TRUSTED NETWORK SOLUTIONS

Web Application Penetration Test – Final Report

Prepared for: **Example a.s.**

Version: 1.0

Created by: Trusted Network Solutions, a.s.
Lukáš Nový, Petr Svoboda, Peter Šinaľ
info@tns.cz
www.tns.cz

Date of issue: 26. 05. 2026

Testing period: 18.05.2026 – 22.05.2026

Project ID: RT #38967

Classification: **TLP: AMBER**

TABLE OF CONTENTS

1 General Information

Contact Information

	Auditor	Company
Entity name	Trusted Network Solutions, a.s.	Example a.s.
Registered office	Ptašínského 6, 602 00 Brno	Hvězdova 1429/68, 120 00 Praha 2
Company ID (IČO)	217 71 189	581 27 443
Point of contact	Jan Papica	Daniel Královec
Position	Business Development Manager	*n/a*
E-mail	Jan.papica@tns.cz	Danie.Kralovec@outlook.cz
Phone	+420 737 171 600	+420 775 208 441

Distribution List

Name and surname	Position / Role	Organization	Method of delivery
Marek Skřivánek	*n/a*	Example a.s.	encrypted zip files

Auditor's Statement

This report reflects the state of the tested environment solely as of the testing period stated on the cover page. The Auditor made a reasonable (*best-effort)* attempt to identify security weaknesses within the agreed scope and test type. The objective was to investigate as many high-severity vulnerabilities as possible; weaknesses with a low impact on the tested environment were allocated correspondingly less time.

Given the dynamic nature of IT, the report does not guarantee an exhaustive list of all vulnerabilities. It cannot be ruled out that configuration changes or the emergence of new zero-day vulnerabilities after the conclusion of testing may affect the relevance of the findings presented herein.

Intentionally destructive activities (such as empirically probing the feasibility of achieving a denial of service – DoS) and activities carrying an elevated risk of undesirable side effects or of disrupting the availability of services and processes in the tested environment were excluded from testing.

This document does not constitute a standalone legal instrument – liability terms, the processing of personal data, the method of delivery, and the conditions governing the handling of results are set out in the Work Contract or Purchase Order.

The Auditor declares that it has no conflict of interest in relation to the Company that could compromise the objectivity of the testing performed.

Testing was conducted in accordance with the *OWASP WSTG v4.2*[1] a *PTES*[2] methodologies and in compliance with applicable Czech legislation. The identified vulnerabilities have not been and will not be exploited beyond what is strictly necessary to demonstrate their existence.

[1] https://owasp.org/www-project-web-security-testing-guide/v42/
[2] http://www.pentest-standard.org/index.php/Main_Page

2 EXECUTIVE SUMMARY

Trusted Network Solutions, a.s. (hereinafter referred to as the "Auditor") performed a security assessment of selected public web application – example.com and alfa.example.com with the objective of identifying security weaknesses and evaluating the overall resilience of the application, authentication mechanisms, session management, and supporting infrastructure against common web application threats and misconfigurations.

2.1 Overview of findings

The assessment identified a total of thirteen findings, including one high-severity finding, two medium-severity findings, two low-severity findings, and eight informational findings. No critical vulnerabilities enabling immediate full compromise of the environment were identified.

Overall, the assessed application demonstrates a moderate level of security maturity. The assessment did not reveal critical server-side vulnerabilities or direct mechanisms for immediate compromise of the environment. Nevertheless, the identified findings indicate weaknesses in dependency management, authentication hardening, browser security controls, and secure application configuration practices. In particular, the presence of outdated vulnerable third-party libraries and insufficient protection against credential attacks represent areas requiring prioritized remediation.

Area	Number of vulnerabilities based on severity				
	Critical	High	Medium	Low	Informative
Web applications example.com and alfa.example.com	0	1	2	2	8

Table 1: Overview of findings

2.2 Web applications example.com and alfa.example.com

The most significant finding relates to the use of an outdated version of the DataTables JavaScript library containing multiple publicly known vulnerabilities, including Prototype Pollution and Cross-Site Scripting (XSS) issues. Successful exploitation of these vulnerabilities could allow an attacker to execute arbitrary JavaScript code within the victim's browser, manipulate application behavior, steal session information, or perform actions on behalf of authenticated users. Because the vulnerable component is publicly accessible and relies on a third-party library with known CVEs, this finding represents an elevated risk to the integrity and confidentiality of user interactions with the application. The issue further highlights the importance of maintaining a robust dependency management and patching process for third-party components integrated into the application.

The identified medium-severity findings primarily affect the authentication and password security model of the application. The application does not sufficiently protect accounts against brute-force attacks, as repeated failed authentication attempts do not result in temporary account locking or rate limiting. In

addition, the implemented password policy does not align with current industry recommendations and allows relatively weak passwords to be used. Together, these weaknesses increase the likelihood of successful credential-based attacks, especially when combined with password reuse or previously leaked credentials from external breaches. While no direct authentication bypass vulnerabilities were identified, these shortcomings reduce the overall resilience of the authentication system against automated attacks and targeted account compromise attempts.

The low-severity findings concern session management and browser-side resource integrity protections. The configured session lifetime exceeds recommended inactivity limits, increasing the exposure window in case a valid authenticated session is compromised or left unattended. Additionally, third-party JavaScript resources are included without Subresource Integrity (SRI) protection, which weakens assurance that externally loaded scripts have not been tampered with. Although these findings do not represent immediate high-impact vulnerabilities, they indicate opportunities to strengthen the overall defensive posture of the application and reduce attack surface related to browser-based threats.

The informational findings mainly relate to security hardening, HTTP configuration, browser security controls, and operational configuration quality. These findings include missing Secure and HttpOnly cookie attributes for non-sensitive cookies, information disclosure through HTTP headers, malformed or incomplete Content-Security-Policy directives, missing recommended security headers, incorrect sitemap references, lack of post-quantum cryptography support, and the use of HTTP GET requests for state-changing operations. Individually, these issues present limited direct risk; however, collectively they indicate inconsistent implementation of security best practices and configuration governance across the application environment. Several findings also demonstrate areas where browser-side protections could be improved to strengthen defense-in-depth capabilities against future attacks.

2.3 Overall Assessment and Security Posture

The Auditor also performed a risk rating based on an overall assessment of the vulnerabilities discovered. For the purposes of this analysis we rely on CVSS scoring – that is, the attack complexity required and the potential impact of the vulnerabilities identified.

Overall risk rating: MEDIUM

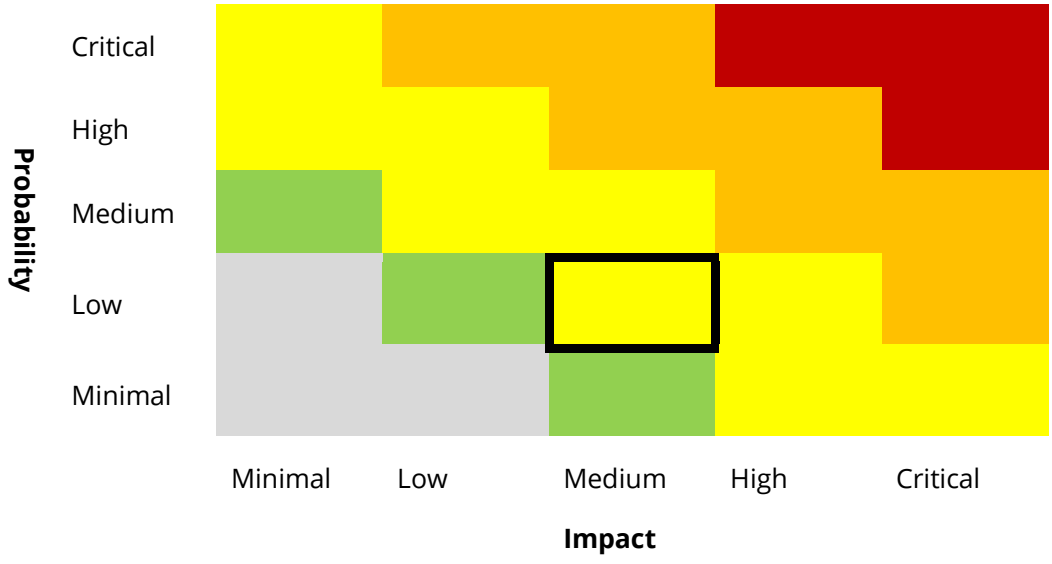

Figure 1: Risk rating heatmap

The overall risk rating can take one of five values: minimal, low, medium, high, and critical. The overall rating correlates with the CVSS scores of the individual vulnerabilities. Grey indicates a minimal risk of compromise, green indicates a low risk of compromise, in which an attacker would have to expend considerable resources to carry out a successful attack. Yellow indicates a medium risk posture, in which the discovered vulnerabilities, combined with other weaknesses outside the attacker's control, could create the conditions for compromising the organization. Orange indicates a high risk posture, describing an environment in which an attacker is able to cause systems to become unavailable or to obtain sensitive information. A security posture marked in red denotes a risk profile in which the vulnerabilities found could be exploited immediately by an attacker to compromise the organization. Further information on the scoring scheme can be found in Chapter **2, Severity Levels**.

2.4 Remediation Measures and Priorities

It is recommended to prioritize the upgrade and ongoing maintenance of vulnerable third-party components, especially the affected DataTables library, and to strengthen authentication security through improved password policies, account lockout mechanisms, and rate limiting controls. Additional recommendations include reducing session lifetimes, implementing missing browser security protections such as SRI and HTTP security headers, correcting CSP configuration issues, and improving configuration validation and deployment review processes.

In the longer term, the organization should continue enhancing its secure development lifecycle practices, maintain regular security assessments, and prepare a roadmap for future cryptographic modernization, including support for post-quantum cryptographic technologies where appropriate.

3 PROCESS AND METHODOLOGICAL FRAMEWORK

3.1 Auditor's Standards and Tools

Methodologies applied:

- CVSS v3.1 for vulnerability scoring
- PTES
- OWASP WSTG v4.2
- NIST SP 800-115 - Technical Guide to Information Security Testing

Auditor's IP address ranges and domains:

- 95.129.96.209, 95.129.96.211-212
- 2a02:e98:20:400::108
- sensei-tns.net.eu

Software:

- Nessus Professional
- Burp Suite Pro
- Nikto
- Skipfish
- Nuclei
- FFUF
- other special and custom tools

3.2 Structure of the Final Report

This final report is structured to serve different target audiences within the organization.

Executive Summary is intended for senior management. It provides a consolidated overview of the number and severity of the identified vulnerabilities without technical detail. It contains sufficient information to understand the current state of affairs and to support strategic decision-making in the area of cybersecurity management.
Technical Analysis is intended for IT administrators, developers, analysts, and auditors. For each finding, this section provides a detailed technical description, evidence of the vulnerability, a severity assessment, and specific remediation measures, including references to relevant sources.

3.3 Management of Risks Associated with Findings

For each finding presented in the technical section of this report, an informed decision must be made on how to treat it. Finding severity is rated on a scale ranging from informational to critical using the CVSS v3.1 methodology, with each finding specifying the attack complexity and the impact on confidentiality, integrity, and availability.

Based on this assessment, the responsible person has the following options:

1. **Risk elimination** – fully removing the vulnerability by implementing the recommended remediation measure (e.g. updating a component, correcting a configuration, or amending code). This is always the preferred approach, particularly for high- and critical-severity findings.

2. **Compensating / alternative controls (mitigation)** – where the vulnerability cannot be eliminated directly (for example, due to a dependency on a legacy system, operational constraints, or disproportionate costs), it is acceptable to introduce alternative security controls that reduce the likelihood of exploitation or limit its impact. Examples include deploying a WAF, network segmentation, tightening monitoring, or restricting access to the vulnerable service.
3. **Risk transfer** (transfer) – transferring responsibility for the financial or operational impact to a third party, typically through cyber-risk insurance or the contractual transfer of liability to the service provider.
4. **Risk acceptance** – a conscious and documented decision to accept the risk without further action. This option is permissible only for low- and informational-severity findings, or where the cost of remediation demonstrably exceeds the potential impact. Acceptance must always be approved and documented at the appropriate management level of the organization, including a justification and a scheduled review date.

3.4 Document Classification – TLP:AMBER

This document is classified in accordance with the Traffic Light Protocol (TLP) version 2.0. Recipients may share this document solely within their own organization, and only with individuals who need to know the information to perform their duties (the "need-to-know" principle). Sharing beyond the recipient's organization is permitted only with the explicit written consent of the Auditor.

The document contains detailed technical information about vulnerabilities, configurations, and security weaknesses in the Company's infrastructure. Unauthorized disclosure or leakage of this information could enable a third party to exploit the vulnerabilities described and pose a direct threat to the security of the tested systems.

Recipients are advised to follow their own internal information-retention policies.

3.5 Severity Levels

To make the test results clearer and to clearly flag both critical and less serious findings, this chapter introduces a set of severity levels for findings.

Each finding describes a security issue or vulnerability uncovered during testing. Findings are evaluated and assigned the corresponding severity level in accordance with the CVSS v3.1 methodology[3]. The score reflects the attack vector, attack complexity, the privileges required of the attacker, the need for user interaction, the severity of impact on other components, and the effect on the confidentiality, integrity, and availability of individual assets.

The following table sets out the vulnerability severity levels together with a description of the risk each represents:

Critical (CVSS: 9.0-10.0)	Vulnerabilities rated critical can be exploited by an attacker immediately and cause damage of considerable significance and scale. Such vulnerabilities are typically characterized as follows: • Their exploitation generally grants control of the application or service at administrator (root) level.

[3] https://www.first.org/cvss/v3.1/specification-document

	• Alternatively, exploitation of the application or service is straightforward, requiring no additional authentication credentials and no specific preconditions to be met.
High (CVSS: 7.0-8.9)	High-severity vulnerabilities can be exploited by an attacker under certain conditions. By exploiting them, an attacker may obtain sensitive information or render the affected systems unavailable. Vulnerabilities of this type can also frequently be leveraged for deeper penetration into the tested network or infrastructure.
Medium (CVSS: 4.0-6.9)	Security flaws rated medium may, in combination with another weakness or vulnerability, create the conditions for a more serious security problem. Their exploitation, however, requires additional specific preconditions outside the attacker's control, or the consequence of exploitation falls within a medium severity range.
Low (CVSS: 0.1-3.9)	Low-severity vulnerabilities do not pose a direct security threat to the operation of the application or infrastructure; however, their presence may disclose details about the application and environment to an attacker, or assist the attacker in completing individual steps of a more complex attack.
Informational (CVSS: 0.0)	Miscellaneous notices and findings of a purely informational nature.

Table 2: Severity levels

To illustrate vulnerability severity more clearly, each finding states the attack complexity (the difficulty of carrying out the attack) and the system impact (the impact on the confidentiality, integrity, and availability of assets in the event of a successful attack). These sub-ratings can each take one of three values (low, medium, high).

3.6 Finding Status

When findings are re-tested, the assessment of each vulnerability can take the following values:

Finding persists	The vulnerability identified during the initial penetration test is still present in the application at the time of the re-test.
Partially remediated	The vulnerability described during the initial test has been partially addressed, although some of its aspects were still found during the re-test.
Remediated	The identified vulnerability has been fully eliminated, and no indications of it were found in the application during the re-test.
Reported / Not tested	The Auditor's notification of the first occurrence of a finding. Alternatively, the finding was excluded from the scope of the re-test.

Table 2: Finding status at re-test.

4 WEB APPLICATIONS

4.1 Assessment scope

The penetration test was carried out against a defined, agreed scope. The scope and conditions of the engagement were as follows:

- **Scope**: Two web applications, accessible at https://www.example.com and https://alfa.example.com. All testing was confined to these targets and their in-scope functionality; no other hosts, subdomains, or systems were assessed.
- **Framework**: A black-box web application penetration test conducted in accordance with the OWASP WSTG v4.2 and PTES methodologies.
- **Prerequisites**: None.
- **Timeframe**: 18 May 2026 to 22 May 2026, with testing performed daily during the agreed window of 12:00 PM to 6:00 AM the following morning.
- **Documentation**: None technical documentation provided.

4.2 List of open ports and running services

IP address (hostname)	Port number and protocol	Service
192.0.2.1	80/tcp	www
198.51.72.14	443/tcp	www
203.0.113.77	2052/tcp	www

Table 1: List of open ports and running services

4.3 Summary of findings

Description	Severity	Complexity	Impact
DataTables < 1.11.3 – Multiple Vulnerabilities	High	Low	High
Insufficient protection of passwords against the brute force attack	Medium	Medium	Medium
Weak password policy	Medium	Medium	Medium
Session lifetime too long	Low	High	Medium
Missing subresource identity protection (SRI)	Low	High	Low
Cookie without HttpOnly flag set	Informative	n/a	n/a
TLS cookie without secure flag set	Informative	n/a	n/a

HTTP – Information Disclosure (Server Type and Version)	Informative	n/a	n/a
SSL/TLS – Remote Services Not Using Post-Quantum Cryptography	Informative	n/a	n/a
Malformed Content-Security-Policy directive	Informative	n/a	n/a
Unused security headers	Informative	n/a	n/a
Incorrect sitemap reference in robots.txt	Informative	n/a	n/a
State-Changing Operation Performed via HTTP GET Method	Informative	n/a	n/a

Table 2: List of detected vulnerabilities

4.4 List of findings

4.4.1 DataTables < 1.11.3 – Multiple Vulnerabilities

Risk level:	**High** CVE-2020-28458 – 7.3 High CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L CVE-2021-23445 – 3.1 Low CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Attack complexity: Low	**System Impact**: High
Affected component:	https://alfa.example.com/fileadmin/js/plugin.min.js
Quick summary:	The application uses an outdated version of the DataTables JavaScript library (prior to 1.11.3) that is affected by known Prototype Pollution (CVE-2020-28458) and Cross-Site Scripting (CVE-2021-23445) vulnerabilities. Successful exploitation could allow an attacker to execute arbitrary JavaScript in users' browsers and tamper with client-side application behavior, potentially leading to session compromise.
Description:	The DataTables JavaScript library (jquery.datatables) is, based on the detected version, prior to version 1.11.3. This version is affected by multiple known vulnerabilities: • insufficient sanitization of user-controlled input allows a *Prototype Pollution* attack, where an attacker may modify properties of JavaScript object prototypes within the application runtime environment. The vulnerability was originally incompletely patched and subsequently further addressed in later library versions (CVE-2020-28458), • additional fixes related to *Prototype Pollution* were introduced in versions 1.10.22 and 1.10.23, • versions prior to 1.11.3 are also affected by a potential *Cross-Site Scripting* (*XSS*) vulnerability, which may allow execution of untrusted JavaScript code

	in the victim's browser due to improper sanitization of processed input data (CVE-2021-23445). The vulnerable library was identified at the following URL: https://alfa.example.com/fileadmin/js/plugin.min.js
Impact:	Successful exploitation of the identified vulnerabilities may allow an attacker to: • manipulate JavaScript object behavior and compromise the integrity of the client-side application through *Prototype Pollution*, • execute arbitrary JavaScript code in the context of the affected application (*XSS*), • steal user session tokens or other sensitive information, • manipulate content displayed to users, • perform actions on behalf of authenticated users. When combined with other application weaknesses, these vulnerabilities may lead to full compromise of affected user sessions.
Recommendation:	Update the DataTables library to the latest stable version or at minimum to version 1.11.3 to remediate the identified vulnerabilities. Additionally, it is recommended to: • verify whether vulnerable versions of the library are bundled within other JavaScript assets, • implement strict input validation and sanitization on both server and client side.
Additional resources:	https://nvd.nist.gov/vuln/detail/CVE-2020-28458 https://nvd.nist.gov/vuln/detail/CVE-2021-23445 https://cdn.datatables.net/1.10.22/ https://cdn.datatables.net/1.10.23/ https://cdn.datatables.net/1.11.3/ https://github.com/DataTables/DataTablesSrc/commit/a51cbe99fd3d02aa5582f97d4af1615d11a1ea03 https://github.com/DataTables/Dist-DataTables/commit/59a8d3f8a3c1138ab08704e783bc52bfe88d7c9b

4.4.2 Insufficient protection of passwords against the brute force attack

Risk level:	**Medium** (6.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N**)**
Attack complexity: Medium	**System Impact**: Medium
Affected component:	https://alfa.example.com
Quick summary:	The application does not enforce any account lockout mechanism, allowing an unlimited number of failed login attempts (over 20 were tested without any blocking). This enables an attacker who knows a valid username to attempt brute force or dictionary attacks to recover the account password.

Description:	During the test, it was found that even 20 attempts to log in with an invalid password will not cause the account to be blocked.
Impact:	An attacker with knowledge of a valid login name is able to obtain a valid user account password through a brute force attack or a dictionary attack.
Recommendation:	Perform (at least temporary) blocking of a user account after a predetermined number of failed logins.
Additional resources:	https://kennel209.gitbooks.io/owasp-testing-guide-v4/content/en/web_application_security_testing/testing_for_weak_lock_out_mechanism_otg-authn-003.html

4.4.3 Weak password policy

Risk level:	**Medium** (4.8 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N)	
Attack complexity: Medium		**System Impact**: Medium
Affected component:	https://alfa.example.com	
Quick summary:	The application's password policy (8-character minimum with complexity rules) does not meet the OWASP ASVS 4.0.3 recommendations, most notably the 12-character minimum length. This weak policy increases the risk of passwords being compromised through dictionary, brute-force, or offline cracking attacks.	
Description:	The web application states its password as follows: • minimum password length is 8 characters • minimum of 1 uppercase letter • minimum of 1 lowercase letter • minimum of 1 special character These requirements are inconsistent with the requirements of the OWASP Application Security Verification Standard (version 4.0.3 chapter V2.1), which imposes the following conditions on passwords: • minimum password length is 12 characters, • passwords can be more than 64 characters, but not more than 128 characters. • passwords are not shortened before processing, only combining multiple spaces in a row is allowed. • all printable Unicode characters are accepted, including spaces and emojis. • simple and most frequently used passwords are not accepted. • no other conditions are required for password complexity (e.g. number of lower/uppercase letters, numbers, etc.).	
Impact:	The password policy used by the application can be considered weak. As a result, password secrecy is at risk in the case of a dictionary attack, a brute-force attack, or reverse guessing in the event of an encrypted form of passwords being leaked.	

Recommendation:	Adjust the password generation requirements as specified in the OWASP Application Security Verification Standard.
Additional resources:	https://owasp.org/www-project-application-security-verification-standard/

4.4.4 Session lifetime too long

Risk level:	**Low** (3.7 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N)	
Attack complexity: High		**System Impact**: Medium
Affected component:	https://alfa.example.com/	
Quick summary:	The application's session lifetime is configured for too long, exceeding the inactivity period mandated by PSD2. An attacker who obtains a valid session token could therefore act with the authorized user's privileges for an extended window of time.	
Description:	Session lifetime is set for too long (> 5 minute as specified by PSD2).	
Impact:	An attacker with access to a valid session secret is is able to perform actions with the authority of the authorized user for a long period of time.	
Recommendation:	Adjust the session length to a maximum of 30 minutes of user inactivity.	
Additional resources:	https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32015L2366	

4.4.5 Missing subresource identity protection (SRI)

Risk level:	**Low** (3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N)	
Attack complexity: High		**System Impact**: Low
Affected component:	https://alfa.example.com	
Quick summary:	The application loads a third-party script (Twitter widgets.js) without Subresource Integrity (SRI) verification. If the third-party source were compromised or modified, malicious code could be executed in users' browsers, potentially leading to data exfiltration.	
Description:	The application includes following 3rd party scripts without checking subresource identity (SRI): • https://platform.twitter.com/widgets.js	
Impact:	The application could be vulnerable to an ex-filtration attack.	

Recommendation:	Protect the included script with subresource integrity protection (SRI).
Additional resources:	http://www.nessus.org/u?c9e76c4f http://www.nessus.org/u?f39144f8 https://www.w3.org/TR/SRI/

4.4.6 Cookie without HttpOnly flag set

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	https://www.example.com/ https://alfa.example.com	
Quick summary:	Numerous cookies issued across both websites are set without the HttpOnly flag, leaving their values readable by client-side JavaScript. This makes the affected cookies susceptible to theft or modification if an attacker manages to inject malicious script into the application, though the impact is limited as these cookies do not carry sensitive data.	
Description:	If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected script. The following cookies were issued by the application and does not have the HttpOnly flag set: www.example.com • kkl • tev • dof • ogpd alfa.example.com: • _vr.vyovl_geze • _bq.f • _BTGZ • _ghp • _de • _de_BE8GC3T85Z • _de_JI608Q9BAD • _cfg_uy • _jkFuccoiymTrun_1984262 • _yazdyb • _povdaq • DQFIKC • SADEMIKASQ • DSASEFHR • QETGGZHNGBND • badx • erfiq_	

	• glRaajli • crCP • etScr • bsNMMiqP8h • kr_d • as_w • tec • teEpireit • rtzw • div • abze **Rem.**: *Severity of this finding was lowered as listed cookies are not used for sensitive data.*
Impact:	The web application does not use all the available means to secure the transmitted data, exposing it to an increased risk of interception or modification. An attacker capable of inserting malicious JavaScript code into the client side of the application can modify or steal the cookie.
Recommendation:	There is usually no good reason not to set the HttpOnly flag on all cookies. Unless you specifically require legitimate client-side scripts within your application to read or set a cookie's value, you should set the HttpOnly flag by including this attribute within the relevant Set-cookie directive. You should be aware that the restrictions imposed by the HttpOnly flag can potentially be circumvented in some circumstances, and that numerous other serious attacks can be delivered by client-side script injection, aside from simple cookie stealing.
Additional resources:	https://capec.mitre.org/data/definitions/31.html https://cwe.mitre.org/data/definitions/16.html https://portswigger.net/research/web-storage-the-lesser-evil-for-session-tokens#httponly https://portswigger.net/web-security/cross-site-scripting/exploiting

4.4.7 TLS cookie without secure flag set

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	https://example.com/ https://alfa.example.com	
Quick summary:	Multiple cookies on both websites are set without the Secure flag, meaning they may be transmitted in clear-text over unencrypted HTTP connections. An attacker positioned to eavesdrop on the victim's network traffic could intercept these cookies, though the impact is limited as they do not carry sensitive data.	

Description:	If the Secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the Secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. The following cookies are set without the Secure flag: www.example.com: • _dfg • _by • _br_ES654E2BBD • _hvt_ui • _qantum • _deskop • deAbCd567Q • SDQERV • QERTWCVD • nFEQWqiW0s • ers alfa.example.com • _ni.kjakd_repq • _fs.e • _SQTE • _req • _dr • _de_AD5QR8A51A • _by_AR124A1BSD • _rtf_re • _fteqda • _etffak • QERART • RSADGHJB • dapd • rades_ • nAIBEuwQ1q • dis • sedf **Rem.**: *Severity of this finding was lowered as listed cookies are not used for sensitive data.*
Impact:	An attacker may be able steal user cookie by feeding a user suitable links, either directly or via another web site. Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form http://example.com:443/ to perform the same attack. To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on 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.

Recommendation:	The Secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications.
Additional resources:	https://cwe.mitre.org/data/definitions/614.html

4.4.8 HTTP – Information Disclosure (Server Type and Version)

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	192.0.2.1 (alfa.example.com) 203.0.113.77 (www.example.com)	
Quick summary:	The HTTP server responses disclose the underlying server software via the Server header (revealing Cloudflare, BigIP, and Apache). This information could help an attacker fingerprint the technology stack and tailor further attacks toward known vulnerabilities in those products.	
Description:	It was possible to determine the software that is used as HTTP server from it's responses. 192.0.2.1 (alfa.example.com) 203.0.113.77 (www.example.com) Server: cloudflare 192.0.2.1:80/tcp (alfa.example.com) Server: BigIP 203.0.113.77 443/tcp (example.com) Server: Apache	
Impact:	Generally, any information about software can be used by the attacker to better aim further attacks or more easily identify known vulnerabilities.	
Recommendation:	Remove the *Server* header from server responses to make identifying vulnerabilities more difficult, but note, that this also makes identifying vulnerabilities more difficult during security audits and vulnerabilities still needs to be patched even if information about software is hidden.	
Additional resources:		

4.4.9 SSL/TLS – Remote Services Not Using Post-Quantum Cryptography

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	203.0.113.77:443/tcp (example.com)	
Quick summary:	The remote services rely solely on classical public-key cryptography (RSA, DH/ECDH, ECDSA) and do not implement any post-quantum or hybrid key exchange mechanisms. While there is no immediate threat, the encrypted traffic is exposed to "harvest now, decrypt later" attacks, putting data with long-term confidentiality requirements at future risk once quantum computers become viable.	
Description:	The remote services rely exclusively on **classical public-key cryptography** whose security is based on the computational difficulty of the **integer factorization problem** or the **discrete logarithm problem**. These assumptions are invalidated in the presence of a sufficiently powerful **quantum computer** due to **Shor's algorithm**. Cryptographic mechanisms affected include, but are not limited to: • **RSA** asymmetric encryption and signatures • **Diffie-Hellman (DH)** and **Elliptic Curve Diffie-Hellman (ECDH)** key exchange • **Elliptic Curve Digital Signature Algorithm (ECDSA)** The service does not currently implement **post-quantum (quantum-resistant) cryptographic algorithms**, nor hybrid key exchange mechanisms that combine classical and post-quantum algorithms.	
Impact:	At present, there is **no immediate practical risk**, as cryptographically relevant quantum computers are not publicly available. However, encrypted traffic protected solely by classical algorithms is vulnerable to **"harvest now, decrypt later"** attacks, in which an attacker records encrypted communications today with the intention of decrypting them once quantum capabilities become available. This risk is particularly relevant for data with **long-term confidentiality requirements**, such as: • Authentication credentials • Personal or sensitive data • Proprietary or regulated information The absence of post-quantum cryptography may also lead to **future non-compliance** with evolving regulatory and industry standards as post-quantum algorithms become mandatory.	
Recommendation:	Develop and implement a **post-quantum cryptography transition strategy**. As an initial step, enable **hybrid key exchange mechanisms** that combine classical algorithms with **NIST-recommended post-quantum algorithms** (e.g., ML-KEM/Kyber) where supported by the platform and client ecosystem.	

	Monitor guidance from standardization bodies such as **NIST** and **ETSI**, and plan for the phased replacement of vulnerable classical algorithms with fully post-quantum-secure alternatives once they are widely supported and production-ready. For systems handling long-lived or highly sensitive data, prioritize early adoption of post-quantum or hybrid cryptographic solutions.
Additional resources:	https://ntruprime.cr.yp.to/ntruprime-20170816.pdf https://www.ietf.org/archive/id/draft-kwiatkowski-pquip-pqc-migration-00.html https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards https://www.rd.ntt/e/research/JN202305_21871.html

4.4.10 Malformed Content-Security-Policy directive

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	https://www.example.com/	
Quick summary:	The application returns a Content-Security-Policy header containing a malformed img-src directive (*.ggpht), which does not conform to valid CSP syntax and is silently ignored by compliant browsers. This weakens the reliability of the application's browser-side security controls and reduces the intended protection against content injection and XSS attacks.	
Description:	The application returns a malformed Content-Security-Policy (*CSP*) header. The img-src directive contains an invalid source expression: `img-src *.llppt` The specified value does not conform to the CSP syntax requirements defined by the specification. In CSP, host source values must include a valid hostname format and, where applicable, a scheme. The value *.llppt is therefore ignored by compliant browsers. Malformed CSP directives may result in browsers silently discarding affected policy components, reducing the effectiveness of the intended security protections.	
Impact:	An incorrectly defined CSP may lead to: • unintended loading of resources due to ignored policy directives, • weakened protection against content injection attacks, • reduced mitigation effectiveness against *Cross-Site Scripting* (*XSS*) attacks, • inconsistent CSP behavior across different browsers.	

	Although the malformed directive alone does not directly introduce a vulnerability, it weakens the reliability of the application's browser-side security controls.
Recommendation:	Correct the malformed img-src directive to use valid CSP syntax. For example: `img-src *.llppt.com` Additionally, it is recommended to: • validate the complete Content-Security-Policy header syntax before deployment, • minimize the use of wildcard source definitions where possible, • regularly test CSP policies using browser developer tools or CSP validation utilities, • enable Content-Security-Policy-Report-Only during policy tuning and monitoring phases.
Additional resources:	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy https://w3c.github.io/webappsec-csp/ https://content-security-policy.com/

4.4.11 Unused security headers

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	https://alfa.example.com	
Quick summary:	The web application does not implement two recommended security HTTP headers: Clear-Site-Data and Permissions-Policy. Their absence means the application forgoes available browser-side protections, increasing exposure to risks such as session fixation and abuse of sensitive browser APIs (camera, microphone, geolocation).	
Description:	Modern web browsers and other web clients generally support a range of standardized "security" HTTP headers. These headers allow the web server to define various rules for client-server communication. These rules help prevent certain types of attacks or the exploitation of potential vulnerabilities in web applications. The tested web application does not use the following security headers recommended by the auditor: • Clear-Site-Data • Permissions-Policy	
Impact:	The web application does not make use of available methods of protection against potential attacks or the exploitation of possible application vulnerabilities.	

Recommendation:	**Clear-Site-Data**
	Used to clear specific types of data stored in the browser (cookies, cache, localStorage, etc.), minimizing the risk of attacks such as session fixation or side-channel attacks.
	
	Example configuration (send this when a user logs out or when their permissions change):
	`Clear-Site-Data: "cache", "cookies", "storage", "executionContexts"`
	
	Permissions-Policy
	Allows restricting website access to sensitive APIs such as camera, microphone, geolocation, etc., thereby reducing the risk of abuse.
	
	Example configuration:
	`Permissions-Policy: geolocation=(), microphone=(), camera=(), payment=()`
	
	You can also check the configuration of these and other security headers for publicly available servers using these online tools:
	• https://developer.mozilla.org/en-US/observatory
	• https://securityheaders.com/

Additional resources:	https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Clear-Site-Data https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Content-Security-Policy https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Permissions-Policy https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Referrer-Policy https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Content-Type-Options https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP https://csp-evaluator.withgoogle.com/ https://developer.mozilla.org/en-US/observatory https://securityheaders.com/

4.4.12 Incorrect sitemap reference in robots.txt

Risk level:	**Informative**	
Attack complexity: N/A		**System Impact**: N/A
Affected component:	https://alfa.example.com/robots.txt	
Quick summary:	The robots.txt file at alfa.example.com references a sitemap belonging to an unrelated domain (https://www.example.com/sitemap.xml). While not a direct security vulnerability, this misconfiguration can lead to incorrect search engine	

	indexing and indicates gaps in deployment validation and configuration management.
Description:	The robots.txt file of the application hosted at https://alfa.example.com contains a reference to an incorrect sitemap location: https://www.example.com /sitemap.xml The referenced sitemap belongs to a different domain and does not correspond to the assessed application. As a result, automated crawlers and search engines may process unrelated indexing information. Incorrect sitemap references may lead to incomplete or unintended indexing behavior and indicate insufficient configuration management or deployment validation.
Impact:	This issue may result in: • incorrect or incomplete indexing of the application's content by search engines, • reduced visibility of legitimate application pages in search engine results, • unintended association of the application with unrelated domains or content, While the issue does not directly introduce a security vulnerability, it may negatively affect search engine optimization (*SEO*), operational integrity, and overall configuration quality.
Recommendation:	Update the robots.txt configuration to reference the correct sitemap file associated with the application domain. Additionally, it is recommended to: • verify the validity and accessibility of all sitemap references, • include deployment validation checks for environment-specific configuration files, • periodically review publicly exposed configuration files for outdated or incorrect references.
Additional resources:	https://developers.google.com/search/docs/crawling-indexing/robots/create-robots-txt https://www.sitemaps.org/protocol.html

4.4.13 State-Changing Operation Performed via HTTP GET Method

Risk level:	**Informative**
Attack complexity: N/A	**System Impact**: N/A
Affected component:	https://alfa.example.com/nastroje/tools/

Quick summary:	The application uses HTTP GET requests to perform state-changing operations, contrary to the HTTP specification which reserves GET for safe, read-only actions. This increases the risk that sensitive operations could be triggered unintentionally by browsers, crawlers, or prefetching mechanisms, although the impact is reduced as all requests are protected by anti-CSRF tokens.
Description:	The application uses the HTTP GET method to perform operations that modify the application state. According to HTTP specification and secure development best practices, the GET method should only be used for safe and idempotent read-only operations. The identified behavior allows state-changing actions to be triggered through a simple URL request. Such requests may be unintentionally executed by browsers, crawlers, prefetching mechanisms, third-party websites, or automatically loaded resources. Using GET requests for actions such as account changes, transaction processing, object modification, or deletion increases the risk of unauthorized or unintended execution of sensitive operations. This issue is commonly associated with insufficient protection against *Cross-Site Request Forgery* (*CSRF*) attacks and improper HTTP method usage. **Rem.:** *Severity of this finding was lowered as all request are protected with a anti-csrf tokens.*
Impact:	An attacker may be able to: • trick authenticated users into unintentionally performing sensitive actions by visiting a malicious webpage, • force execution of state-changing operations through embedded content such as images, iframes, or links, • abuse browser prefetching, caching, or automated crawlers to trigger unintended requests, • bypass security controls that rely on HTTP method validation. Depending on the affected functionality, successful exploitation may lead to unauthorized modification of application data, privilege misuse, or unintended transactions.
Recommendation:	Use the HTTP POST, PUT, PATCH, or DELETE methods for all operations that modify application state. Additionally, it is recommended to: • implement anti-CSRF protections for all state-changing requests, • validate the Origin and Referer HTTP headers where appropriate, • ensure sensitive operations require explicit user interaction and confirmation, • configure HTTP caching policies to prevent unintended replay of sensitive requests.

Additional resources:	https://owasp.org/www-community/attacks/csrf https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html https://datatracker.ietf.org/doc/html/rfc9110#section-9.2.1