Web exposure self-check checklist for a terminal

For: IT managers, CIOs, security leads and webmasters of a container or multipurpose terminal who want to see what their own website, mail domain and public servers show to the outside

Go down the list for one name you own at a time: the main website, the truck appointment portal, the staff login, the mail domain. For each line, use a free tool or your browser to look from the outside, write what you found, and give the line a weight for your terminal before you start, not after. The checklist does not score your site for you: the lines that matter most are the ones your gate, your hauliers and your partners depend on. Only check names your organisation owns and is allowed to test.

Scope. This is what an attacker sees of your terminal from the public internet, before any login: the certificate on each name and when it expires, the security headers the site sends, the services and ports that answer, what your DNS and mail records say about who may send mail for you, and the files or messages that leak more than they should. You can check all of it yourself, from outside, with free tools. It is a surface review, not a penetration test, and Port Cyber Alliance does not certify sites. Check only names you own and are authorised to test.

Six families, 24 checks

• **T · Certificates and TLS**: The padlock: a valid certificate on every name, not expired, and only strong, current protocol versions.

• **H · Security headers**: The instructions the site sends the browser so a page cannot be framed, injected or downgraded.

• **S · Exposed services and ports**: What answers from the internet beyond the website, and what the server tells a stranger about itself.

• **D · DNS and email authentication**: What your public records say about who may send mail in your name, and whether old records still point somewhere.

• **L · Leaked information**: Source control, backups, errors, metadata and keys that reach the outside without anyone meaning to publish them.

• **W · Watching and responding**: Who owns the external surface, who gets told of a problem, and how often it is looked at again.

Result per name you check

• ok · checked from the outside and correct, with the date

• weak · present but below what the source asks: write what is missing

• fail · missing, wrong or exposed

• not checked yet

Weight per line (set before you start)

• must · a fail here is fixed first

• should · counts in the review

• nice to have

Score sheet · name checked: _____ · date: _____

Write ok / weak / fail / na in the Result column. No total: read the « must » lines first. Check only a name you own and are authorised to test.

Check	Weight	Result	Owner	Fix by
T1 · A valid certificate on every public name				
T2 · Certificate expiry is watched, not discovered				
T3 · Only current protocol versions and strong ciphers				
T4 · HTTP is sent to HTTPS, and HSTS is set				
H1 · A content security policy				
H2 · Clickjacking protection				
H3 · MIME-sniffing and referrer control				
H4 · Session cookies marked Secure, HttpOnly and SameSite				
H5 · No software versions announced in the headers				
S1 · Only the web ports answer from the internet				
S2 · No admin or staging interface reachable from outside				
S3 · Directory listing is off				
D1 · An SPF record that says who may send your mail				
D2 · Mail is signed with DKIM				
D3 · A DMARC policy, with reports going somewhere				
D4 · No dangling records an attacker can take over				
D5 · Transport security for mail and a closed zone				
L1 · No source control or configuration files served				
L2 · Error pages do not leak paths, versions or stack traces				
L3 · No secrets in the pages the browser receives				
L4 · You know what your public footprint already reveals				
W1 · A published way to report a problem (security.txt)				
W2 · Someone owns the external surface, with an inventory				
W3 · The surface is re-checked on a cadence and after every change				

T · Certificates and TLS

The padlock: a valid certificate on every name, not expired, and only strong, current protocol versions.

T1 · A valid certificate on every public name *(NIST SP 800-52r2 §3 · RFC 8446)*

Check from outside: Does every name you serve over HTTPS (the apex, www, the portal, the staff login, every live subdomain) present a certificate that matches the name, is inside its validity dates, and chains to a trusted root with its intermediate certificates sent?

Why: A name mismatch, a missing intermediate or an untrusted root makes the browser warn the visitor, and users learn to click through warnings, which is exactly what an attacker wants. NIST SP 800-52 Rev. 2 (section 3) requires server certificates that validate to a trusted root. The first thing an outside scan reports is which of your names fail.

A passing result: [ ] The certificate for each name, read in the browser (the padlock) or by a free TLS test site · [ ] That the chain is complete, with the intermediate certificates served, not only the leaf

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

T2 · Certificate expiry is watched, not discovered *(NIST SP 800-52r2 §3 (certificate management))*

Check from outside: For each certificate, how many days are left, who is alerted before it expires, and does renewal happen automatically or by a person who might be on leave?

Why: An expired certificate is the most common visible failure of a site, and it takes a public portal down for every haulier and partner at once. Certificates are public: anyone, including an attacker planning the day of maximum disruption, can read your expiry dates in the certificate transparency logs.

A passing result: [ ] The expiry date of each certificate and the number of days left today · [ ] Who or what renews it, and the alert that fires before the last week

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

T3 · Only current protocol versions and strong ciphers *(NIST SP 800-52r2 §3.1 · RFC 8446)*

Check from outside: Does the site accept only TLS 1.2 and TLS 1.3, with SSL 3.0, TLS 1.0 and TLS 1.1 and the weak ciphers turned off?

Why: NIST SP 800-52 Rev. 2 requires TLS 1.2 at least and recommends TLS 1.3, and the old versions carry known attacks. A free TLS test site grades this from the outside in a minute and names each weak cipher still offered.

A passing result: [ ] The list of protocol versions the server accepts, from a free TLS test · [ ] That SSL 3.0, TLS 1.0 and TLS 1.1 are refused, and no cipher is flagged weak

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

T4 · HTTP is sent to HTTPS, and HSTS is set *(RFC 6797 · CISA BOD 18-01)*

Check from outside: Does a request to the plain-HTTP address (port 80) redirect to HTTPS, and does the HTTPS response carry a Strict-Transport-Security header with a long max-age?

Why: Without the redirect and HSTS, a visitor's first request can be downgraded to plain HTTP and read or changed on the way. RFC 6797 defines HSTS for exactly this; CISA BOD 18-01 required federal sites to enforce HTTPS and HSTS. Both the redirect and the header are visible from the outside with no login.

A passing result: [ ] That http:// redirects to https:// for each name · [ ] The Strict-Transport-Security header and its max-age on the HTTPS response

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

H · Security headers

The instructions the site sends the browser so a page cannot be framed, injected or downgraded.

H1 · A content security policy *(OWASP Secure Headers Project (CSP))*

Check from outside: Does the site send a Content-Security-Policy header that says which origins its scripts, styles and frames may load from, rather than allowing anything?

Why: A content security policy is the browser-side limit that turns a cross-site scripting bug into a blocked request instead of a stolen session. The OWASP Secure Headers Project lists it first. Its absence is visible in one look at the response headers.

A passing result: [ ] The Content-Security-Policy header on the main pages and the login page · [ ] That it is not a permissive policy that allows every origin or inline script

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

H2 · Clickjacking protection *(OWASP HTTP Security Response Headers (X-Frame-Options, frame-ancestors))*

Check from outside: Can your pages, especially the login and the portal, be loaded inside a frame on another site, or do they send frame-ancestors (in the content security policy) or X-Frame-Options to forbid it?

Why: Without a frame limit, an attacker can overlay your login in an invisible frame and trick a user into clicking or typing into it (clickjacking). OWASP's cheat sheet asks for frame-ancestors or X-Frame-Options on every page that holds a session.

A passing result: [ ] The frame-ancestors directive or the X-Frame-Options header on the login and portal pages · [ ] A test that the login page refuses to render inside a frame

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

H3 · MIME-sniffing and referrer control *(OWASP HTTP Security Response Headers)*

Check from outside: Does the site send X-Content-Type-Options set to nosniff, and a Referrer-Policy that does not leak the full URL of internal pages to other sites?

Why: Without nosniff a browser may treat an uploaded file as a script; a loose referrer policy sends the full address of internal pages, including tokens in the query string, to every third-party resource. Both are one-line headers that OWASP recommends and a scan reports at once.

A passing result: [ ] The X-Content-Type-Options header set to nosniff · [ ] The Referrer-Policy header, set to no-referrer or same-origin for internal pages

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

H4 · Session cookies marked Secure, HttpOnly and SameSite *(OWASP Secure Headers Project (cookie attributes))*

Check from outside: Do the cookies that carry a session have the Secure, HttpOnly and SameSite flags, so they travel only over HTTPS, cannot be read by a script, and are not sent on cross-site requests?

Why: A session cookie without these flags can be stolen by a script or sent over plain HTTP. The flags are set in the Set-Cookie response header and are visible from the outside at login; OWASP treats them as a baseline.

A passing result: [ ] The Set-Cookie header at login, with Secure, HttpOnly and SameSite present · [ ] That no session cookie is sent over plain HTTP

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

H5 · No software versions announced in the headers ***(AA23-278A (misconfigurations) · OWASP Secure Headers Project)***

Check from outside: Do the Server and X-Powered-By headers (and the login page's hidden version strings) hide the exact version of the web server, the language and the content management system?

Why: An exact version in a header tells an attacker which known vulnerabilities to try first, with no effort. AA23-278A lists weak configurations among the ten most common findings. Trimming the version banner is free and the result is visible in the response headers.

A passing result: [ ] The Server and X-Powered-By headers, with no exact version number · [ ] The page source and generator tags, with no exact CMS version

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

S · Exposed services and ports

What answers from the internet beyond the website, and what the server tells a stranger about itself.

S1 · Only the web ports answer from the internet *(CISA Stuff Off Search · AA23-278A)*

Check from outside: From outside your network, do only the web ports (80 and 443) answer on your public addresses, with databases, remote desktop, SSH, file shares and management consoles not reachable from the open internet?

Why: Every open port is a door an attacker can knock on. The CISA "Stuff Off Search" guidance asks organisations to take management interfaces and services off the public internet and reach them over a private path instead. A free port scan of your own addresses shows what is open in minutes.

A passing result: [ ] A port scan of your own public addresses, showing only 80 and 443 open · [ ] That remote desktop, SSH, databases and admin consoles answer only over a private path or allow-list

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

S2 · No admin or staging interface reachable from outside *(33 CFR 101.650(a)(5),(b)(3) · AA23-278A)*

Check from outside: Are the content-management login, database tools, staging and test copies, and vendor dashboards behind a login, an allow-list or a private network, rather than open to anyone who guesses the address?

Why: A forgotten staging copy or an open admin panel is a common way in, and search engines index them. The USCG rule 33 CFR 101.650 requires least privilege and an accurate inventory of the systems that are reachable; an exposed admin panel is neither.

A passing result: [ ] The admin and database-tool addresses, each returning a login, an allow-list block or nothing from outside · [ ] A list of staging and test sites, each closed to the public internet

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

S3 · Directory listing is off *(AA23-278A (default configurations))*

Check from outside: When a visitor asks for a folder with no index page (for example an uploads or a backup folder), does the server return nothing or a 403, rather than an "Index of /" list of every file in it?

Why: An open directory listing hands an attacker the names of backups, documents and scripts to fetch next. It is a default that is easy to leave on and easy to test from the outside, one folder at a time.

A passing result: [ ] A request to a folder with no index page, returning 403 or no listing · [ ] No "Index of /" page anywhere a scan or a search engine finds

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

D · DNS and email authentication

What your public records say about who may send mail in your name, and whether old records still point somewhere.

D1 · An SPF record that says who may send your mail (RFC 7208)

Check from outside: Does your mail domain publish exactly one SPF record that lists the servers allowed to send mail in your name, and ends in a way that tells receivers to reject or mark the rest?

Why: Without SPF, anyone can send mail that appears to come from your terminal to your hauliers and partners. RFC 7208 defines SPF; a missing, duplicated or permissive SPF record is readable from the outside with a single DNS query.

A passing result: [ ] The SPF (TXT) record of your mail domain, with the real sending servers listed · [ ] That there is one SPF record, not two, and it does not end in a soft "allow all"

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

D2 · Mail is signed with DKIM (RFC 6376)

Check from outside: Is outgoing mail signed with DKIM, with the public key published in DNS, so a receiver can tell a real message from a forged one?

Why: DKIM (RFC 6376) lets a receiver verify that a message really came from your domain and was not changed. Together with SPF it is what a DMARC policy relies on. The public key is in DNS and the signature is in every message header.

A passing result: [ ] The DKIM public key record in DNS for each selector you use · [ ] A real message showing a passing DKIM signature

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

D3 · A DMARC policy, with reports going somewhere (RFC 7489 · CISA BOD 18-01)

Check from outside: Does your domain publish a DMARC record, is its policy stronger than "do nothing" (quarantine or reject), and do the aggregate reports go to a mailbox someone reads?

Why: DMARC (RFC 7489) ties SPF and DKIM together and tells receivers what to do with mail that fails: a policy of "none" only watches, it does not protect. CISA BOD 18-01 required federal domains to reach a reject policy. The record is public and a free checker reads it from the outside.

A passing result: [ ] The DMARC (TXT) record, with its policy and the address its reports go to · [ ] That the policy is quarantine or reject, or a dated plan to get there

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

D4 · No dangling records an attacker can take over (CISA Stuff Off Search (reduce exposed assets))

Check from outside: Do any of your DNS records still point at a cloud service, bucket or subdomain you no longer use, which someone else could claim and serve content from your name?

Why: A record left pointing at a decommissioned service lets an attacker register that service and take over the subdomain, serving pages or collecting mail under your brand. It is invisible to your staff and visible to anyone who enumerates your subdomains from the outside.

A passing result: [ ] A list of your subdomains and where each record points, with the dead ones removed · [ ] That no record points at a service you no longer control

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

D5 · Transport security for mail and a closed zone (RFC 8461 · RFC 8460)

Check from outside: Does your domain publish MTA-STS (and TLS reporting) so other mail servers must use TLS to reach you, and does your authoritative DNS refuse a full zone transfer to a stranger?

Why: MTA-STS (RFC 8461) stops a downgrade of mail-in-transit to plain text; an open zone transfer (AXFR) hands an attacker your whole list of names at once. Both are checked from the outside with a single query.

A passing result: [ ] The MTA-STS policy and TLS-RPT record, if you accept mail · [ ] A zone-transfer request from an outside address, refused

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

L · Leaked information

Source control, backups, errors, metadata and keys that reach the outside without anyone meaning to publish them.

L1 · No source control or configuration files served *(AA23-278A · dms-logistics/portcyberalliance.org#206 (CyberScan incident))*

Check from outside: Does the web server refuse to serve version-control folders (for example a .git directory), environment and configuration files, and leftover archives (.zip, .sql, .bak) to anyone who asks for them by name?

Why: An exposed .git folder or a .env file can hand over the whole source and its secrets; the incident behind this page began with a key committed in plaintext. These files are fetched by name from the outside, so a scan finds them even when nothing links to them.

A passing result: [ ] A request for the common paths (a .git folder, a .env file, a database dump), each returning 403 or nothing · [ ] No configuration or backup file reachable over HTTP

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

L2 · Error pages do not leak paths, versions or stack traces *(AA23-278A (default configurations))*

Check from outside: When a page errors, does the visitor see a plain message, rather than a stack trace, a file path, a database query or the exact software version?

Why: A detailed error page is a map of your server: the framework, the file layout and sometimes a query an attacker can reuse. Turning off debug output in production is free, and the leak is visible by asking for a page that errors.

A passing result: [ ] A deliberately bad request, returning a plain error with no path, trace or version · [ ] That debug mode is off on the live site

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

L3 · No secrets in the pages the browser receives *(AA23-278A · dms-logistics/portcyberalliance.org#206)*

Check from outside: Do the HTML, JavaScript and comments that the site sends to every visitor contain no API keys, access tokens, internal hostnames or credentials?

Why: Anything in the page source is public: a key left in a script is read by the first person who looks, and found at scale by free secret-scanning tools. The CyberScan incident behind this page started from a committed secret; secrets in served code are the same leak, one step closer to the attacker.

A passing result: [ ] A search of the served HTML and JavaScript for keys, tokens and passwords · [ ] That internal hostnames and build paths are not left in comments

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

L4 · You know what your public footprint already reveals *(CISA Stuff Off Search · RFC 6797 (public HSTS context))*

Check from outside: Have you looked at what the certificate transparency logs, public search engines and internet-wide scan services already show about your names, subdomains and exposed services?

Why: Certificates are logged publicly, search engines index what they reach, and internet-wide scanners catalogue open services, so an attacker starts from a free map of your surface. Reading that same public map is the cheapest way to find a name or a service you forgot you had.

A passing result: [ ] The list of your names from the certificate transparency logs, each still meant to be public · [ ] A search of the public scan services for your addresses, with nothing exposed by surprise

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

W · Watching and responding

Who owns the external surface, who gets told of a problem, and how often it is looked at again.

W1 · A published way to report a problem (security.txt) (RFC 9116)

Check from outside: Does your site publish a security.txt file at the well-known path, giving a contact for anyone who finds a problem, so a finder tells you instead of posting it?

Why: RFC 9116 defines security.txt so a researcher who spots an exposed file or an expiring certificate has one clear, public address to warn you. Without it, the finder either gives up or goes public. The file is a few lines and is itself checked from the outside.

A passing result: [ ] The security.txt file at /.well-known/security.txt, with a contact and an expiry date · [ ] That the contact reaches someone who acts on it

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

W2 · Someone owns the external surface, with an inventory (33 CFR 101.650(b)(3) · NIS2 Art. 21(2))

Check from outside: Is there one list of your public names, their certificates, their DNS and mail records and their open services, with an owner for each, kept up to date?

Why: You cannot watch what you have not written down, and the surface grows every time a project adds a subdomain. The USCG rule asks for an accurate inventory of network-connected systems; the names and records that reach the internet are part of it, and the inventory is what a review starts from.

A passing result: [ ] The inventory of public names, certificates, records and services, with an owner each · [ ] The date it was last reconciled with a fresh outside scan

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

W3 · The surface is re-checked on a cadence and after every change (IMO Rev.4 §3.5 · NIS2 Art. 21(2))

Check from outside: Do you run this outside check on a schedule and again whenever a certificate renews, a subdomain is added or a service is published, with the result going to the owner?

Why: A surface is correct the day you check it and drifts afterwards: a cert renews wrong, a staging site goes up, a record is added. IMO Rev.4 and NIS2 Art. 21 both ask for the regular review and improvement of the measures; a repeatable outside scan is how you see the drift before an attacker does.

A passing result: [ ] The schedule of the outside check and who receives the result · [ ] A change (a new subdomain, a renewed certificate) that triggered a fresh check

Weight: [ ] must [ ] should [ ] nice Result: _____ Owner: _____ Fix by: _____

Free tools to check your own site

Two ways to look: public online checkers that run in a browser (described in general terms, no brand named) and open-source tools you run yourself. The open-source projects are examples only, in alphabetical order inside each category, in their publisher's own words, with the licence their repository declares (read on 11 November 2026). Not a ranking, not a test. Commercial products exist in every category. Run any scanner only against names and addresses you own.

Certificate and TLS checkers

Read a site's certificate, its chain and expiry, and grade the protocol versions and ciphers it accepts, from the outside. Public online TLS test sites do the same in a browser, with no install.

• **SSLyze**: « Fast and powerful SSL/TLS scanning library. » · licence: AGPL-3.0 · https://github.com/nabla-c0d3/sslyze

• **testssl.sh**: « Testing TLS/SSL encryption anywhere on any port. » · licence: GPL-2.0 · https://testssl.sh/

Web server and header scanners

Fetch a site's response headers and pages and report missing security headers, server banners, directory listings and known weak spots. Public online header-grading sites show the headers alone, with no install.

• **Nikto**: « Nikto web server scanner. » · licence: GPL-2.0 · https://github.com/sullo/nikto

• **OWASP ZAP**: « The world's most widely used web app scanner. Free and open source. » · licence: Apache-2.0 · https://www.zaproxy.org/

Port and service discovery

List which ports answer on an address and what service each one runs, so you see what is reachable beyond the website. Run them only against addresses you own.

• **Naabu**: « A fast port scanner written in go with a focus on reliability and simplicity. » · licence: MIT · https://github.com/projectdiscovery/naabu

• **Nmap**: « the Network Mapper — a free and open source utility for network discovery and security auditing. » · licence: Nmap Public Source License (NPSL) · https://nmap.org/

Attack surface and subdomain discovery

Enumerate the subdomains and assets attached to a domain, from public sources such as the certificate transparency logs, so you find the names you forgot. Public online subdomain finders do a lighter version in a browser.

• **OWASP Amass**: « In-depth attack surface mapping and asset discovery. » · licence: see the repository's licence file · https://github.com/owasp-amass/amass

• **Subfinder**: « Fast passive subdomain enumeration tool. » · licence: MIT · https://github.com/projectdiscovery/subfinder

DNS and email authentication checkers

Read your SPF, DKIM and DMARC records and your DNS delegation and report what is missing or wrong. Public online SPF/DKIM/DMARC checkers read the same records in a browser.

• **checkdmarc**: « A Python module and command line tool for validating a domain's DMARC, SPF, DKIM, and more. » · licence: Apache-2.0 · https://github.com/domainaware/checkdmarc

• **Zonemaster**: « a software that validates the quality of the delegation and configuration of a DNS zone. » · licence: BSD-2-Clause · https://github.com/zonemaster/zonemaster

Leaked-secret and exposed-file scanners

Scan your own source, pages and repositories for API keys, tokens and passwords, and for configuration or backup files that reach the web. The same family that caught the incident behind this page.

• **Gitleaks**: « Find secrets with Gitleaks. Detect and prevent secrets in your code. » · licence: MIT · https://github.com/gitleaks/gitleaks

• **TruffleHog**: « Find, verify, and analyze leaked credentials. » · licence: AGPL-3.0 · https://github.com/trufflesecurity/trufflehog

Sources (public, dated)

IETF, RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3, August 2018. https://www.rfc-editor.org/rfc/rfc8446

NIST SP 800-52 Rev. 2, Guidelines for the Selection, Configuration, and Use of TLS Implementations, August 2019. https://csrc.nist.gov/pubs/sp/800/52/r2/final

IETF, RFC 6797, HTTP Strict Transport Security (HSTS), November 2012. https://www.rfc-editor.org/rfc/rfc6797

OWASP Secure Headers Project and HTTP Security Response Headers Cheat Sheet, read on 11 November 2026. https://owasp.org/www-project-secure-headers/

IETF, RFC 7208, Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1, April 2014. https://www.rfc-editor.org/rfc/rfc7208

IETF, RFC 6376, DomainKeys Identified Mail (DKIM) Signatures, September 2011. https://www.rfc-editor.org/rfc/rfc6376

IETF, RFC 7489, Domain-based Message Authentication, Reporting, and Conformance (DMARC), March 2015. https://www.rfc-editor.org/rfc/rfc7489

IETF, RFC 8461, SMTP MTA Strict Transport Security (MTA-STS), September 2018. https://www.rfc-editor.org/rfc/rfc8461

IETF, RFC 8460, SMTP TLS Reporting (TLS-RPT), September 2018. https://www.rfc-editor.org/rfc/rfc8460

IETF, RFC 9116, A File Format to Aid in Security Vulnerability Disclosure (security.txt), April 2022. https://www.rfc-editor.org/rfc/rfc9116

CISA, Binding Operational Directive 18-01, Enhance Email and Web Security, 16 October 2017. https://www.cisa.gov/news-events/directives/bod-18-01-enhance-email-and-web-security

CISA, Stuff Off Search (S.O.S.), reduce internet attack surface, read on 11 November 2026. https://www.cisa.gov/resources-tools/resources/stuff-off-search

NSA and CISA, advisory AA23-278A, Top Ten Cybersecurity Misconfigurations, 5 October 2023. https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-278a

eCFR, 33 CFR 101.650, Cybersecurity measures (US Coast Guard rule), in force since 16 July 2025, read on 11 November 2026. https://www.ecfr.gov/current/title-33/chapter-I/subchapter-H/part-101/subpart-F/section-101.650

IMO, MSC-FAL.1/Circ.3/Rev.4, Guidelines on maritime cyber risk management, 28 May 2026. https://wwwcdn.imo.org/localresources/en/OurWork/Security/Documents/MSC-FAL.1-Circ.3-Rev.4.pdf

Directive (EU) 2022/2555 (NIS2), Article 21(2), 14 December 2022. https://eur-lex.europa.eu/eli/dir/2022/2555/oj/eng

Licence

Creative Commons Attribution 4.0 International (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/ . You may copy, adapt and share this checklist, including commercially, if you credit Port Cyber Alliance and link to https://portcyberalliance.org/resources/tools-directory/web-exposure-scan/ . Generic: no terminal, port or client is named, no figure is invented, no tool is ranked or priced. This checklist is not legal advice, and a scan is your responsibility: check only names you own and are authorised to test. Port Cyber Alliance does not certify sites or terminals.