OUTSCAN Vulnerability Detailed Report

2025-10-13

Report Information

Report type	Vulnerability
Assessment types	Web application
Report ID	4486CA81AF1C3066B266C0069CA9CF64
Date report was created	2025-10-13 13:00
Timezone for dates	GMT+2
Report created for	Troostwijk Titania Holding 2 B.V.
Report generated by	Jasper van der Straten (Scheduled)
Report interval	2025-07-13 02:00 - 2025-10-14 01:59
Number of risks found	12
Report filtering	Report has filtering active and this can result in a report with partial findings.
Findings sorted by	Risk level

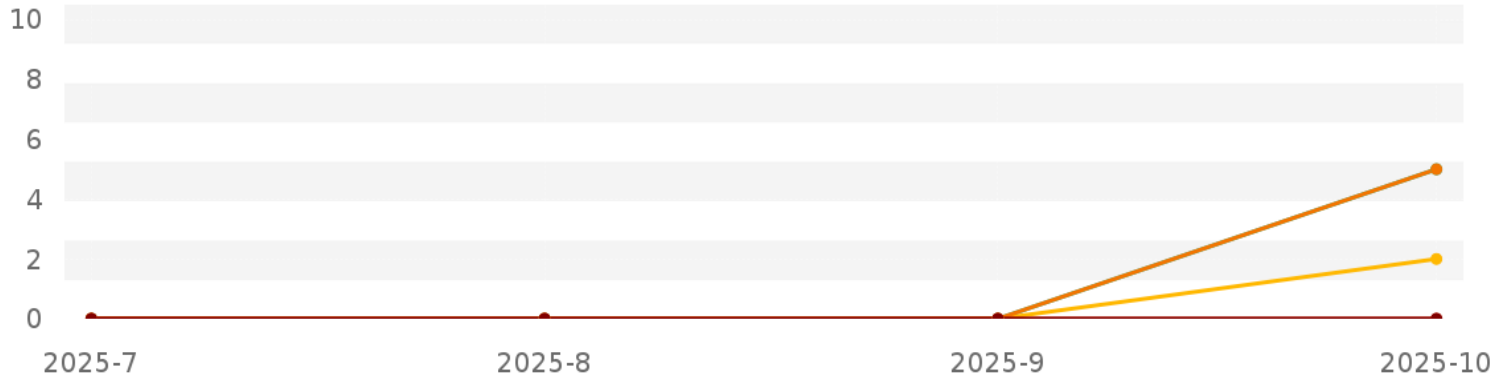
Executive Summary

Risk Level Overview

Critical	0 findings
High	0 findings
Medium	5 findings
Low	2 findings
Recommendation	5 findings

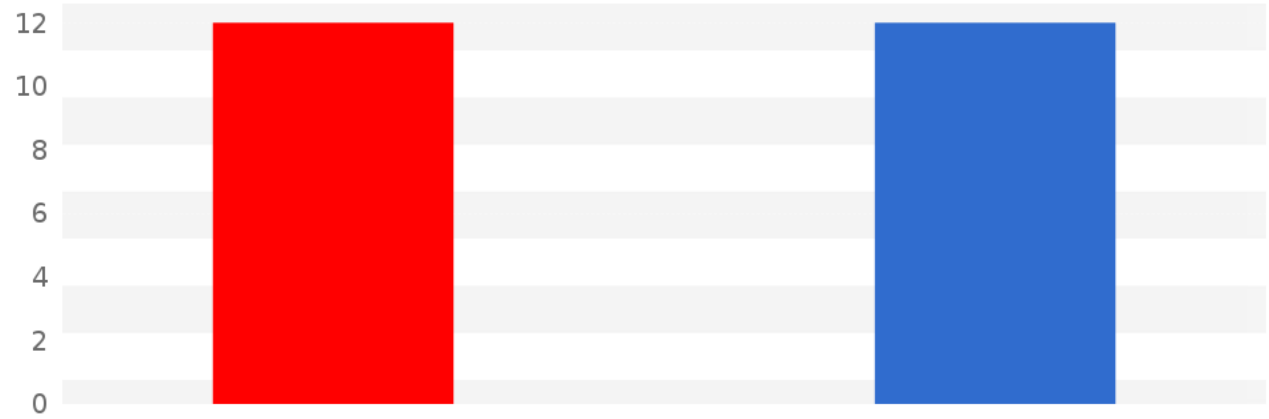
Trend

Number of findings for each risk level between 2025-07-13 and 2025-10-13



Delta Overview

Number of added, removed and unchanged findings between 2025-07-13 and 2025-10-14



Added	12 findings
Removed	12 findings
Unchanged	0 findings

Top 10 Findings

Name	Open Issues	Asset Groups
Component "AngularJS 1.8.3" End-of-Life	1	1
Component "Bootstrap 4.0.0" End-of-Life	1	1
Component "jQuery 2.2.2" End-of-Life	1	1
HTTP Strict Transport Security not Configured	1	1
Missing Content Security Policy	1	1
Missing "X-Content-Type-Options: nosniff" Response Header	1	1
Preload for HTTP Strict Transport Security not Configured	1	1
Sensitive Cookie "JSESSIONID" not set as SameSite	1	1
Sensitive Cookie "JSESSIONID" not set as Secure	1	1
Session Fixation	1	1

OWASP Top 10 2021

A01	Broken Access Control	✖
A02	Cryptographic Failures	✔
A03	Injection	✔
A04	Insecure Design	✖
A05	Security Misconfiguration	✖
A06	Vulnerable and Outdated Components	✖
A07	Identification and Authentication Failures	✖
A08	Software and Data Integrity Failures	✔
A09	Security Logging and Monitoring Failures	–
A10	Server-Side Request Forgery (SSRF)	✔

✔ No issues
✖ One or more issues
– Can't be assessed externally

Auksjonen

As of 2025-10-06, Outpost24's Ghost Labs AppSec Team has concluded a review of the web application that is continuously assessed using the SWAT service for TB Auctiones Netherlands BV". The goal of this project is to determine the security posture of the "Auksjonen"-application and deliver the results through this report.

Overall, the application is considered to have a moderate security posture, with several medium-severity issues requiring attention. Most findings relate to outdated components and session management weaknesses, which should be addressed to strengthen the platform's overall resilience.

During testing, a Session Fixation issue was identified, where session identifiers were not properly regenerated after login. This could, in specific cases, allow an attacker to hijack a valid user session. User Enumeration was also possible during account registration, as the application's responses differed when an existing username or email was submitted. Furthermore, the libraries "jQuery", "AngularJS", and "Bootstrap" were all found to be end-of-life (EOL) and no longer receive security updates. Upgrading to supported versions is recommended to prevent potential exposure to known vulnerabilities.

In addition, the "JSESSIONID" session cookie lacked both the "Secure" and "SameSite" attributes, slightly reducing protection against session theft and cross-site request forgery (CSRF). Beyond that, only minor improvements were

identified, such as implementing the “Content-Security-Policy” (CSP) and “HTTP Strict-Transport-Security” (HSTS) headers to enhance the application’s resilience against common web attacks.

In conclusion, concise remediation and recommendations are provided for each finding, with the intention of enhancing the security standing to a higher level.

Auksjonen

	CVSS v2	CVSS v3
Critical		0
High	0	0
Medium	5	5
Low	2	2
Recommendation	5	5
Not classified	0	0
Average CVSS score	2.5	3.0

OWASP Top 10 2021

A01	Broken Access Control	✖
A02	Cryptographic Failures	✔
A03	Injection	✔
A04	Insecure Design	✖
A05	Security Misconfiguration	✖
A06	Vulnerable and Outdated Components	✖
A07	Identification and Authentication Failures	✖
A08	Software and Data Integrity Failures	✔
A09	Security Logging and Monitoring Failures	–
A10	Server-Side Request Forgery (SSRF)	✔

✔ No issues
✖ One or more issues
– Can't be assessed externally

Active Subscriptions SWAT

Component "AngularJS 1.8.3" End-of-Life

Risk factor	▲ Medium This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with some impact on confidentiality, some impact to the integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	5.0 - (AV:N/AC:L/Au:N/C:N/I:P/A:N)
CVSS v3	6.1 - (AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N)
Description	The component AngularJS is no longer supported by the vendor, as it has reached its end-of-life. This means that the component will no longer receive regular updates nor any security patches from the vendor. Note that this is solely based on the detected version of the component, it is possible that custom software patches have been applied as a mitigation technique. Alternatives, such as different frameworks or third party support, are available as a solution for the ending of security updates releases from the AngularJS project. In addition, several versions of AngularJS are affected by known Cross-Site Scripting and Prototype pollution issues due to many developers overseeing the risk of allowing users to control parts of the AngularJS templating or expressions. For reference, please see: https://docs.angularjs.org/misc/version-support-status https://blog.angular.io/discontinued-long-term-support-for-angularjs-cc066b82e65a
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	A malicious actor may be able to stage attacks using known vulnerabilities in the component, as the component will not receive any security updates.
Recreation Flow	1. Visit the following URL: https://www.auksjonen-test.no/ 2. Open the browser's built in developer tools 3. Enter the following string into the JavaScript console: angular.version.full 4. Note that the application includes an End-of-Life version of AngularJS which has known security issues associated with it
Solution	Updating is not sufficient as the component is end-of-life. Migrate to a supported replacement.
Status	present
CWE	CWE-1035
OWASP Top 10 2013	A09: Using components with known vulnerabilities
OWASP Top 10 2017	A09: Using Components with Known Vulnerabilities
OWASP Top 10 2021	A06: Vulnerable and Outdated Components
Age	12 days - First seen: 2025-10-01 09:09, Last seen: 2025-10-01 09:22
Tags	, auksjonen
Attachments	AngularJS 1,8.3.png — See appendix 1.1

Component "Bootstrap 4.0.0" End-of-Life

Risk factor	▲ Medium This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with some impact on confidentiality, some impact to the integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	4.3 - (AV:N/AC:M/Au:N/C:N/I:P/A:N)
CVSS v3	6.1 - (AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N)
Description	The detected version 4.0.0 of Bootstrap is no longer supported by the vendor, as it has reached its end-of-life. This means that the component will no longer receive regular updates nor any security patches from the vendor. In addition, several minor versions within the major 4.x of Bootstrap have known issues associated with them, including but not limited to issues with input processing, which in some cases may result in Cross-Site Scripting. For reference, please see: https://github.com/twbs/release The CVE identifiers associated with this version are: CVE-2018-14040 CVE-2018-14041 CVE-2018-14042
URL	https://www.auksjonen-test.no/info/CybotCookiebotDialogBodyLevelButtonMarketing
HTTP method	GET
Port	443
Impact	A malicious actor may be able to stage attacks using known vulnerabilities in the component, as the component will not receive any security updates.
Recreation Flow	1. Visit the following URL: https://www.auksjonen-test.no/info/CybotCookiebotDialogBodyLevelButtonMarketing 2. Open the browser's built in developer tools 3. Enter the following string into the JavaScript console: jQuery.fn.collapse.Constructor.VERSION 4. Note that the application includes an End-of-Life version of Bootstrap, which has known security issues associated with it
Solution	Upgrade Bootstrap to the latest secure version.
Status	present
CWE	**CWE-1035**
OWASP Top 10 2013	**A09: Using components with known vulnerabilities**
OWASP Top 10 2017	**A09: Using Components with Known Vulnerabilities**
OWASP Top 10 2021	**A06: Vulnerable and Outdated Components**
Age	9 days - First seen: 2025-10-03 13:26, Last seen: 2025-10-03 14:06
Tags	, auksjonen
Attachments	Component_Bootstrap_4.0.0_End-of-Life.jpg See appendix 1.1

User Enumeration via Account Registration

Risk factor	▲ Medium This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with some impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	5.0 - (AV:N/AC:L/Au:N/C:P/I:N/A:N)
CVSS v3	5.3 - (AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N)

Description	User enumeration allows an attacker to determine whether or not a specific user account is registered on an application. This can occur when an application provides different responses for known and unknown usernames or email addresses, which is often the result of a compromise between security and convenience. From a security standpoint, the application should provide an identical response for both known and unknown usernames or email addresses, such as displaying a generic error message instead of a specific one indicating that the username or email address is not known. For more information on authentication and its potential vulnerabilities, check out our blog post: https://outpost24.com/blog/authentication-vulnerabilities-web-applications/
URL	https://www.auksjonen-test.no/api/registration/verify-step-1
HTTP method	POST
POST Data	{"countryCode":"NO","cellphone":"97178250","birthDate":"829699200000","email":"swatuser2@outpost24.com","email2":"swatuser2@outpost24.com","firstName":"swat","lastName":"user","address1":"Fake","zipCode":"1231","city":"123","type":1}
Port	443
Impact	An attacker could enumerate users registered on the application, which in turn could be used for brute-force attacks.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no/registrer/privat?phone=1234 2. Enter a registered email into the "Brukernavn (E-post)" field and press "Neste" (testing used email "swatuser2@outpost24.com") 3. Note the resulting error message below the "Brukernavn (E-post)" field stating: Epost er allerede registrert. 4. Repeat step #2, this time using an email that is not registered with the application (testing used email "fakeaccount@outpost24.com") 5. Note that there is now no error message below the "Brukernavm (E-post)" field 6. This difference can be used in order to check whether or not a certain email is registered with the application
Solution	Make sure the application behaves and responds identically to the end client whether or not the queried user account exists.
Status	present
CWE	**CWE-203**
CAPEC	**118: CAPEC-118**
OWASP Top 10 2004	A07: Improper Error Handling
OWASP Top 10 2007	A06: Information Leakage and Improper Error Handling
OWASP Top 10 2010	A06: Security Misconfiguration
OWASP Top 10 2013	**A06: Sensitive Data Exposure**
OWASP Top 10 2017	**A03: Sensitive Data Exposure**
Age	10 days - First seen: 2025-10-02 13:11, Last seen: 2025-10-02 13:54
Tags	, auksjonen
Attachments	Registered Email.png — See appendix 1.1 Unregistered Email.png — See appendix 1.2

Session Fixation

Risk factor	▲ Medium This vulnerability can be exploited with advanced skills and network access to the system by an attacker who does not have access to credentials with some impact on confidentiality, some impact to the integrity of information and some impact on system or information availability. Successful attacks require human interaction from a person other than the attacker. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	5.1 - (AV:N/AC:H/Au:N/C:P/I:P/A:P)
CVSS v3	5.0 - (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:L)

Description	Session fixation happens when the server does not generate a new session ID upon login for the newly authenticated session, instead opting to use the existing session. A malicious user could use this to get another user to validate a session ID known to them, thus gaining access to their session. The application does not renew the session token when the user authenticates. This becomes the underlying issue for Session Fixation, where an attacker can pre-set the token of a user to a known value, and then become authenticated together with the user upon them logging in. This attack is not trivial to perform, as it requires either direct access to the target computer, or access to another attack vector such as a Cross-Site Scripting attack.
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	An attacker may be able to gain unauthorized access to user accounts by gaining access to non-authenticated session tokens. This finding requires authentication in order to be reproduced.
Recreation Flow	1. Open two separate web browsers that do not share sessions (or, for example, use one normal window and one private window). Pick one window to be the attacker, and one to be the victim 2. In the attacker window, navigate to the login page: https://www.auksjonen-test.no/login 3. Take a note of the session cookie "JSESSIONID": JSESSIONID=IUWNfbrthgPjzzfLaCFxphOFdkQuKmfaaxmSb0jW 4. Using the victim window, navigate to the login page from step #2 5. Modify the value of the victim's JSESSIONID cookie, and set it to the value of the attacker (noted in step #3): Original victim cookie: JSESSIONID=rFYRlg4WffF7NF680hpkKH_j-5OlwqgbZaIlFHGm Modified victim cookie: JSESSIONID=IUWNfbrthgPjzzfLaCFxphOFdkQuKmfaaxmSb0jW 6. Remove the victims "XSRF-TOKEN" cookie, and refresh the page 7. Still in the victim window, log in as the victim (testing used account "swatuser1@outpost24.com"). Note that the session cookie is not regenerated and renewed upon authenticating 8. Using the attacker window, go to the following URL: https://www.auksjonen-test.no/bruker 9. Note how the malicious user is now also authenticated as the victim
Solution	Make sure that the session cookie is renewed with a new value after a successful login.
Status	present
CWE	CWE-384
CAPEC	196: CAPEC-196 21: CAPEC-21 39: CAPEC-39 59: CAPEC-59 60: CAPEC-60 61: CAPEC-61 31: CAPEC-31
OWASP Top 10 2004	A03: Broken Authentication and Session management
OWASP Top 10 2010	A03: Broken Authentication and Session Management
OWASP Top 10 2013	A02: Broken Authentication and Session Management
OWASP Top 10 2017	A02: Broken Authentication
OWASP Top 10 2021	A07: Identification and Authentication Failures
Age	10 days - First seen: 2025-10-03 09:23, Last seen: 2025-10-03 09:40
Tags	, auksjonen

Component "jQuery 2.2.2" End-of-Life

Risk factor	▲ Medium This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with some impact on confidentiality, some impact to the integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	5.0 - (AV:N/AC:L/Au:N/C:N/I:P/A:N)
CVSS v3	6.1 - (AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N)
Description	The detected version of jQuery is no longer supported by the vendor, as it has reached its end-of-life. This means that the component will no longer receive regular updates nor any security patches from the vendor. Note that this is solely based on the detected version of the component; it is possible (although very unlikely) that custom software patches have been applied as a mitigation technique. jQuery has a known issue with how it handles content received via $.get() since version 1.4.0. It can by default interpret script content received through that function, regardless of whether or not it originates from a third-party location. This issue was temporarily fixed in versions 1.12.0 through 1.12.2, but was reverted as it was considered a breaking change. All jQuery 2 versions exhibit this issue. jQuery 3 is the only branch that addresses this issue. Successful exploitation requires that the attacker either can supply the third-party location to the function, or that an existing third-party dependency goes rogue (i.e. is compromised).
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	A malicious actor may be able to stage attacks using known vulnerabilities in the installed version of jQuery. No updates or security patches will be provided by the vendor to mitigate these risks.
Recreation Flow	1. Visit the following URL: https://www.auksjonen-test.no/ 2. Open the browser's built in developer tools 3. Enter the following string into the JavaScript console: jQuery.fn.jquery 4. Note that the application includes an End-of-Life version of jQuery, which has known security issues associated with it
Solution	Upgrade to the latest secure version for jQuery 3.
Status	present
CWE	CWE-1035
OWASP Top 10 2013	A09: Using components with known vulnerabilities
OWASP Top 10 2017	A09: Using Components with Known Vulnerabilities
OWASP Top 10 2021	A06: Vulnerable and Outdated Components
Age	12 days - First seen: 2025-10-01 09:02, Last seen: 2025-10-01 09:08
Tags	, auksjonen
Attachments	jQuery 2.2.2.png See appendix 1.1

Sensitive Cookie "JSESSIONID" not set as Secure

Risk factor	⚠ Low This vulnerability can be exploited with advanced skills and network access to the system by an attacker who does not have access to credentials with some impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	2.6 - (AV:N/AC:H/Au:N/C:P/I:N/A:N)
CVSS v3	3.4 - (AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:N/A:N)
Description	Cookies which do not have the Secure flag set are transmittable over unencrypted HTTP connections. Such behavior can be desired in HTTP/HTTPS deployments where some cookies must be shared between both services. However, not setting the Secure flag for cookies containing sensitive data (such as access tokens) could allow an attacker to capture the cookies by downgrading the connection to HTTP, or by making the victim visit a URL pointing to the HTTP service. This cookie can hold sensitive data, and should therefore be better protected in order to mitigate the risk of exposure.
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	An attacker could be able to access the cookie if the connection is downgraded to HTTP.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no/ 2. Review the "JESESSIONID" cookie (e.g. through the HTTP response header), and note that the Secure attribute is not set: Set-Cookie: JSESSIONID=wZeWP32ygj8_tyXUEWgXfsat8H2Xync8ElaBAEy4; HttpOnly; Path=/api/
Solution	Set the Secure flag for the cookie.
Status	present
CWE	**CWE-614**
CAPEC	**614: CAPEC-614**
OWASP Top 10 2004	A03: Broken Authentication and Session management
OWASP Top 10 2007	A07: Broken Authentication and Session Management
OWASP Top 10 2010	A03: Broken Authentication and Session Management
OWASP Top 10 2013	**A02: Broken Authentication and Session Management**
OWASP Top 10 2017	**A02: Broken Authentication**
OWASP Top 10 2021	**A05: Security Misconfiguration**
Age	10 days - First seen: 2025-10-02 14:08, Last seen: 2025-10-02 14:14
Tags	, auksjonen
Attachments	Secure Not Set.png See appendix 1.1

Sensitive Cookie "JSESSIONID" not set as SameSite

Risk factor	⚠ Low This vulnerability can be exploited with advanced skills and network access to the system by an attacker who does not have access to credentials with no impact on confidentiality, some impact to the integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	2.6 - (AV:N/AC:H/Au:N/C:N/I:P/A:N)
CVSS v3	3.4 - (AV:N/AC:H/PR:N/UI:R/S:C/C:N/I:L/A:N)

Description	The application issues a sensitive cookie without a set "SameSite" attribute. This allows the cookie to be submitted in cross-site requests provided that the browser is not defaulting to a proper value. The values that the attribute can be given are "Strict", "Lax" and "None". The "Strict" value prevents the cookie to be submitted in any cross-site request. If the "Lax" value is given, the cookie can only be submitted in cross-site requests using the GET method provided it got issued by a top-level navigation. Lastly, a "None" value allows the cookie to be submitted in any cross-site request. Setting a proper value can hinder vulnerabilities such as cross-site request forgery (CSRF) and clickjacking provided that other controls are not in place. Keep in mind that solely relying on the "Lax" value as a mitigation for CSRF attacks, for instance, is not infallible; some applications may accept state-changing requests to be sent via GET, yielding this mitigation useless.
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	An attacker might be able to, for instance, carry out cross-site request forgery (CSRF) attacks if requests are not properly validated by other means.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no/ 2. Review the "JESESSIONID" cookie (e.g. through the HTTP response header), and note that the SameSite attribute is not set: Set-Cookie: JSESSIONID=wZeWP32ygj8_tyXUEWgXfsat8H2Xync8EIaBAEy4; HttpOnly; Path=/api/
Solution	Set the SameSite value of the cookie to either "Strict" or "Lax".
Status	present
CWE	**CWE-1275**
OWASP Top 10 2021	**A01: Broken Access Control**
Age	10 days - First seen: 2025-10-02 13:45, Last seen: 2025-10-02 14:08
Tags	, auksjonen
Attachments	SameSite Not Set.png See appendix 1.1

Missing Content Security Policy

Risk factor	▲ Recommendation This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with no impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	0.0 - (AV:N/AC:L/Au:N/C:N/I:N/A:N)
CVSS v3	0.0 - (AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N)
Description	CSP is a defense-in-depth mechanism that can help prevent client-side data/content injection attacks, such as cross-site scripting (XSS), by specifying the origins from which resources can be loaded from. CSP also offers assorted hardening options, like the rewrite of URLs with the HTTP scheme to HTTPS, and the hindering of an application from being displayed in a frame by untrusted origins. Implementation can be done via the http-equiv attribute in the HTML <meta> tag or the Content-Security-Policy response header. For the policy to be effective as a preventative control, it is important to steer clear of several misconfigurations. The most common being including the 'unsafe-inline' keyword in the script-src directive when the policy is not based on nonces, hashes, or the 'strict-dynamic' keyword, as it will allow any inline scripts to be executed. Another pitfall is to include standalone schemes such as data:, http:, or https: in a policy that does not have the 'strict-dynamic' keyword set. This will result in an overly permissive policy, allowing resources from any origins to be loaded as long as they are using these schemes. For more information and guidance, check out our blog post on CSP at: https://outpost24.com/blog/content-security-policy-guide
URL	https://www.auksjonen-test.no/
HTTP method	GET

Port	443
Impact	As the policy solely provides an additional layer of security, no direct impact is associated by not implementing it.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no/ 2. Note that the "Content-Security-Policy" response header is missing as well as how the CSP is not set directly in the markup
Solution	Consider implementing a properly configured CSP.
Status	present
Age	9 days - First seen: 2025-10-03 11:30, Last seen: 2025-10-03 11:41
Tags	, auksjonen

Unvalidated Redirection via HTTP Host Header

Risk factor	▲ Recommendation This vulnerability can be exploited with advanced skills and network access to the system by an attacker who does not have access to credentials with no impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	0.0 - (AV:N/AC:H/Au:N/C:N/I:N/A:N)
CVSS v3	0.0 - (AV:N/AC:H/PR:N/UI:R/S:C/C:N/I:N/A:N)
Description	Unvalidated redirection (or open redirect) occurs when a web application fails to validate a destination URL before forwarding users there. This is exploited by manipulating user input to send victims to malicious sites, create deceptive links, and carry out phishing attacks. In this case, the finding is classified as a recommendation, as the attacker would need to manipulate the target's host header. This requires highly specific circumstances, making such an attack non-trivial to perform.
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	An attacker could leverage the redirection functionality present here in order to send users to an unexpected location, such as a malicious web page.
Recreation Flow	1. Issue an HTTP request to the following URL and note the resulting redirect: http://www.auksjonen-test.no/ 2. Repeat the request above, but modify the value of the Host header from www.auksjonen-test.no to outpost24.com: Host: outpost24.com 3. Note how the application freely redirects without first validating the destination: Location: https://outpost24.com/
Solution	Do not rely on user-supplied input in order to determine the redirection destination. If this is not feasible, consider applying a domain- or resource-whitelist, and replace any non-conforming values with a default value.
Status	present
CWE	CWE-601
OWASP Top 10 2004	A01: Unvalidated Input
OWASP Top 10 2007	A10: Failure to Restrict URL Access
OWASP Top 10 2010	A10: Unvalidated Redirects and Forwards
OWASP Top 10 2013	A10: Unvalidated Redirects and Forwards
OWASP Top 10 2021	A01: Broken Access Control
Age	9 days - First seen: 2025-10-03 11:34, Last seen: 2025-10-03 11:41
Tags	, auksjonen

Missing "X-Content-Type-Options: nosniff" Response Header

Risk factor	▲ Recommendation This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with no impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	0.0 - (AV:N/AC:L/Au:N/C:N/I:N/A:N)
CVSS v3	0.0 - (AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N)
Description	Having the X-Content-Type-Options header set to nosniff instructs the browser to not interpret a given file's media type based on its contents and instead rely on the media type declared in the Content-Type header provided by the server. By not having this header set, the application might be prone to MIME-sniffing attacks, leading to malicious client-side code being executed in a victim's browser. This report entry aims to cover all responses as a simplified best practice even though the header might already be set in the responses prone to exploitation.
URL	https://www.auksjonen-test.no
HTTP method	GET
Port	443
Impact	An attacker could cause the web browser to interpret files as something else than declared by the content type.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no 2. Review the server HTTP response headers: HTTP/2 200 OK Date: Fri, 03 Oct 2025 09:28:29 GMT Content-Type: text/html Server: cloudflare Last-Modified: Mon, 29 Sep 2025 13:02:34 GMT Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800} Vary: Accept-Encoding Server-Timing: cfCacheStatus;desc="DYNAMIC" Server-Timing: cfEdge;dur=13,cfOrigin;dur=9 Cache-Control: no-cache Report-To: {"group":"cf- nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=Lc%2BAEQ P%2Frk52cpkuubxBM36kRT%2Ff8vlXrhU2ADVUg5qzmEYJL3JTs2cFOV916i57U94zTiP7vloBgjq BkW1dwpzTt4KHCwzDOrjrtCW3W8CT6kcq"}]} Alt-Svc: h3=":443"; ma=86400 Cf-Cache-Status: DYNAMIC Cf-Ray: 988b68e23b7cbd27-ARN 3. Note that the "X-Content-Type-Options: nosniff" header is not set
Solution	Consider setting the 'X-Content-Type-Options: nosniff' HTTP response header on all responses.
Status	present
Age	9 days - First seen: 2025-10-03 11:30, Last seen: 2025-10-03 11:41
Tags	, auksjonen

HTTP Strict Transport Security not Configured

Risk factor	▲ Recommendation This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with no impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	0.0 - (AV:N/AC:L/Au:N/C:N/I:N/A:N)
CVSS v3	0.0 - (AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N/E:F/RL:W/RC:C)

Description	The web application accepts connections over encrypted HTTPS, but it does not instruct the client user agent to enforce HTTPS for future connections. This could by extension lead to that a user either directly or inadvertently accesses the application over unencrypted HTTP - a downgrade which could subsequently be exploited by an adjacent attacker. For example, an attacker could trick the victim user agent to perform a request over HTTP, and then intercept all non-Secure client cookies. Implementing HSTS require careful consideration of the security/convenience implications, as it would also prevent users from legitimately downgrading their connections. For reference, please see: https://tools.ietf.org/html/rfc6797
URL	https://www.auksjonen-test.no/
HTTP method	GET
Port	443
Impact	Inadvertently connecting once over an insecure connection could allow attackers to perform Man-in-the-Middle-Attacks.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no/ 2. Inspect the server HTTP response headers, and note that the "Strict-Transport-Security" HSTS header is not present: HTTP/2 200 OK Date: Fri, 03 Oct 2025 09:28:29 GMT Content-Type: text/html Server: cloudflare Last-Modified: Mon, 29 Sep 2025 13:02:34 GMT Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800} Vary: Accept-Encoding Server-Timing: cfCacheStatus;desc="DYNAMIC" Server-Timing: cfEdge;dur=13,cfOrigin;dur=9 Cache-Control: no-cache Report-To: {"group":"cf-nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=Lc%2BAEQP%2Frk52cpkuubxBM36kRT%2Ff8vlXrhU2ADVUg5qzmEYJL3JTs2cFOV916i57U94zTiP7vloBgjqBkW1dwpzTt4KHCwzDOrjrtCW3W8CT6kcq"}]} Alt-Svc: h3=":443"; ma=86400 Cf-Cache-Status: DYNAMIC Cf-Ray: 988b68e23b7cbd27-ARN
Solution	Set the "Strict-Transport-Security" HTTP response header on all HTTPS responses.
Status	present
CWE	**CWE-311**
CAPEC	**384: CAPEC-384** **385: CAPEC-385** **609: CAPEC-609** **65: CAPEC-65** **386: CAPEC-386** **387: CAPEC-387** **388: CAPEC-388** **37: CAPEC-37** **204: CAPEC-204** **157: CAPEC-157** **477: CAPEC-477** **158: CAPEC-158** **31: CAPEC-31** **383: CAPEC-383**
OWASP Top 10 2004	A08: Insecure Storage
OWASP Top 10 2007	A08: Insecure Cryptographic Storage A09: Insecure Communications
OWASP Top 10 2010	A07: Insecure Cryptographic Storage A09: Insufficient Transport Layer Protection
OWASP Top 10 2013	**A02: Broken Authentication and Session Management** **A06: Sensitive Data Exposure**
OWASP Top 10 2017	**A02: Broken Authentication** **A03: Sensitive Data Exposure**
OWASP Top 10 2021	**A04: Insecure Design**
Age	9 days - First seen: 2025-10-03 11:29, Last seen: 2025-10-03 11:41

Tags	, auksjonen

Preload for HTTP Strict Transport Security not Configured

Risk factor	▲ Recommendation This vulnerability can be exploited with ease and network access to the system by an attacker who does not have access to credentials with no impact on confidentiality, no impact to integrity of information and without affecting the availability of the information or system. Successful attacks require human interaction from a person other than the attacker. There are currently no exploits in the public domain. However, attacks may be well described or privately held.
CVSS v2	0.0 - (AV:N/AC:L/Au:N/C:N/I:N/A:N)
CVSS v3	0.0 - (AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N)
Description	The application implements HSTS, but does not express the intention to be included in the preloading lists. This means that the application HSTS policy is not preloaded by the major browsers. HSTS is intended to prevent users from accessing the application if the integrity of the connection cannot be established. Despite this, a window of opportunity may still exist for an attacker in the case where the user-agent has not yet seen the policy. The HSTS preloading lists exist to supply web browsers with a list of domains which should always be considered to use HSTS. In order for the preload directive to work, the domain must be registered. Please see https://www.chromium.org/hsts for further information.
URL	https://www.auksjonen-test.no/favicons/favicon-32x32.png
HTTP method	GET
Port	443
Impact	Inadvertently connecting once over an insecure connection could allow attackers to perform so called Man-in-the-Middle attacks.
Recreation Flow	1. Navigate to the following URL: https://www.auksjonen-test.no/favicons/favicon-32x32.png 2. Inspect the server HTTP response headers, and note that the "Strict-Transport-Security" HSTS header is specified: Strict-Transport-Security: max-age=5184000 3. Note that the "preload" directive is not included
Solution	Set the preload directive in the response header and register the domain in the Chromium preload list. Please see https://www.chromium.org/hsts for further information.
Status	present
Age	9 days - First seen: 2025-10-03 11:33, Last seen: 2025-10-03 11:41
Tags	, auksjonen

Methodology

Approach

The manual components of the testing in based on the methodology set forth in the OWASP Testing Guide to the extent where so is applicable, as well as takes influence in risk management guidelines as ISO27005, testing guidelines as published by the PCI SSC in regards to testing of PCI DSS environments, the OSSTMM, and NIST publications on penetration testing.

The scope is established, application content, behavior and context mapped, interactions identified and then subjected to testing. Risks are noted and explored further.
All identified risks are scored according to business best practice using the Common Vulnerability Scoring System. When known vulnerabilities are discovered, their reference and CVE-id are included in the reporting.

Each risk identified by an analyst or the supporting technical platforms and scanners is manually verified, recreated and documented, including explanations of its impact and a recommendations in regards to problem resolution.

Scope and Vectors

The scope is defined as the targets of the tests. The locations of the tester and the routes from the tester to the scope is defined as the vector. The scope was supplied by the customer and contains the application(s) to be tested.

Asset Group

Auksjonen **(Web application)**

The vector is shown below. The list was supplied by Outpost24 AB and contains the remote IP addresses which were used during the test:

Vectors	IP Addresses
Public Internet	

Activities

The following activities were executed during the testing:

Network and Application Scans	Determining host, IP, route and location information of the network systems related to the applications to be tested, as well as initial configuration of the continuous monitoring.
Component and Service Identification	After the manners to communicate with the server are inventoried, it will be attempted to identify which services, operating systems, patches and components are running on the targets to be tested. Further it will be inspected if security devices or redundant systems can be detected.
Vulnerability Research and Configuration Inspection	The detected components will be checked against known vulnerabilities, problems, and configuration issues.
Application Exploiting	Testing will attempt to tamper/exploit the detected weaknesses. If such tests are expected to endanger the confidentiality, integrity, or availability of data, then such testing may only occur in consultation with the customer.
Analysis and Reporting	The vulnerability information, evidence, and other data collected are analyzed and compiled into the vulnerability report.

Explicit Exceptions

The following tests were not executed during the testing:

Denial of Service Attacks	The result of a denial of service attack might cause the application to cease normal behavior. Therefore, attacks of this type will not be executed, unless explicitly requested by the customer.
Social Engineering	In social engineering, an adversary attempts to gain access or otherwise manipulate an application by attacking the people and employees with privileged access, e.g. by enticing them to divulge information.
Physical Security	Physical security is the protection of personnel, hardware, programs, networks, and data from physical circumstances and events that could cause serious loss or damage to an enterprise, agency, or institution.

Purpose

Outpost24 AB AB has executed a security assessment on the in-scope assets, comprising the following test type(s), applied as appropriate: Web application.

The objective of these tests was to get an impression of the information security of the in-scope assets and their supporting environment. Based on the test results, Outpost24 AB AB will compile a vulnerability report and give recommendations for improvements where applicable.

The end result will be that the customer will gain insight into the robustness and security of their in-scope assets. Conclusions will be provided with clear suggestions for operational solutions and managerial focus, leading to heightened security.

OWASP Top 10 2021 Description

For web applications, mobile applications and APIs observations are mapped onto the applicable OWASP standard. The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

A01	Broken Access Control	Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.
A02	Cryptographic Failures	Poor protection of data in transit and/or at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS).
A03	Injection	Injection flaws occur when an attacker can manipulate user input to inject malicious code into an application as part of a command or a query and execute it. This can lead to data loss, corruption, or unauthorized access to sensitive data.
A04	Insecure Design	Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design". An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.
A05	Security Misconfiguration	Security misconfigurations occur when security features are not configured properly. This includes presence of improperly configured permissions, default accounts and their passwords, stack traces or overly verbose error messages as well as poor hardening of the used frameworks and libraries leaving systems and applications vulnerable to attacks.
A06	Vulnerable and Outdated Components	Vulnerabilities introduced by the use of third-party or open source components with known security issues. These components may contain unpatched vulnerabilities, which can be exploited by attackers to gain unauthorized access, steal data, or execute malicious code.
A07	Identification and Authentication Failures	Vulnerabilities related to the verification of user identity and access control. This can include weak password policies, lack of multi-factor authentication, insufficient user validation, and improper session management. These vulnerabilities can be exploited by attackers to bypass authentication mechanisms, impersonate legitimate users, gain unauthorized access to sensitive data or functionalities, and conduct other malicious activities.
A08	Software and Data Integrity Failures	Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise.
A09	Security Logging and Monitoring Failures	Lack of proper logging and monitoring of security events. This can include issues such as missing or incomplete logs, insufficient monitoring, and inadequate incident response procedures. These vulnerabilities can be exploited by attackers to evade detection and remain undetected on the affected system for an extended period, leading to further compromises and data theft.
A10	Server-Side Request Forgery (SSRF)	SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

Common Vulnerability Scoring System (CVSS) v3.1 Description

The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat. Scores are calculated based on a formula that depends on several metrics that approximate ease of exploit and the impact of exploit. Scores range from 0 to 10, with 10 being the most severe. While many utilize only the CVSS Base score for determining severity, Temporal and Environmental scores also exist, to factor in availability of mitigation and how widespread vulnerable systems are within an organization, respectively.

Severity	Definition
Critical	The attack scenario tested in this exercise succeeded and resulted in a systems compromise. Exploitation is trivial. Exploitation requires no authentication, or authentication is available to a member of the public with minimal effort. Exploitation results in a large-scale loss of sensitive information or tangible assets. A strong need for immediate corrective measures exists.
High	The attack scenario tested in this exercise succeeded and resulted in a systems compromise. The attack can only be performed by an authenticated user. Technical vulnerability details and/or exploit code are publicly available. An additional attack vector may be needed to craft a successful attack using this exploit, but that vector is trivial. Exploitation of the vulnerability may result in the costly loss of sensitive data or tangible assets, or may significantly violate, harm, or impede the organization's mission, reputation, or interest. A strong need for corrective measures exists.
Medium	Exploitation requires a skilled attacker. Exploitation does not result in elevated privileges. Controls are in place that may impede successful exploitation of the vulnerability. To craft a successful attack using this exploit/vulnerability an additional vector is needed (such as phishing or social engineering); this additional attack vector is not trivial. Exploitation of the vulnerability may violate, harm, or impede the organization's mission, reputation, or interest.
Low	Exploitation is extremely difficult. Controls are in place to prevent, or at least significantly impede, the vulnerability from being exploited. The attack scenario under which this vulnerability can be exploited is possible, but extremely unlikely. The application owners must determine whether corrective actions are required or decide to accept the risk.
None / Informational	Information disclosed may aid an attacker in system reconnaissance. The information disclosed will be useful to an attacker when exploiting higher risk issues. Information is disclosed that implies poor management practices of the application's infrastructure. The described observation arouses suspicions and requires closer examination but does not pose a direct threat.

Test case appendix - SWAT

SWAT is a hybrid service delivery covering automated monitoring and web application scanning as well as at least quarterly penetration testing, including application logics, of web applications under service.

The test-cases are oriented around the OWASP TESTING GUIDE, and for the application the following controls has been performed. Note that a control will be marked as audited either if found present and audited, or were found not present and hence not auditable - This to show that the application has been audited for this class of risks.

Test Activities and Descriptions	OWASP testing guide	Audit note
4.1 Information Gathering		
4.1.1 Conduct Search Engine Discovery and Reconnaissance for Information Leakage (WSTG-INFO-01)	WSTG-INFO-01	Not in scope
Use a search engine to search for potentially sensitive information. This may include:		Not in scope
Network diagrams and configurations		Not in scope
Archived posts and emails by administrators and other key staff		Not in scope
Log on procedures and username formats		Not in scope
Usernames and passwords		Not in scope
Third-party or cloud service configuration files		Not in scope
Revealing error message content		Not in scope
Development, test, UAT and staging versions of the website		Not in scope
4.1.2 Fingerprint Web Server (WSTG-INFO-02)	WSTG-INFO-02	Audited
Determine web server software and (if possible) version		Audited
HTTP response headers, sending malformed requests, etc.		Audited
4.1.3 Review Webserver Metafiles for Information Leakage (WSTG-INFO-03)	WSTG-INFO-03	Audited
Identify hidden or obfuscated paths and functionality through the analysis of metadata files		Audited
Locate and review robots.txt and security.txt		Audited
4.1.4 Enumerate Applications on Webserver (WSTG-INFO-04)	WSTG-INFO-04	Audited
Enumerate the applications within scope that exist on a web server		Audited
Validate that the scope is correct and no additional applications are "hidden" within scope		Audited
4.1.5 Review Webpage Comments and Metadata for Information Leakage (WSTG-INFO-05)	WSTG-INFO-05	Audited
Review webpage comments and metadata to find any information leakage		Audited
Identify JavaScript code, gather JavaScript files and source map files		Audited
4.1.6 Identify application entry points (WSTG-INFO-06)	WSTG-INFO-06	Audited
Identify possible entry and injection points through request and response analysis:		Audited
- Query (GET) parameters		Audited
- Body parameters		Audited
- Cookies		Audited
- Request headers		Audited
- REST-style parameters		Audited
- Where are cookies set?		Audited
- Does the application fail during normal operation (i.e. HTTP 500, 404)		Audited

- Load balancers in place (might mean that exploits have to be repeated until vulnerable back-end server is hit)		Not in scope
4.1.7 Map execution paths through application (WSTG-INFO-07)	WSTG-INFO-07	Audited
Map the application structure and paths		Audited
Note what parts of the application might share server-side components and code		Audited
Note which parts might contain unique functionality		Audited
Note which functionality might not be exposed		Audited
4.1.8 Fingerprint Web Application Framework (WSTG-INFO-08)	WSTG-INFO-08	Audited
Fingerprint the components being used by the web application		Audited
HTTP headers, cookies, HTML source code, specific files and folders, file extensions, error messages		Audited
4.1.9 Fingerprint Web Application (WSTG-INFO-09)	WSTG-INFO-09	Audited
Merged into 4.1.8 Fingerprint Web Application Framework (WSTG-INFO-08)		Not in scope
4.1.10 Map Application Architecture (WSTG-INFO-10)	WSTG-INFO-10	Not in scope
Generate a map of the application at hand based on the research conducted		Not in scope
4.2 Configuration and Deployment Management Testing		
4.2.1 Test Network/Infrastructure Configuration (WSTG-CONF-01)	WSTG-CONF-01	Audited
Determine if the in-use components and versions has any known vulnerabilities and/or are considered End of Life		Audited
Validate that used frameworks and systems are secure by auditing if they are susceptible to known vulnerabilities due to unmaintained software or default settings (such as default credentials)		Audited
4.2.2 Test Application Platform Configuration (WSTG-CONF-02)	WSTG-CONF-02	Audited
Ensure that defaults and known files have been removed		Audited
Validate that no debugging code or extensions are left in the production environments		Audited
Review the logging mechanisms set in place for the application		Not in scope
4.2.3 Test File Extensions Handling for Sensitive Information (WSTG-CONF-03)	WSTG-CONF-03	Audited
- Assert how the web server presents files according to their file extension		Audited
- Determine if any server-side code can be discovered by forceful browsing		Audited
- Determine if file upload or file access restrictions based on file extensions can be circumvented		Audited
4.2.4 Review Old, Backup and Unreferenced Files for Sensitive Information (WSTG-CONF-04)	WSTG-CONF-04	Audited
Attempt to reveal unreferenced files through:		Audited
- Forceful browsing, "blind guessing"		Audited
- Server misconfigurations or vulnerabilities (such as enabled directory listing, or IIS short name)		Audited
- Search engines and public information		Not in scope
- HTML (or other) source comments		Audited
- Inference from the Naming Scheme Used for Published Content		Audited
4.2.5 Enumerate Infrastructure and Application Admin Interfaces (WSTG-CONF-05)	WSTG-CONF-05	Audited
Identify hidden administrator interfaces and functionality		Audited
Validate that default credentials are not in use		Audited
4.2.6 Test HTTP Methods (WSTG-CONF-06)	WSTG-CONF-06	Audited

- Determine which HTTP methods are supported, and to what extent		Audited
- Determine if TRACE is enabled (XST)		Audited
- Determine if regular (such as HEAD) or arbitrary (such as ASDF) methods can be used in order to bypass authorisation or cause other issues		Audited
4.2.7 Test HTTP Strict Transport Security (WSTG-CONF-07)	WSTG-CONF-07	Audited
- Determine if HSTS is properly configured for the application		Audited
- Determine whether or not HSTS preloading is properly configured		Audited
4.2.8 Test RIA cross domain policy (WSTG-CONF-08)	WSTG-CONF-08	Audited
- Determine if crossdomain.xml and clientaccesspolicy.xml exists, and if so, if they are properly set up		Audited
4.2.9 Test File Permission (WSTG-CONF-09)	WSTG-CONF-09	Not in scope
- Review and identify any rogue file permissions (on the web-server)		Not in scope
4.2.10 Test for Subdomain Takeover (WSTG-CONF-10)	WSTG-CONF-10	Not in scope
- Enumerate all possible domains (previous and current)		Not in scope
- Identify forgotten or misconfigured domains		Not in scope
4.2.11 Test Cloud Storage (WSTG-CONF-11)	WSTG-CONF-11	Audited
Assess that the access control configuration for the storage services is properly in place		Audited
Determine if it is possible to read unauthorized data		Audited
Determine if possible to upload new arbitrary file		Audited
4.3 Identity Management Testing		
4.3.1 Test Role Definitions (WSTG-IDNT-01)	WSTG-IDNT-01	Audited
- Identify and document roles used by the application		Audited
- Verify that user roles can not exceed their intended permissions		Audited
4.3.2 Test User Registration Process (WSTG-IDNT-02)	WSTG-IDNT-02	Audited
- Verify that the registration requirements are properly implemented and can not be circumvented or altered		Audited
- Verify that the registration process aligns with the business requirements		Audited
4.3.3 Test Account Provisioning Process (WSTG-IDNT-03)	WSTG-IDNT-03	Audited
Verify which accounts may provision other accounts and of what type		Audited
Is there any verification, vetting and authorization of provisioning requests?		Audited
Is there any verification, vetting and authorization of de-provisioning requests?		Audited
Can an administrator provision other administrators or just users?		Audited
Can an administrator or other user provision accounts with privileges greater than their own?		Audited
Can an administrator or user de-provision themselves?		Not in scope
How are the files or resources owned by the de-provisioned user managed? Are they deleted? Is access transferred?		Not in scope
4.3.4 Testing for Account Enumeration and Guessable User Account (WSTG-IDNT-04)	WSTG-IDNT-04	Audited
Review processes that pertain to user identification (e.g. registration, login, etc.)		Audited
Enumerate users where possible through response analysis		Audited
4.3.5 Testing for Weak or unenforced username policy (WSTG-IDNT-05)	WSTG-IDNT-05	Audited
- Determine whether or not there is a naming scheme in place for usernames		Audited
- Evaluate application response in regards to usernames following or breaking the scheme		Audited

4.4 Authentication Testing

4.4.1 Testing for Credentials Transported over an Encrypted Channel (WSTG-ATHN-01)	WSTG-ATHN-01	Audited
- Assert whether or not all credentials are transmitted over an encrypted channel		Audited
- Test if credentials are accepted over plaintext connections		Audited
4.4.2 Testing for default credentials (WSTG-ATHN-02)	WSTG-ATHN-02	Audited
- Determine if access can be achieved using standard credentials		Audited
- Determine if a common or guessable set of credentials are in use		Audited
- Determine if a default or guessable password is set for new accounts		Audited
4.4.3 Testing for Weak lock out mechanism (WSTG-ATHN-03)	WSTG-ATHN-03	Audited
- Determine if password brute forcing is possible (lacking automation protection)		Audited
- Determine if there is an account lockout in place, and the boundaries associated with it		Audited
- Determine if the lockout can be circumvented		Audited
4.4.4 Testing for bypassing authentication schema (WSTG-ATHN-04)	WSTG-ATHN-04	Audited
Ensure that authentication is applied across all services that require it. Attempt bypass using:		Audited
- Direct page request (forced browsing)		Audited
- Parameter or cookie modification		Audited
- Session token prediction		Audited
- SQL injections		Audited
4.4.5 Test remember password functionality (WSTG-ATHN-05)	WSTG-ATHN-05	Audited
Assert whether or not the application has a "remember me"-function. If so:		Audited
- Determine how the feature is implemented and how it functions		Audited
- Determine if any sensitive data is stored client-side (perhaps in a cookie)		Audited
Verify that credentials are only sent when authenticating, not for each request		Audited
4.4.6 Testing for Browser cache weakness (WSTG-ATHN-06)	WSTG-ATHN-06	Audited
- Determine if user agents are allowed to store sensitive documents in the history storage		Audited
- Determine if user agents are allowed to cache sensitive documents		Audited
4.4.7 Testing for Weak password policy (WSTG-ATHN-07)	WSTG-ATHN-07	Audited
What characters are permitted and forbidden for use within a password? Is the user required to use characters from different character sets such as lower and uppercase letters, digits and special symbols?		Audited
How often can a user change their password? How quickly can a user change their password after a previous change? Users may bypass password history requirements by changing their password 5 times in a row so that after the last password change they have configured their initial password again.		Audited
When must a user change their password?		Audited
Both NIST and NCSC recommend against forcing regular password expiry, although it may be required by standards such as PCI DSS.		Audited
How often can a user reuse a password? Does the application maintain a history of the user's previous used 8 passwords?		Audited
How different must the next password be from the last password?		Audited
Is the user prevented from using his username or other account information (such as first or last name) in the password?		Audited
What are the minimum and maximum password lengths that can be set, and are they appropriate for the sensitivity of the account and application?		Audited
Is it possible set common passwords such as Password1 or 123456?		Audited

4.4.8 Testing for Weak security question/answer (WSTG-ATHN-08)	WSTG-ATHN-08	Audited
Check whether or not answers to pre-generated security questions:		Audited
- Can be known by family members or friends (e.g. date of birth)		Audited
- Can easily be guessed (e.g. favourite colour)		Audited
- Can be publicly discovered (e.g. favourite movie, listed on Facebook)		Audited
Check whether or not self-generated questions can be weak ("What is 1 + 1?")		Audited
Check whether or not secret question answer can be found by brute force		Audited
4.4.9 Testing for weak password change or reset functionalities (WSTG-ATHN-09)	WSTG-ATHN-09	Audited
- Determine if one user can change the password of another user (unless this is expected, e.g. administrator)		Audited
- Determine if existing password reset functionality can be leveraged to change the password of other user accounts		Audited
- Determine if the password reset functionality has any flaws, e.g. guessable tokens		Audited
- Determine whether or not the password change or reset functions can be attacked via CSRF or similar vectors		Audited
4.4.10 Testing for Weaker authentication in alternative channel (WSTG-ATHN-10)	WSTG-ATHN-10	Audited
- Identify and understand the primary authentication method and channel		Audited
- Identify other authentication channels and map their scope		Audited
- Determine if the alternative channels undermine the primary channel		Audited
4.5 Authorization Testing		
4.5.1 Testing Directory traversal/file include (WSTG-ATHZ-01)	WSTG-ATHZ-01	Audited
- Identify injection points that pertain to path traversal		Audited
- For these entry points, determine whether or not directory traversal or file inclusion can occur		Audited
4.5.2 Testing for bypassing authorization schema (WSTG-ATHZ-02)	WSTG-ATHZ-02	Audited
Assess if horizontal or vertical access is possible, by determine if:		Audited
- Is it possible to access that resource even if the user is not authenticated?		Audited
- Is it possible to access that resource after the log-out?		Audited
- Is it possible to access functions and resources that should be accessible to a user that holds a different role or privilege?		Audited
- Is it possible to access the administrative functionality, using the same checks		Audited
4.5.3 Testing for Privilege Escalation (WSTG-ATHZ-03)	WSTG-ATHZ-03	Audited
- For all functionality associated with sessions, or specifically assigned privileges, determine whether or not it is possible to access or modify it using an unauthorised account		Audited
- Determine if the authorisation flaw can be used to escalate privileges		Audited
4.5.4 Testing for Insecure Direct Object References (WSTG-ATHZ-04)	WSTG-ATHZ-04	Audited
- Enumerate all object references exposed throughout the application		Audited
- Determine if these references can be altered to access data not intended for the current user		Audited
4.6 Session Management Testing		
4.6.1 Testing for Bypassing Session Management Schema (WSTG-SESS-01)	WSTG-SESS-01	Audited
Enumerate all cookies set by the application, and determine:		Audited
- Which cookies could have value to an attacker?		Audited

- Whether or not cookies are (or can be) sent over an unencrypted channel		Audited
- What HTTP/1.1 and HTTP/1.0 Cache-Control settings are used to protect cookies		Audited
- If sensitive data is exposed through the cookie		Audited
- If there is any obfuscation in place of the cookie name or value.		Audited
- Are the Session IDs provably random in nature?.		Audited
- Are the Session IDs provably resistant to statistical or cryptanalysis?		Not in scope
- Is the cookie tamper resistant? Will the application reject modified cookies?		Audited
- Does the cookie expire within a sane time period?		Audited
4.6.2 Testing for Cookies attributes (WSTG-SESS-02)	WSTG-SESS-02	Audited
Ensure that the proper security configuration is set for cookies containing sensitive data		Audited
Secure, HttpOnly, Domain, SameSite and Path		Audited
4.6.3 Testing for Session Fixation (WSTG-SESS-03)	WSTG-SESS-03	Audited
- Determine whether or not a fresh session token (cookie) is set upon successful authentication.		Audited
4.6.4 Testing for Exposed Session Variables (WSTG-SESS-04)	WSTG-SESS-04	Audited
Assert whether or not the session tokens are always transmitted securely (HTTPS, POST parameters, etc..)		Audited
Determine if the caching directives provide sufficient protection		Audited
4.6.5 Testing for Cross Site Request Forgery (CSRF) (WSTG-SESS-05)	WSTG-SESS-05	Audited
- For each unique request or function call, establish whether or not it can be triggered via CSRF, and whether or not that has any impact		Audited
4.6.6 Testing for logout functionality (WSTG-SESS-06)	WSTG-SESS-06	Audited
Verify the appearance and visibility of the log out functionality in the user interface		Audited
Verify that the logout function properly terminates the session client-side		Audited
Verify that the logout function properly terminates the session server-side		Audited
4.6.7 Test Session Timeout (WSTG-SESS-07)	WSTG-SESS-07	Audited
- Validate that a hard session timeout exists		Audited
- Assert if the session is invalidated by the client, by the server, or both		Audited
4.6.8 Testing for Session puzzling (WSTG-SESS-08)	WSTG-SESS-08	Audited
- Audit the gathered session variables and assert if it is possible to gain or escalate privileges by leveraging ("puzzling" together) a partial session		Audited
4.6.9 Testing for Session hijacking (WSTG-SESS-09)	WSTG-SESS-09	Audited
Determine if any sessions cookies are transmitted over unencrypted communication (if not already done in previous sections)		Audited
If HSTS not properly configured, validate that cookies are protected by secure flag and/or ensure that the domain attribute is not set		Not in scope
4.7 Input Validation Testing		
4.7.1 Testing for Reflected Cross Site Scripting (WSTG-INPV-01)	WSTG-INPV-01	Audited
- Identify variables that are reflected in responses, assess the input they accept and the encoding that gets applied on return (if any)		Audited
4.7.2 Testing for Stored Cross Site Scripting (WSTG-INPV-02)	WSTG-INPV-02	Audited
- Identify stored input that is reflected on the client-side, assess the input they accept and the encoding that gets applied on return (if any)		Audited

4.7.3 Testing for HTTP Verb Tampering (WSTG-INPV-03)	WSTG-INPV-03	Audited
Merged into 4.2.6 Test HTTP Methods (WSTG-CONF-06)		Not in scope
4.7.4 Testing for HTTP Parameter pollution (WSTG-INPV-04)	WSTG-INPV-04	Audited
- Determine if setting two parameters with identical name has any impact on the server response or filter validation		Audited
4.7.5 Testing for SQL Injection (WSTG-INPV-05)	WSTG-INPV-05	Audited
Identify and assess parameters which have potential database interaction for injection issues		Audited
If possible; determine which database is in use and go to relevant section below:		Audited
- Testing for Oracle		Audited
- Testing for MySQL		Audited
- Testing for SQL Server		Audited
- Testing for PostgreSQL		Audited
- Testing for MS Access		Audited
- Testing for NoSQL Injection		Audited
- Testing for ORM Injection		Audited
- Testing for Client-side		Not in scope
4.7.6 Testing for LDAP Injection (WSTG-INPV-06)	WSTG-INPV-06	Audited
Identify and assess parameters which have potential LDAP interaction for injection issues		Audited
4.7.7 Testing for XML Injection (WSTG-INPV-07)	WSTG-INPV-07	Audited
Identify and assess parameters which might be handled by XML parsers for injection issues		Audited
4.7.8 Testing for SSI Injection (WSTG-INPV-08)	WSTG-INPV-08	Audited
Identify and assess parameters which have the potential to echo user input into context that would allow for SSI injection		Audited
4.7.9 Testing for XPath Injection (WSTG-INPV-09)	WSTG-INPV-09	Audited
Identify and assess parameters which might be part of an XPath expression for injection issues		Audited
4.7.10 Testing for IMAP/SMTP Injection (WSTG-INPV-10)	WSTG-INPV-10	Audited
Identify and assess parameters that may (directly or indirectly) be used as parameters related to email handling		Audited
4.7.11 Testing for Code Injection (WSTG-INPV-11)	WSTG-INPV-12	Audited
Identify and assess paramters which could potentially be evaluated as code		Audited
- Testing for Local File Inclusion		Audited
- Testing for Remote File Inclusion		Audited
4.7.12 Testing for Command Injection (WSTG-INPV-12)	WSTG-INPV-12	Audited
Identify and assess parameters which could potentially be evaluated as OS commands		Audited
4.7.13 Testing for Format String Injection (WSTG-INPV-13)	WSTG-INPV-13	Audited
Assess whether injecting format string conversion specifiers into user-controlled fields causes undesired behaviour from the application		Audited
4.7.14 Testing for Incubated Vulnerability (WSTG-INPV-14)	WSTG-INPV-14	Audited
Identify and assess injections that are stored and may be leveraged in order to stage new attacks (such as Stored XSS, HTML, etc.)		Audited

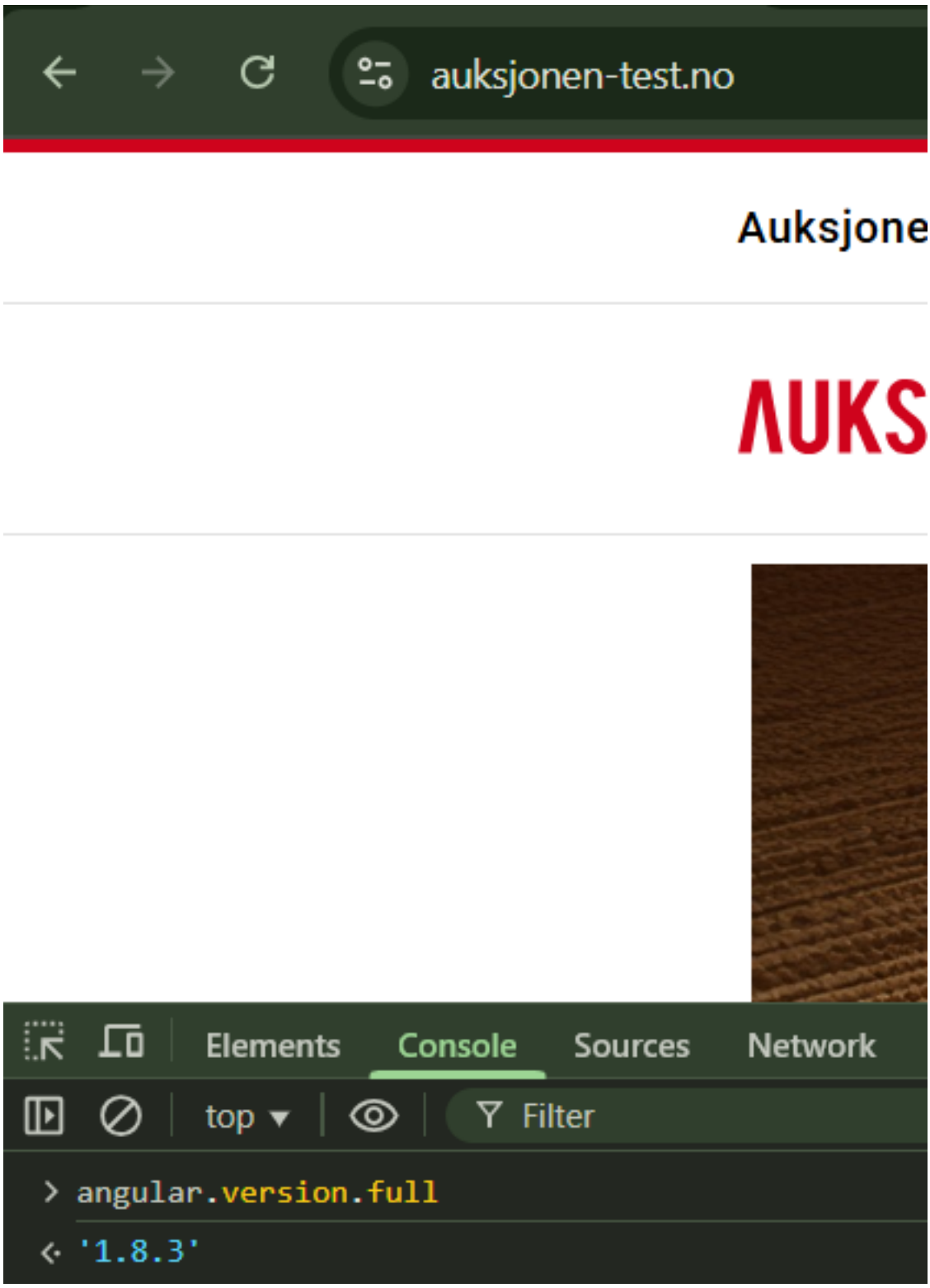
4.7.15 Testing for HTTP Splitting/Smuggling (WSTG-INPV-15)	WSTG-INPV-15	Audited
- Assert if HTTP request splitting is possible by leveraging echoed values present in the HTTP response header section		Audited
- Assert if HTTP request smuggling is possible in the target environment		Audited
4.7.16 Testing for HTTP Incoming Requests (WSTG-INPV-16)	WSTG-INPV-16	Not in scope
Monitor all incoming and outgoing HTTP requests to the Web Server to inspect any suspicious requests		Not in scope
Verify that there are no unnecessary or suspicious HTTP request sending in the background		Not in scope
4.7.17 Testing for Host Header Injection (WSTG-INPV-17)	WSTG-INPV-17	Audited
- Assess if the Host header is being parsed dynamically in the application		Audited
- If yes; assess if it can be abused for redirection or to bypass any other security controls that trely on the header		Audited
4.7.18 Testing for Server-side Template Injection (WSTG-INPV-18)	WSTG-INPV-18	Audited
Identify and assess parameters that might be evaluated by server-side template engine		Audited
If possible, identify template engine and potential impact		Audited
4.7.19 Testing for Server-Side Request Forgery (WSTG-INPV-19)	WSTG-INPV-19	Audited
Identify and assess parameters that might be parsed by the underlying components to make additional requests		Audited
Assess potential PDF generators (rendering HTML server-side)		Audited
4.8 Testing for Error Handling		
4.8.1 Testing for Improper Error Handling (WSTG-ERRH-01)	WSTG-ERRH-01	Audited
Review all error messages generated by activities		Audited
Determine how the application responds to:		Audited
- Resource not found or forbidden		Audited
- Malformed HTTP requests		Audited
- Accessing application in an unexpected way (bypassing regular flow)		Audited
- Invalid input types		Audited
4.8.2 Testing for Stack Traces (WSTG-ERRH-02)	WSTG-ERRH-02	Audited
Merged into 4.8.1 Testing for Improper Error Handling (WSTG-ERRH-01)		Not in scope
4.9 Testing for weak Cryptography		
4.9.1 Testing for Weak Transport Layer Security (WSTG-CRYP-01)	WSTG-CRYP-01	Audited
Assess the service configuration; protocols, ciphers and TLS extensions		Audited
Determine if any weak SSL/TLS ciphers are in use, and if any weak protocols are in use		Audited
Determine if the certificate is signed by a recognized CA		Audited
Determine if the certificate is valid and has a proper signing key set		Audited
4.9.2 Testing for Padding Oracle (WSTG-CRYP-02)	WSTG-CRYP-02	Not in scope
Identify encrypted messages that rely on padding and attempt to break the padding of the encrypted messages to analyse the returned error messages for further analysis		Not in scope
4.9.3 Testing for Sensitive Information sent via Unencrypted Channels (WSTG-CRYP-03)	WSTG-CRYP-03	Audited
Determine if any sensitive (Personal Identifying Information) data is transmitted in clear text (partially already covered in previous sections)		Audited

4.9.4 Testing for Weak Encryption (WSTG-CRYP-04)	WSTG-CRYP-04	Not in scope
Provide a guideline for the identification weak encryption or hashing uses and implementations (source code review)		Not in scope
4.10 Business Logic Testing		
4.10.1 Test Business Logic Data Validation (WSTG-BUSL-01)	WSTG-BUSL-01	Audited
Determine how the application front-end and back-end validates data, and note any discrepancies		Audited
Attempt to break the format of the expected data and analyze how the application is handling it		Audited
4.10.2 Test Ability to Forge Requests (WSTG-BUSL-02)	WSTG-BUSL-02	Audited
Review application requests looking for guessable, predictable, or hidden functionality of fields		Audited
Forge HTTP request to leverage these parameters and functions, and determine if any impact can be established		Audited
4.10.3 Test Integrity Checks (WSTG-BUSL-03)	WSTG-BUSL-03	Audited
Identify controls that dynamically generate output based on some criteria, and determine how the functionality or parameters presented differs		Audited
- For each different parameter or function, determine the impact of unexpected or unauthorised input or access		Audited
Attempt to insert, update, or delete data values used by each component that should not be allowed per the business logic workflow		Audited
4.10.4 Test for Process Timing (WSTG-BUSL-04)	WSTG-BUSL-04	Audited
Identify which processes are dependent on time and if there is a meaningful difference in response time between various inputs, function calls, or results		Audited
4.10.5 Test Number of Times a Function Can be Used Limits (WSTG-BUSL-05)	WSTG-BUSL-05	Audited
- Assess functions that has an implied limit and determine if it is possible to circumvent the limit		Audited
- Assess functions that has no limit and determine if the lack of such restriction can result in some form of impact		Audited
4.10.6 Testing for the Circumvention of Work Flows (WSTG-BUSL-06)	WSTG-BUSL-06	Audited
- Identify work flows and procedures within the application and determine if it is possible to navigate non linearly or skip steps		Audited
4.10.7 Test Defenses Against Application Mis-use (WSTG-BUSL-07)	WSTG-BUSL-07	Audited
Review results from previous sections and determine how the application handles abuse of intended functionality		Audited
Understand the defences in place and verify if they are enough to protect the system against bypassing techniques		Audited
4.10.8 Test Upload of Unexpected File Types (WSTG-BUSL-08)	WSTG-BUSL-08	Audited
- For each file upload feature, determine whether or not only intended file types can be uploaded (both for file name, and actual file type)		Audited
4.10.9 Test Upload of Malicious Files (WSTG-BUSL-09)	WSTG-BUSL-09	Audited
Determine which file types should be considered malicious within the context of the application		Audited
Determine how the uploaded files are processed and stored. If applicable, attempt to upload the EICAR anti-malware test file		Audited
4.11 Client Side Testing		
4.11.1 Testing for DOM based Cross Site Scripting (WSTG-CLNT-01)	WSTG-CLNT-01	Audited

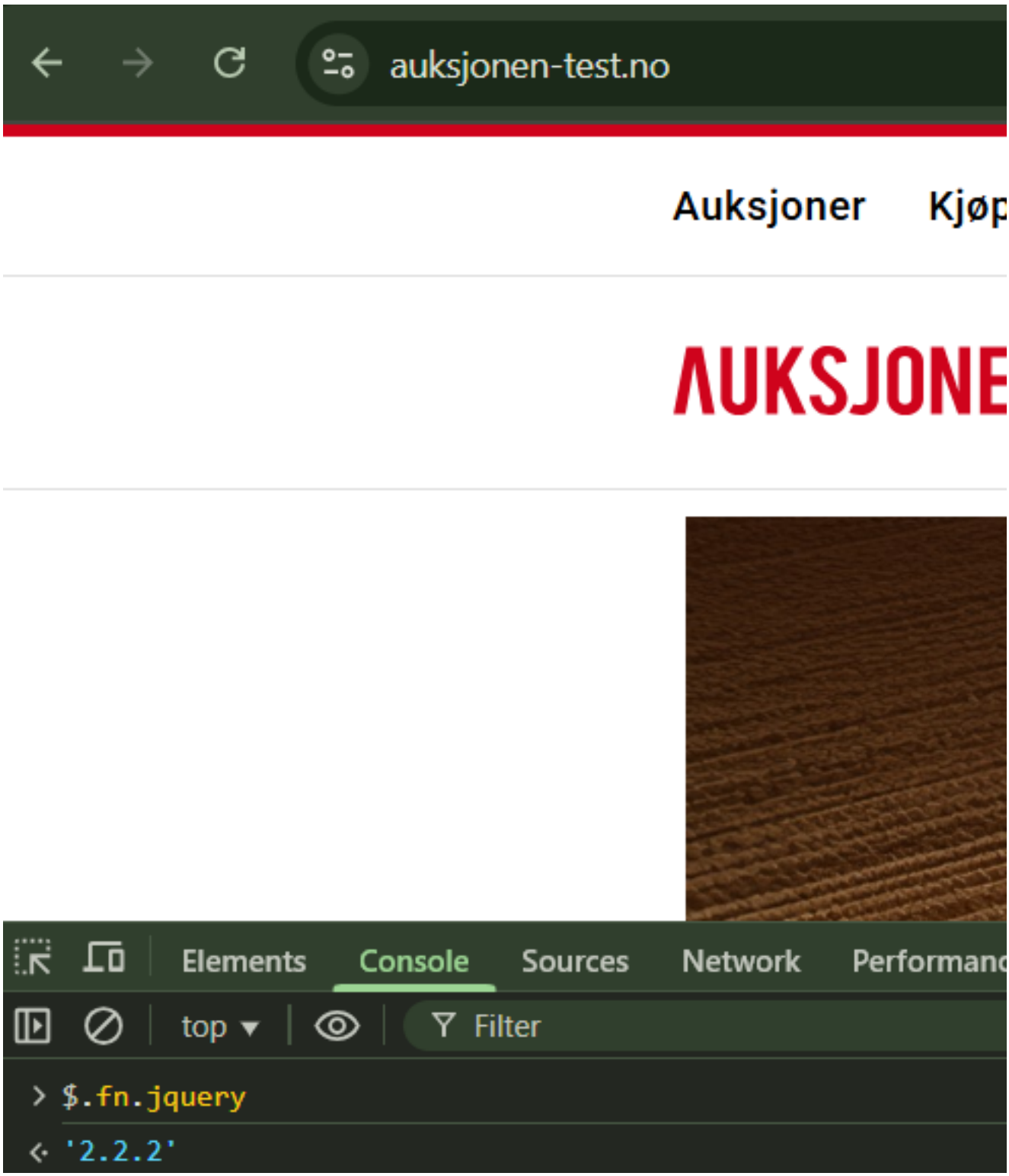
Review custom JavaScript code (that are not from a known library) and enumerate objects that supply or are used as input to JavaScript functions to determine if it is possible to cause attacker-supplied code to be evaluated		Audited
4.11.2 Testing for JavaScript Execution (WSTG-CLNT-02)	WSTG-CLNT-02	Audited
Review custom JavaScript code (that are not from a known library) and enumerate objects that supply or are used as input to JavaScript functions to determine if it is possible to cause attacker-supplied code to be evaluated		Audited
4.11.3 Testing for HTML Injection (WSTG-CLNT-03)	WSTG-CLNT-03	Audited
Review custom JavaScript code (that are not from a known library) and enumerate objects that supply or are used as input to JavaScript functions to determine if it is possible to inject HTML indistinguishable from site content		Audited
4.11.4 Testing for Client Side URL Redirect (WSTG-CLNT-04)	WSTG-CLNT-04	Audited
Review custom JavaScript code (that are not from a known library) and enumerate objects that supply or are used as input to JavaScript functions to determine if it is possible to redirect the user to an arbitrary destination		Audited
4.11.5 Testing for CSS Injection (WSTG-CLNT-05)	WSTG-CLNT-05	Audited
Enumerate objects used as input to dynamically generate CSS, and determine if it is possible to leverage the generation to cause an impact		Audited
4.11.6 Testing for Client Side Resource Manipulation (WSTG-CLNT-06)	WSTG-CLNT-06	Audited
Enumerate objects used to determine a URL, and assert whether or not this can be modified to load arbitrary content into the page		Audited
4.11.7 Test Cross Origin Resource Sharing (WSTG-CLNT-07)	WSTG-CLNT-07	Audited
- Identify endpoints that implement CORS and ensure that the CORS configuration is secure or harmless		Audited
- Determine if XHR controls can be used to load arbitrary content by allowing the scope origin in the CORS headers (and if any possible filters can be bypassed)		Audited
4.11.8 Testing for Cross Site Flashing (WSTG-CLNT-08)	WSTG-CLNT-08	Audited
- Determine which parameters are passed to the flash object, and whether or not they can be leveraged in order to inject code or alter the object logic		Audited
- Determine whether or not the flash object loads remote flash objects, and whether or not any arbitrary object can be loaded		Audited
4.11.9 Testing for Clickjacking (WSTG-CLNT-09)	WSTG-CLNT-09	Audited
- Assess the security measures in place to protect against Clickjacking and determine if it is possible to frame the target application		Audited
- Determine if a malicious impact can be caused by framing the application		Audited
4.11.10 Testing WebSockets (WSTG-CLNT-10)	WSTG-CLNT-10	Audited
- Assess if WebSockets are used by the application		Audited
- Determine if the origin is properly verified		Audited
- Determine if the WebSocket is transferring data over encrypted connection (wss://)		Audited
- Determine that proper input validation/sanitation is performed		Audited
- Authentication and authorization should be tested as part of previous sections		Audited
4.11.11 Test Web Messaging (WSTG-CLNT-11)	WSTG-CLNT-11	Audited
- Determine if any event listeners for Cross Document Messaging are implemented, and if they can be leveraged in order to cause an impact		Audited
4.11.12 Test Browser Storage (WSTG-CLNT-12)	WSTG-CLNT-12	Audited
Determine whether the website is storing sensitive data in client-side storage:		Audited

- Local Storage		Audited
- Session Storage		Audited
- IndexedDB		Not in scope
- Cookies		Audited
4.11.13 Testing for Cross Site Script Inclusion (WSTG-CLNT-13)	WSTG-CLNT-13	Audited
Determine if sensitive data can be leaked using previously identified JavaScript injections (similar to CSRF but with JavaScript)		Audited

Appendix 1.1 - AngularJS 1,8.3.png

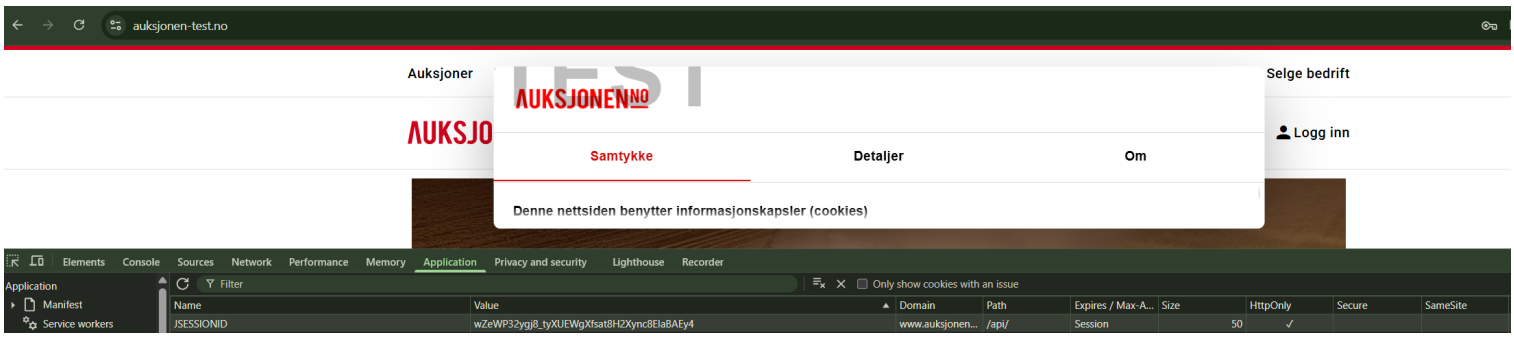


Appendix 1.1 - jQuery 2.2.2.png

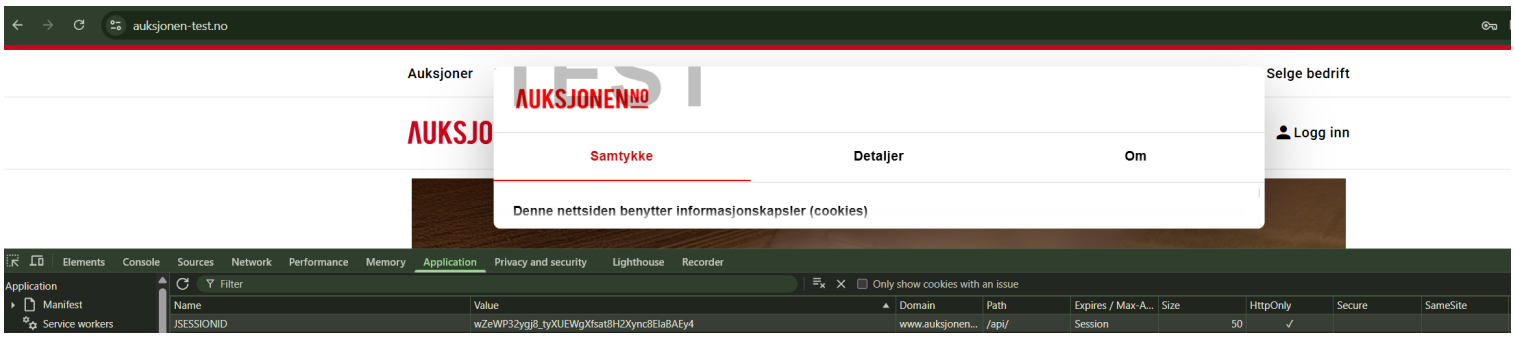


Appendix 1.1 - Registered Email.png

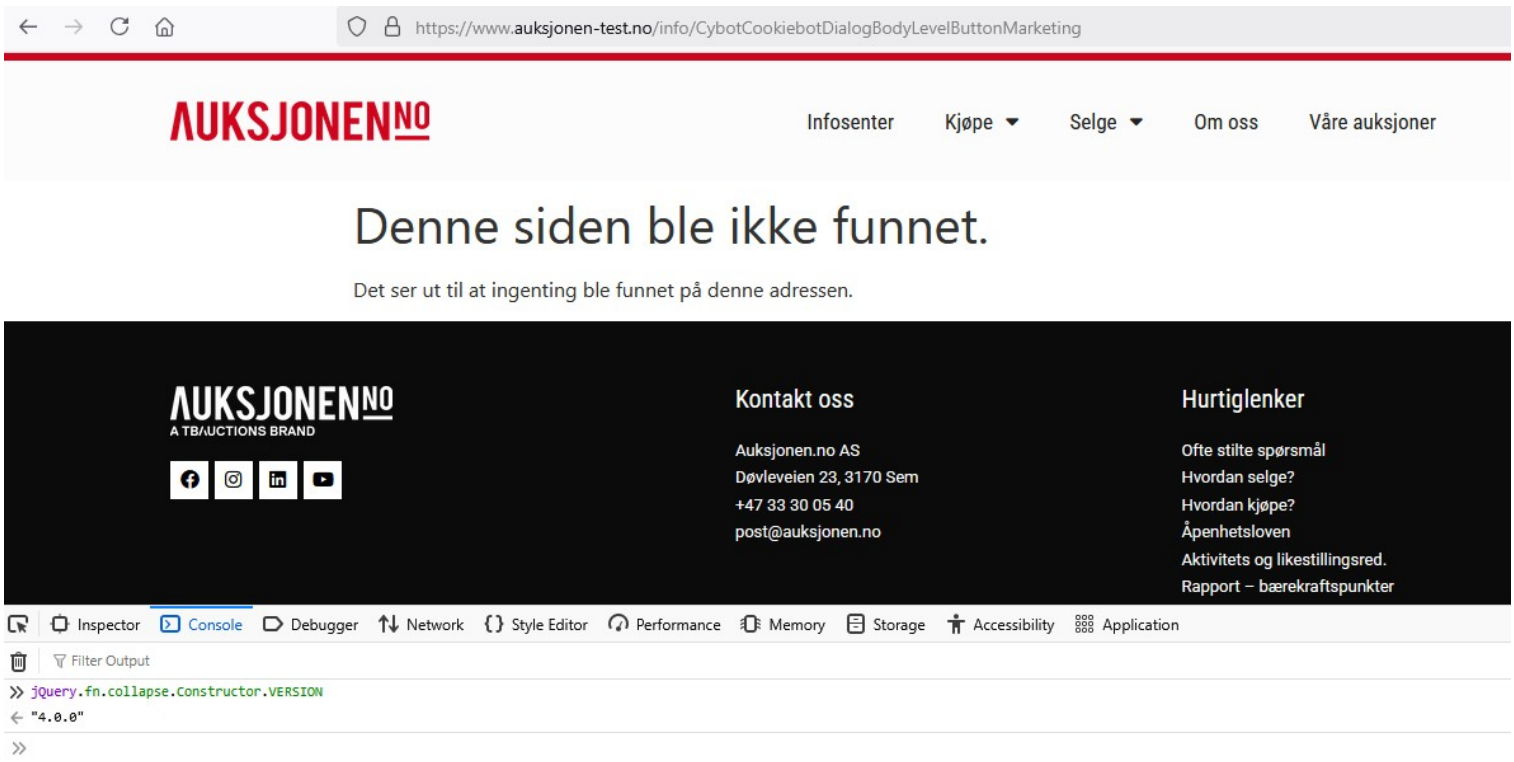
Appendix 1.1 - SameSite Not Set.png



Appendix 1.1 - Secure Not Set.png



Appendix 1.1 - Component_Bootstrap_4.0.0_End-of-Life.jpg



Appendix 1.2 - Unregistered Email.png