ [https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_\(OTG-INPVAL-002\)](https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_(OTG-INPVAL-002))

¹² [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)



Name	Message
James	Such a nice site!
Adrian	Looking good so far!
NW4Life92	I think this is broken...

When the Submit button is pressed, the code is send to the server and stored in the database. Because the server does not properly filter the data when the page is requested, the browser interpreters this as code and executes it.



Name	Message
James	Such a nice site!
Adrian	Looking good so far!
NW4Life92	I think this is broken...
pentestNW_2018	XSS here -->

1

OK

Above screenshot shows that the attack is successful and that the payload has been executed and a popup-window is shown.

The source-code of the page shows that the entered code is returned unfiltered.

```

    <tr></tr>
  <tr>
    <td>pentestMW_2018</td>
    <td>
      XSS here -->
      <script>alert(1)</script>
    </td>
  </tr>
</tbody>
</table>
</div>
</body>
</html>

```

This proves that the web application stored the payload and subsequently executes the injected JavaScript whenever a visitor request the page.

Finding 2: Stored cross-site scripting (XSS)	
CVSS Score	8.2 - high
Finding	The application is vulnerable to a stored XSS-attack.
Risk	A malicious user could execute JavaScript code in the victim's browser.
Recommendation	Validate and properly encode input and output data.
Reference	CS-WEB-6.3
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/H/I:L/A:N

3.3 MEDIUM RISK

3.3.1 Unauthenticated access to sensitive information

The web application has an authentication model that limits the access to (sensitive) information for unauthenticated users. However, not all areas are properly protected. It is possible for an unauthenticated user to request sensitive information via a specific URL. A malicious user can extract this information by accessing the URL. It is recommended to limit the access to the URL to authenticated users only.

For more information on this vulnerability, visit the website of OWASP¹³.

<https://www.test-voorbeeld.nl>

The web application allows an unauthorised user to access sensitive information via browsing to a specific URL. It is possible to request a file that normally is only available via the administration portal with the correct credentials. It is for a malicious user easy to guess the URL “/admin/phpinfo” or use automated scanners to extract it.

Request:

```
GET /admin/phpinfo HTTP/1.1
Host: www.test-voorbeeld.nl
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 14:51:53 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSIONID=asdh12308asdkjllrsafasf1; Path=/; HttpOnly
Content-Length: 50549

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml11-transitional.dtd">
<html><head>
<snip>
<title>phpinfo()</title></head>
<body><div class="center">
<table border="0" cellpadding="3" width="600">
<tr class="h"><td>
<a href="http://www.php.net/"></a><h1
class="p">PHP Version 5.5.11</h1>
</td></tr>
</table><br />
<table border="0" cellpadding="3" width="600">
<tr><td class="e">System </td><td class="v">Windows NT 6.2 build 9200 (Windows 8 Business Edition
i586</td></tr>
```

¹³ [https://www.owasp.org/index.php/Testing_for_Bypassing_Authentication_Schema_\(OTG-AUTHN-004\)](https://www.owasp.org/index.php/Testing_for_Bypassing_Authentication_Schema_(OTG-AUTHN-004))

```
<tr><td class="e">Build Date </td><td class="v">Nov 6 2017 12:27:14 </td></tr>
-snip-
```

The above response shows that it is possible to retrieve system information from the webserver. This proves that it is possible to request this page without authentication. A malicious user can extract information from this file with regards to software components used by the application.

Finding 3: Unauthenticated access to sensitive information	
CVSS Score	6.5 - medium
Finding	Unauthenticated access to sensitive information.
Risk	A malicious user can access sensitive information by visiting a specific URL.
Recommendation	It is recommended to review the access to the URL and limit the access to the sensitive information to authenticated users only.
Reference	CS-WEB-5.1
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

3.3.2 Weak TLS cipher-suite

SSL/TLS uses encryption algorithms to encrypt the network traffic. These encryption algorithms alongside the key sizes are bundled into so-called cipher-suites. By encrypting the network traffic, malicious users are unable to intercept the traffic to read and edit it. However, the server uses one or more weak or out-dated cipher-suites which could in some cases allow an attacker adapt enough at exploiting its weaknesses to extract sensitive data from the network traffic or even edit it.

The NCSC has created ICT-security guidelines for Transport Layer Security (TLS)¹⁴ which contains an overview of the cipher-suites that are considered safe as well as a recommendation of the configuration of TLS. The NCSC differentiates in the guidelines between safe cipher-suites and acceptable cipher-suites. The following cipher-suites are considered safe:

```
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
```

¹⁴ "https://www.ncsc.nl/actueel/whitepapers/ict-beveiligingsrichtlijnen-voor-transport-layer-security-tls.html":https://www.ncsc.nl/actueel/whitepapers/ict-beveiligingsrichtlijnen-voor-transport-layer-security-tls.html

To stay backwards compatible with older systems and still have a reasonably safe configuration, it is also possible to enable the following accepted cipher-suites:

```
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

The configuration of the cipher-suites depends on the software in use. Please consult the appropriate documentation for the affected item.

<https://www.test-voorbeeld.nl>

The following configuration was extracted from TCP/IP port 443 using the tool SSLScan¹⁵. Below output shows the list of the ciphers that are in use by the webserver and are considered weak.

The output of the tool:

```
Here is the list of weak SSL ciphers supported by the remote server :
Low Strength Ciphers (<= 64-bit key)

    TLSv1
      EDH-RSA-DES-CBC-SHA      Kx=DH      Au=RSA      Enc=DES-CBC (56)      Mac=SHA1
      EXP-DES-CBC-SHA         Kx=RSA (512)  Au=RSA      Enc=DES-CBC (40)      Mac=SHA1
export
      EXP-RC4-MD5              Kx=RSA (512)  Au=RSA      Enc=RC4 (40)          Mac=MD5
export
      DES-CBC-SHA              Kx=RSA      Au=RSA      Enc=DES-CBC (56)      Mac=SHA1

The fields above are :

{OpenSSL ciphername}
Kx={key exchange}
Au={authentication}
Enc={symmetric encryption method}
Mac={message authentication code}
{export flag}
```

¹⁵ <https://github.com/rbsec/ssllscan>

Finding 4: Weak TLS cipher-suite	
CVSS Score	4.7 - medium
Finding	The TLS configuration is set to use weak cipher-suites.
Risk	A malicious user may be able to decrypt the secure network connection to view or edit sensitive data.
Recommendation	Only use the strong cipher-suites advised by the NCSC ICT-security guidelines for Transport Layer Security (TLS).
Reference	CS-WEB-3.11
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:N/A:N

3.3.3 HTTP response header Strict-Transport-Security is missing

HTTP is a protocol for communication between the web browser of a user and the web server to serve web pages. In order to exchange extra information, the server can use certain HTTP headers. Web servers that receive certain HTTP request headers can respond with different HTTP response headers to instruct the browser how to display the requested web page.

The web server does not use the HTTP response header Strict-Transport-Security (HSTS). This header ensures that the current request and every future request is send over an encrypted communications channel (HTTPS). An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP¹⁶, so that if a targeted user follows a link to the site from an HTTP page, their web browser never attempts to use an encrypted connection.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer.

It is recommended to enable Strict-Transport-Security with the value:

```
Strict-Transport-Security: max-age= 31536000; includeSubdomains; preload;
```

The example above sets the HSTS header to have a lifetime of one year and is enforced on all (future) sub domains. The web application is listed (hard coded) in the Google Chrome project that enforces HTTPS only¹⁷.

¹⁶ <https://moxie.org/software/sslstrip/>

¹⁷ https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

For further reference, please refer to the manual page from Mozilla¹⁸.

<https://www.test-voorbeeld.nl>

Below request and response example shows that the Strict-Transport-Security header is missing when requesting the login page.

Request:

```
GET /login.php HTTP/1.1
Host: www.test-voorbeeld.nl
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 16:32:34 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSIONID=asdh12308asdkjllrsafasf1; Path=/; HttpOnly
Content-Length: 16068
-snip-
```

Finding 5: HTTP response header Strict-Transport-Security is missing	
CVSS Score	4.2 - medium
Finding	The HTTP response header Strict-Transport-Security is not used.
Risk	An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users.
Recommendation	It is recommended to set the HTTP response header Strict-Transport-Security with the suggested value.
Reference	CS-WEB-3.9
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N

¹⁸ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security>

3.4 LOW RISK

3.4.1 Cookie without Secure Flag

The secure flag is an option that can be set by the application server when sending a new cookie to the user. By setting the secure flag, browsers which supports the secure flag, will prevent the transmission of a cookie over an unencrypted channel and prevents a malicious user to execute a man-in-the-middle attack. When a malicious user is able to grab a cookie, he can impersonate the user.

More information about this vulnerability can be found on the website of OWASP¹⁹.

<https://www.test-voorbeeld.nl>

The cookie PHPSESSIONID is created without the secure flag on URL www.test-voorbeeld.nl.

Request:

```
GET / HTTP/1.1
Host: www.test-voorbeeld.nl
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 12:51:53 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSIONID=asdh12308asdkj11rsafasf1; Path=/; HttpOnly
Content-Length: 113
-snip-
```

Finding 6: Cookie without Secure Flag

CVSS Score	3.7 - low
Finding	The secure flag will prevent the browser to sent a cookie over an unencrypted channel.
Risk	A malicious user could perform a man-in-the-middle attack which could lead to impersonate the user.
Recommendation	It is recommended to enable the secure flag for all cookies that are set by the application server.
Reference	CS-WEB-7.7

¹⁹ <https://www.owasp.org/index.php/SecureFlag>

Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:A/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N

3.4.2 Stack trace is enabled

The web server displays a stack trace²⁰ when an error occurs. A stack trace is often used in a test environment to easily detect and fix bugs. A stack trace usually shows sensitive information regarding the architecture and software in use by the web application. A malicious user can use this information to further penetrate the system.

It is recommended to disable the stack trace on a production environment to prevent the leaking of sensitive information. For more information regarding stack traces please look to the website of OWASP²¹.

<https://www.test-voorbeeld.nl>

The webpage test.php uses a parameter called ID from the URL to load the content. Whenever the URL parameter does not contain an integer, the web server returns an error message containing a stack trace with sensitive information about the web application.

Request:

```
GET /test.php?id=bla HTTP/1.1
Host: www.test-voorbeeld.nl
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Response:

```
HTTP/1.1 500 Error
Date: Wed, 21 Feb 2018 10:13:01 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSIONID=234dsf2p0df2tdh2wdesg; Path=/; HttpOnly
Content-Length: 2307
-snip-
```

In the screenshot below an example is shown of such an error message:

²⁰ https://en.wikipedia.org/wiki/Stack_trace

²¹ [https://www.owasp.org/index.php/Testing_for_Stack_Traces_\(OTG-ERR-002\)](https://www.owasp.org/index.php/Testing_for_Stack_Traces_(OTG-ERR-002))

www.test-voorbeeld.nl/test.php?id=bla

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter Getting Started

Server Error in '/' Application.

Object reference not set to an instance of an object.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.NullReferenceException: Object reference not set to an instance of an object.

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the stack trace.

Stack Trace:

```
[NullReferenceException: Object reference not set to an instance of an object.]
  Com.Vizz.Web.Index.Page_Load(Object sender, EventArgs e) +72
  System.Web.UI.Control.LoadRecursive() +70
  System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +3177
```

Finding 7: Stack trace is enabled	
CVSS Score	3.7 - low
Finding	The web server displays a stack trace when an error occurs.
Risk	Through the information displayed in the stack trace, a malicious user gets insight into the architecture and software in use by the web application.
Recommendation	Disable the stack trace on the production environment.
Reference	CS-WEB-2.4
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:N/A:N

3.5 NO RISK

3.5.1 HTTP response header Content-Security-Policy is missing

HTTP is a protocol for communication between the web browser of a user and the web server to serve web pages. In order to exchange extra information, the server can use certain HTTP headers. Web servers that receive certain HTTP request headers can respond with different HTTP response headers to instruct the browser how to display the requested web page.

The web server does not set the HTTP response header Content-Security-Policy (CSP). This header contains a policy that web browsers use to determine which content-types from which sources should be allowed. This protects the user of modern web browsers from client-side code injection vulnerabilities and clickjacking attacks. To prevent clickjacking attacks, two options are possible. The method with the most support from a modern browser is to set the frame-ancestor policy within the Content-Security-Policy HTTP header. It is also possible to set the X-Frame-Options HTTP header. More information about clickjacking attacks can be found at the website of OWASP²².

Enabling this header provides an extra layer of security for the user when the web application contains vulnerabilities like cross site scripting. More information about the user of this header (and browser support) can be found on the OWASP website²³.

It is recommended to apply the header Content-Security-Policy on every page that outputs HTML. Implement the policy CSP as strongly as possible (disallowing the directives unsafe-inline and unsafe-eval). For further reference, please refer to the manual page from Mozilla²⁴.

<https://www.test-voorbeeld.nl>

Below request and response example shows that the Content-Security-Policy header is missing when requesting the login page.

Request:

```
GET /login.php HTTP/1.1
Host: www.test-voorbeeld.nl
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 16:32:34 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
```

²² https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet

²³ https://www.owasp.org/index.php/Content_Security_Policy

²⁴ <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

```
Content-Type: text/html;charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSIONID=asdh12308asdkj11rsafasf1; Path=/; HttpOnly
Content-Length: 16068
-snip-
```

Finding 8: HTTP response header Content-Security-Policy is missing	
CVSS Score	0.0 - none
Finding	The HTTP response header Content-Security-Policy is not used on pages displaying HTML content.
Risk	In the case of other vulnerabilities like Cross-Site-Scripting or vulnerabilities discovered in the future, the web browser does not offer the user additional protection against client-side attacks.
Recommendation	It is recommended to set the HTTP response header Content-Security-Policy with a policy as strict as possible.
Reference	CS-WEB-12.1
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

3.5.2 HTTP response header X-Frame-Options is missing

HTTP is a protocol for communication between the web browser of a user and the web server to serve web pages. In order to exchange extra information, the server can use certain HTTP headers. Web servers that receive certain HTTP request headers can respond with different HTTP response headers to instruct the browser how to display the requested web page.

The web server does not use the HTTP response header X-Frame-Options. By enabling this header, a web browser is instructed the use of frames. This protects the user of modern web browsers from clickjacking attacks. To prevent clickjacking attacks, two options are possible. The method with the most support from a modern browser is to set the frame-ancestor policy within the Content-Security-Policy HTTP header. It is also possible to set the X-Frame-Options HTTP header. More information about clickjacking attacks can be found at the website of OWASP²⁵.

Due the lack of the header X-Frame-Options, an attacker is able to craft a web page that includes a frame of the web application. For example, an attacker might be able to circumvent a user by using a transparent frame to perform unwanted actions. It is recommended to make use of one of the following values:

A page may only be displayed in a frame on the same origin as the page itself:

²⁵ https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet

```
X-Frame-Options: SAMEORIGIN
```

The page may not be displayed in a frame, regardless of the web application attempting to do so:

```
X-Frame-Options: DENY
```

For further reference, please refer to the manual page from Mozilla²⁶.

<https://www.test-voorbeeld.nl>

Below request and response example shows that the 'X-Frame-Options' header is missing when requesting the login page.

Request:

```
GET /login.php HTTP/1.1
Host: www.test-voorbeeld.nl
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 16:32:34 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSIONID=asdh12308asdkjllrsafasf1; Path=/; HttpOnly
Content-Length: 16068
-snip-
```

Finding 9: HTTP response header X-Frame-Options is missing	
CVSS Score	0.0 - none
Finding	The HTTP response header X-Frame-Options is not used.
Risk	A user is vulnerable to clickjacking attacks by using the web application.
Recommendation	It is recommended to set the HTTP response header Strict-Transport-Security with one of the suggested values.
Reference	CS-WEB-11.1
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

²⁶ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>

3.5.3 HTTP response header contains version information

HTTP is a protocol for communication between the web browser of a user and the web server to serve web pages. In order to exchange extra information, the server can use certain HTTP headers. Web servers that receive certain HTTP request headers can respond with different HTTP response headers to instruct the browser how to display the requested web page.

The web application discloses information about the used software and architecture. An attacker is able to search for weaknesses and vulnerabilities related to the disclosed version numbers. It might be possible in the future that an exploit related to the disclosed version number is publicly available. In this case, the web application might be an attractive target. See for more information about the vulnerability the website of Troy Hunt²⁷.

At the moment, there are no working exploits publicly available for the discovered version numbers. Still, it is recommended to hide information about the used software and architecture.

<https://www.test-voorbeeld.nl>

The HTTP-header server displays the Microsoft IIS version number.

Request:

```
GET / HTTP/1.1
Host: www.test-voorbeeld.nl
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 12:51:53 GMT
Server: Microsoft-IIS/8.5
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Connection: close
Set-Cookie: PHPSESSID=asdh12308asdkj11rsafasf1; Path=/; HttpOnly
Content-Length: 113
-snip-
```

²⁷ <https://www.troyhunt.com/shhh-dont-let-your-response-headers/>

Finding 10: HTTP response header contains version information	
CVSS Score	0.0 - none
Finding	HTTP response headers discloses version information about the used software and architecture.
Risk	An attacker is able to obtain the version number of the used software and architecture. With this information, an attacker is able to perform more targeted attacks to the web application.
Recommendation	It is recommended to hide information about the used software and architecture.
Reference	CS-WEB-12.7
Location	https://www.test-voorbeeld.nl
CVSS-vector	CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

4 Conclusion and recommendations

4.1 CONCLUSION

A total of 11 findings have been identified during the penetration test. Of these findings, 1 have been classified as a 'critical' risk and 1 as a 'high' risk. These findings are advised to be remedied as soon as possible. It is unwise to deploy, or have in use, an application with these types of vulnerabilities. For the 3 findings that have been classified as 'medium' risk, it is strongly advised to remedy them short-term. All other findings have been classified as risk level 'low' or 'none'.

Critical	1
High	1
Medium	3
Low	2
None	3

The web application is prone to several vulnerabilities that can be classified as critical and high. Northwave recommended to address these as soon as possible. Most notably, an SQL-injection vulnerability has been discovered, leading to the complete compromise of the database. This is a serious flaw that needs to be addressed immediately. Furthermore, a stored Cross-Site Scripting vulnerability exists within the application that puts your users at risk to several attacks. Northwave recommends to resolve these issues as soon as possible. Several low risk findings have been discovered as well. Northwave recommends to review the development process to implement routine checks to ensure you are coding securely. Using pair-wise programming or peer reviews can help ensure code is being written according to the latest security standards. Furthermore, routine penetration tests should also be performed to verify that your products are secure. Based on the findings discovered during the penetration test, Northwave concludes that the application is not secure. It is not recommended to publish the website in its current state.

4.2 FINDINGS

Summarized, the following findings were identified during the test:

- The web application is vulnerable to SQL injection.
- The application is vulnerable to a stored XSS-attack.
- Unauthenticated access to sensitive information.
- The TLS configuration is set to use weak cipher-suites.
- The HTTP response header Strict-Transport-Security is not used.
- The secure flag will prevent the browser to sent a cookie over an unencrypted channel.
- The web server displays a stack trace when an error occurs.

- The HTTP response header Content-Security-Policy is not used on pages displaying HTML content.
- The HTTP response header X-Frame-Options is not used.
- HTTP response headers disclose version information about the used software and architecture.

4.3 RECOMMENDATIONS

From the previous summary of findings, the following recommendations are made:

- Perform proper input validation on all user supplied data.
- Validate and properly encode input and output data.
- It is recommended to review the access to the URL and limit the access to the sensitive information to authenticated users only.
- Only use the strong cipher-suites advised by the NCSC ICT-security guidelines for Transport Layer Security (TLS).
- It is recommended to set the HTTP response header Strict-Transport-Security with the suggested value.
- It is recommended to enable the secure flag for all cookies that are set by the application server.
- Disable the stack trace on the production environment.
- It is recommended to set the HTTP response header Content-Security-Policy with a policy as strict as possible.
- It is recommended to set the HTTP response header Strict-Transport-Security with one of the suggested values.
- It is recommended to hide information about the used software and architecture.

5 Appendix: CVSS 3.0 scoring system

5.1 CVSS 3.0 SCORE MODEL

In this appendix, a summary of the CVSS 3.0 scoring system is given. For extended information on the different elements that compose the CVSS-scores, see the CVSS 3.0 specification document²⁸.

5.1.1 Base Metric

The base metric is comprised of eight different criteria with corresponding choices in the field of exploitability, impact and scope:

Exploitability

- **Attack vector (AV):** Describes the way in which the vulnerability can be abused.
 - Network (N): The vulnerability can be abused remotely via the network or the Internet.
 - Adjacent (A): The vulnerability can be abused from the local network.
 - Local (L): A vulnerability that can only be abused on the local system.
 - Physical (P): To abuse the vulnerability, physical access is necessary.
- **Attack complexity (AC):** Describes the complexity of the vulnerability.
 - Low (L): The attack is quite simple can be reproduced easily.
 - High (H): The attack is quite complex and difficult to execute.
- **Privileges required (PR):** Describes the necessary user rights to abuse the vulnerability:
 - None (N): No privileges are necessary.
 - Low (L): Access with limited user privileges is necessary.
 - High (H): Access with extended privileges is required (for example, an admin account).
- **User interaction (UI):** Describes if interaction from a user is necessary to exploit the vulnerability.
 - None (N): The vulnerability can be abused without a user's actions.
 - Required (R): A user must interact with the system to abuse the vulnerability.

Impact

- **Confidentiality (C):** Defines the impact on confidentiality of information.
 - High (H): The vulnerability directly impacts confidentiality. Examples of this are an administration password or special categories of personal information that can be gained.
 - Low (L): The vulnerability impacts confidentiality, but no confidential information can be gained directly; the information that can be gained is limited.
 - None (N): The vulnerability has no impact on confidentiality.
- **Integrity (I):** Defines the impact on integrity of the system or the stored information.
 - High (H): The vulnerable component has direct impact on integrity; a malicious user is able to manipulate or remove information at will.

²⁸ <https://www.first.org/cvss/specification-document>

- Low (L): The vulnerable component has limited impact on integrity. For example, a malicious user might have limited control over the consequences of changes.
 - None (N): The vulnerability has no impact on integrity.
- **Availability (A):** Defines the impact on availability on the system or the information.
 - High (H): The vulnerable component directly impacts availability. For example, it might be possible to make the system unavailable to users permanently or even temporarily.
 - Low (L): It is possible to make the vulnerable component unavailable in a temporary or limited manner.
 - None (N): The vulnerability has no impact on availability.

Scope

- **Scope (S):** Defines if the vulnerability influences other systems.
 - Unchanged (U): Exploiting the vulnerability does not influence other systems or networks.
 - Changed (C): Exploiting the vulnerability does influence other systems or networks.

5.1.2 Temporal Metric

The temporal score consists of the following three criteria with corresponding choices:

- **Exploit Code Maturity (E):** Describes the availability of exploit code to abuse the vulnerability.
 - Not defined (X): Not applicable.
 - High (H): There is working, simple and reliable code available to exploit the code in every situation.
 - Functional (F): There is exploit code available that works in most situations.
 - Proof-of-Concept (P): Demonstration code of the exploit is available.
 - Unproven (U): There is no code available or the exploit is theoretical.
- **Remediation Level (RL):** Describes the existence of a solution to mitigate the vulnerability.
 - Not defined (X): Not applicable.
 - Unavailable (U): There is no solution available to mitigate the vulnerability.
 - Workaround (W): An unofficial workaround exists to mitigate the vulnerability.
 - Temporary fix (T): There is an official, temporary solution.
 - Official fix (O): The product vendor has published an official solution.
- **Report Confidence (RC):** Describes the reliability of information about the vulnerability.
 - Not defined (X): Not applicable.
 - Confirmed (C): Detailed information exists and the exploit can be reproduced.
 - Reasonable (R): Details have been published, but there is no full confidence about the cause of the vulnerability and it has not been confirmed by researchers.
 - Unknown (U): Information has been published without details. Researchers are not sure about the existence of the vulnerability.

5.1.3 Environmental Metric

The environmental score consists of the following criteria and corresponding choices:

- **Confidentiality Requirement (CR):** The degree with which the confidentiality of information should be guaranteed within the IT environment.
 - Not defined (X): No value assigned by the customer.
 - High (H): Loss of confidentiality has a very large impact.
 - Medium (M): Loss of confidentiality has serious consequences.
 - Low (L): Loss of confidentiality has limited consequences.
- **Integrity Requirement (IR):** The degree with which the integrity of the system and information should be guaranteed within the IT environment.
 - Not defined (X): No value assigned by the customer.
 - High (H): Loss of integrity has a very large impact.
 - Medium (M): Loss of integrity has serious consequences.
 - Low (L): Loss of integrity has limited consequences.
- **Availability Requirement (AR):** The degree with which the availability of the system and information should be guaranteed within the IT environment.
 - Not defined (X): No value assigned by the customer.
 - High (H): Loss of availability has a very large impact.
 - Medium (M): Loss of availability has serious consequences.
 - Low (L): Loss of availability has limited consequences.

Besides this, the CVSS 3.0 model also offers the possibility to adapt the eight criteria in the base score to the specific business environment. These criteria are called the 'Modified base metrics'.

5.2 CVSS 3.0 VECTOR VALUES

Every finding in this report has been graded according to the CVSS 3.0 model. To give a short summary of the choices that were made for this grade, a 'vector string' or vector value is used. In such a string or value, an abbreviation of the different criteria and value is noted. The abbreviations are listed in the following table:

Metric Group	Metric Name	Possible values	Mandatory
Base	Attack Vector, AV	[N,A,L,P]	Yes
	Attack Complexity, AC	[L,H]	Yes
	Privileges Required, PR	[N,L,H]	Yes
	User Interaction, UI	[N,R]	Yes
	Scope, S	[U,C]	Yes
	Confidentiality, C	[H,L,N]	Yes
	Integrity, I	[H,L,N]	Yes
	Availability, A	[H,L,N]	Yes
Temporal	Exploit Code Maturity, E	[X,H,F,P,U]	No
	Remediation Level, RL	[X,U,W,T,O]	No
	Report Confidence, RC	[X,C,R,U]	No

Environmental	Confidentiality Req., CR	[X,H,M,L]	No
	Integrity Req., IR	[X,H,M,L]	No
	Availability Req., AR	[X,H,M,L]	No
	Modified Attack Vector, MAV	[X,N,A,L,P]	No
	Modified Attack Complexity, MAC	[X,L,H]	No
	Modified Privileges Required, MPR	[X,N,L,H]	No
	Modified User Interaction, MUI	[X,N,R]	No
	Modified Scope, MS	[X,U,C]	No
	Modified Confidentiality, MC	[X,N,L,H]	No
	Modified Integrity, MI	[X,N,L,H]	No
	Modified Availability, MA	[X,N,L,H]	No

6 Appendix: Web Application Security Checklist

This Web Application Security Checklist is based on the existing checklist of Certified Secure²⁹ and is extended with additional checks found in section 12 of the checklist. These additional checks are based on practical experience of the Northwave Red Team with web application penetration testing. The checklist provides an overview of the performed tests where they are applicable. A green checkmark indicates that no vulnerabilities related to the check have been found within the available testing time. A red cross mark indicates that the check failed and one or more vulnerabilities related to the check are present in the tested environment.

#	Web Application Security Testing Checklist	Result	Paragraph
CS-WEB-1.0 Deployment			
CS-WEB-1.1	Test for missing security updates	✓	-
CS-WEB-1.2	Test for unsupported or end-of-life software versions	✓	-
CS-WEB-1.3	Test for HTTP TRACK and TRACE methods	✓	-
CS-WEB-1.4	Test for extraneous functionality	✓	-
CS-WEB-1.5	Test the server using the Server Security Test Checklist	✓	-
CS-WEB-2.0 Information Disclosure			
CS-WEB-2.1	Test for extraneous files in the document root	✓	-
CS-WEB-2.2	Test for extraneous directory listings	✓	-
CS-WEB-2.3	Test for accessible debug functionality	✓	-
CS-WEB-2.4	Test for sensitive information in log and error messages	✗	3.4.2
CS-WEB-2.5	Test for sensitive information in robots.txt	✓	-
CS-WEB-2.6	Test for sensitive information in source code	✓	-
CS-WEB-2.7	Test for disclosure of internal addresses	✓	-
CS-WEB-3.0 Privacy and Confidentiality			
CS-WEB-3.1	Test for sensitive information stored in URLs	✓	-
CS-WEB-3.2	Test for unencrypted sensitive information stored at the client-side	✓	-
CS-WEB-3.3	Test for sensitive information stored in (externally) archived pages	✓	-
CS-WEB-3.4	Test for content included from untrusted sources	✓	-
CS-WEB-3.5	Test for caching of pages with sensitive information	✓	-

²⁹ <https://www.certifiedsecure.com/checklists/cs-web-application-security-test.pdf>

#	Web Application Security Testing Checklist	Result	Paragraph
CS-WEB-3.6	Test for insecure transmission of sensitive information	✓	-
CS-WEB-3.7	Test for non-SSL/TLS pages on sites processing sensitive information	✓	-
CS-WEB-3.8	Test for SSL/TLS pages served with mixed content	✓	-
CS-WEB-3.9	Test for missing HSTS header on full SSL sites	✗	3.3.3
CS-WEB-3.10	Test for known vulnerabilities in SSL/TLS	✓	-
CS-WEB-3.11	Test for weak, untrusted or expired SSL certificates	✗	3.3.2
CS-WEB-3.12	Test for the usage of unproven cryptographic primitives	✓	-
CS-WEB-3.13	Test for the incorrect usage of cryptographic primitives	✓	-
CS-WEB-4.0 State Management			
CS-WEB-4.1	Test for client-side state management	✓	-
CS-WEB-4.2	Test for invalid state transitions	✓	-
CS-WEB-5.0 Authentication and Authorization			
CS-WEB-5.1	Test for missing authentication or authorization	✗	3.3.1
CS-WEB-5.2	Test for client-side authentication	✓	-
CS-WEB-5.3	Test for predictable and default credentials	✓	-
CS-WEB-5.4	Test for predictable authentication or authorization tokens	✓	-
CS-WEB-5.5	Test for authentication or authorization based on obscurity	✓	-
CS-WEB-5.6	Test for identifier-based authorization	✓	-
CS-WEB-5.7	Test for acceptance of weak passwords	✓	-
CS-WEB-5.8	Test for plaintext retrieval of passwords	✓	-
CS-WEB-5.9	Test for missing rate limiting on authentication functionality	✓	-
CS-WEB-5.10	Test for missing re-authentication when changing credentials	✓	-
CS-WEB-5.11	Test for missing logout functionality	✓	-
CS-WEB-6.0 User Input			
CS-WEB-6.1	Test for SQL injection	✗	3.1.1
CS-WEB-6.2	Test for path traversal and filename injection	✓	-
CS-WEB-6.3	Test for cross-site scripting	✗	3.2.1
CS-WEB-6.4	Test for system command injection	✓	-
CS-WEB-6.5	Test for XML injection	✓	-
CS-WEB-6.6	Test for XPath injection	✓	-
CS-WEB-6.7	Test for XSL(T) injection	✓	-
CS-WEB-6.8	Test for SSI injection	✓	-
CS-WEB-6.9	Test for HTTP header injection	✓	-
CS-WEB-6.10	Test for HTTP parameter injection	✓	-

#	Web Application Security Testing Checklist	Result	Paragraph
CS-WEB-6.11	Test for LDAP injection	✓	-
CS-WEB-6.12	Test for dynamic scripting injection	✓	-
CS-WEB-6.13	Test for regular expression injection	✓	-
CS-WEB-6.14	Test for data property/field injection	✓	-
CS-WEB-6.15	Test for protocol-specific injection	✓	-
CS-WEB-6.16	Test for expression language injection	✓	-
CS-WEB-7.0			
Sessions			
CS-WEB-7.1	Test for cross-site request forgery (CSRF)	✓	-
CS-WEB-7.2	Test for predictable CSRF tokens	✓	-
CS-WEB-7.3	Test for missing session revocation on logout	✓	-
CS-WEB-7.4	Test for missing session regeneration on login	✓	-
CS-WEB-7.5	Test for missing session regeneration when changing credentials	✓	-
CS-WEB-7.6	Test for missing revocation of other sessions when changing credentials	✓	-
CS-WEB-7.7	Test for missing Secure flag on session cookies	✗	3.4.1
CS-WEB-7.8	Test for missing HttpOnly Flag on session cookies	✓	-
CS-WEB-7.9	Test for non-restrictive domain on session cookies	✓	-
CS-WEB-7.10	Test for non-restrictive or missing path on session cookies	✓	-
CS-WEB-7.11	Test for predictable session identifiers	✓	-
CS-WEB-7.12	Test for session identifier collisions	✓	-
CS-WEB-7.13	Test for session fixation	✓	-
CS-WEB-7.14	Test for insecure transmission of session identifiers	✓	-
CS-WEB-7.15	Test for external session hijacking	✓	-
CS-WEB-7.16	Test for missing periodic expiration of sessions	✓	-
CS-WEB-8.0			
File Uploads			
CS-WEB-8.1	Test for storage of uploaded files in the document root	✓	-
CS-WEB-8.2	Test for execution or interpretation of uploaded files	✓	-
CS-WEB-8.3	Test for uploading outside of designated upload directory	✓	-
CS-WEB-8.4	Test for missing size restrictions on uploaded files	✓	-
CS-WEB-8.5	Test for missing type validation on uploaded files	✓	-
CS-WEB-9.0			
Content			
CS-WEB-9.1	Test for missing or non-specific content type definitions	✓	-
CS-WEB-9.2	Test for missing character set definitions	✓	-
CS-WEB-9.3	Test for missing anti content sniffing measures	✓	-
CS-WEB-10.0			
XML Processing			

#	Web Application Security Testing Checklist	Result	Paragraph
CS-WEB-10.1	Test for XML external entity expansion	✓	-
CS-WEB-10.2	Test for external DTD parsing	✓	-
CS-WEB-10.3	Test for extraneous or dangerous XML extensions	✓	-
CS-WEB-10.4	Test for recursive entity expansion	✓	-
CS-WEB-11.0 Miscellaneous			
CS-WEB-11.1	Test for missing anti-clickjacking measures	✗	3.5.2
CS-WEB-11.2	Test for open redirection	✓	-
CS-WEB-11.3	Test for insecure cross-domain access policy	✓	-
CS-WEB-11.4	Test for missing rate limiting on e-mail functionality	✓	-
CS-WEB-11.5	Test for missing rate limiting on resource intensive functionality	✓	-
CS-WEB-11.6	Test for inappropriate rate limiting resulting in a denial of service	✓	-
CS-WEB-11.7	Test for application- or setup-specific problems	✓	-
CS-WEB-12.0 Northwave specific checks			
CS-WEB-12.1	Test for missing Content-Security-Policy HTTP header	✗	3.5.1
CS-WEB-12.2	Test for Insecure Direct Object Reference	✓	-
CS-WEB-12.3	Test for missing X-XSS-Protection HTTP header	✓	-
CS-WEB-12.4	Test for public availability of test environment	✓	-
CS-WEB-12.5	Test for Local File Inclusion	✓	-
CS-WEB-12.6	Test for Remote File Inclusion	✓	-
CS-WEB-12.7	Test for version information in HTTP response header	✗	3.5.3
CS-WEB-12.8	Test for Username enumeration	✓	-
CS-WEB-12.9	Test for missing option to change password	✓	-
CS-WEB-12.10	Test for Cross-Origin Resource Sharing misconfiguration	✓	-
CS-WEB-12.11	Test for Privilege Escalation vulnerabilities	✓	-
CS-WEB-12.12	Test for HTML Injection	✓	-
CS-WEB-12.13	Test for Server-Side Request Forgery	✓	-