Website test: gettr.com

70%

- ✓ Reachable via modern internet address (IPv6)
- ✗ Domain name *not* signed (DNSSEC)
- ✗ Connection *not* or insufficiently secured (HTTPS)
- ! One or more recommended application security options *not* set (Security options)

✓ Modern address (IPv6)

Well done! Your website is reachable for visitors using a modern internet address (IPv6), making it fully part of the modern Internet.

Name servers

✓ IPv6 addresses for name servers

Verdict:
Two or more name servers of your domain have an IPv6 address.

Technical details:

Name server	IPv6 address	IPv4 address
`mira.ns.cloudflare.com.`	`2606:4700:50::adf5:3acc`	`173.245.58.204`
`...`	`2803:f800:50::6ca2:c0cc`	`172.64.32.204`
`...`	`2a06:98c1:50::ac40:20cc`	`108.162.192.204`
`leo.ns.cloudflare.com.`	`2606:4700:58::a29f:2cf1`	`108.162.195.241`
`...`	`2a06:98c1:50::ac40:23f1`	`162.159.44.241`
`...`	`2803:f800:50::6ca2:c3f1`	`172.64.35.241`

Test explanation:
We check if your domain name has at least two name servers with an IPv6 address. This is consistent with the "Technical requirements for the registration and use of .nl domain names" d.d. 13 November 2017 by SIDN (.nl TLD registry) that require each .nl domain to have at least two name servers.

✓ IPv6 reachability of name servers

Verdict:
All name servers that have an IPv6 address are reachable over IPv6.

Test explanation:
We check if all name servers, that have an AAAA record with IPv6 address, are reachable over IPv6.

Web server

✓ IPv6 addresses for web server

Verdict:
At least one of your web servers has an IPv6 address.

Technical details:

Web server	IPv6 address	IPv4 address
gettr.com	2600:9000:21c7:da00:1a:d6d6:9e80:93a1	65.9.83.96
...	2600:9000:21c7:ce00:1a:d6d6:9e80:93a1	65.9.83.47
...	2600:9000:21c7:8000:1a:d6d6:9e80:93a1	65.9.83.114
...	2600:9000:21c7:f000:1a:d6d6:9e80:93a1	65.9.83.26
...	2600:9000:21c7:0:1a:d6d6:9e80:93a1	-
...	2600:9000:21c7:d800:1a:d6d6:9e80:93a1	-
...	2600:9000:21c7:9800:1a:d6d6:9e80:93a1	-
...	2600:9000:21c7:2200:1a:d6d6:9e80:93a1	-

Test explanation:
We check if there is at least one AAAA record with IPv6 address for your web server.

✓ IPv6 reachability of web server

Verdict:
All your web servers with an IPv6 address are reachable over IPv6.

Test explanation:
We check if we can connect to your web server(s) over IPv6 on any available ports (80 and/or 443). We test all IPv6 addresses that we receive from your name servers. A partial score will be given if not all IPv6 addresses are reachable. If an IPv6 address is (syntactically) invalid, we consider it unreachable.

✓ Same website on IPv6 and IPv4

Verdict:
Your website on IPv6 seems to be the same as your website on IPv4.

Test explanation:
We compare the web content that we receive from your web server over both IPv6 and IPv4 on any available ports (80 and/or 443). In case there are multiple IPv6 and IPv4

addresses, we pick one IPv6 address and one IPv4 address. If the content difference is not higher than 10%, we expect the main web content to be the same. Therefore websites with small differences (for example due to changing ads) will pass this subtest as well.

Signed domain name (DNSSEC)

Too bad! Your domain is *not* signed with a valid signature (DNSSEC). Therefore visitors with enabled domain signature validation, are *not* protected against manipulated translation from your domain into rogue internet addresses. You should ask your name server operator (often your registrar and/or hosting provider) to enable DNSSEC.

DNSSEC existence

Verdict:
Your domain is insecure, because it is *not* DNSSEC signed.

Technical details:

Domain	Registrar
gettr.com	Epik Inc.

Test explanation:
We check if your domain, more specifically its SOA record, is DNSSEC signed.

If a domain redirects to another domain via `CNAME`, then we also check if the CNAME domain is signed (which is conformant with the DNSSEC standard). If the CNAME domain is not signed, the result of this subtest will be negative.

Note: the validity of the signature is not part of this subtest, but part of the next subtest.

DNSSEC validity

Verdict:
This subtest did not run, because either a parent test that this subtest depends on gave a negative result, or not enough information was available to run this subtest.

Technical details:

Domain	Status
gettr.com	insecure

Test explanation:
We check if your domain, more specifically its SOA record, is signed with a valid signature making it 'secure'.

If a domain redirects to another signed domain via `CNAME`, then we also check if the signature of the CNAME domain is valid (which is conformant with the DNSSEC standard). If the signature of the CNAME domain is not valid, the result of this subtest will be negative.

Secure connection (HTTPS)

Too bad! The connection with your website is *not* or *insufficiently* secured (HTTPS). Therefore information in transit between your website and its visitors is *not* sufficiently protected against eavesdropping and tampering. You should ask your hosting provider to enable HTTPS and to configure it securely.

HTTP

HTTPS available

Verdict:
Your website offers HTTPS.

Technical details:

Web server IP address	HTTPS existent
2600:9000:2104:1200:1a:d6d6:9e80:93a1	yes
13.227.219.60	yes

Test explanation:
We check if your website is reachable on HTTPS.

If so, we also check in the below subtests whether HTTPS is configured sufficiently secure in conformance with the 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL.

HTTPS guarantees the confidentiality and integrity of the exchanged information. Because it is situation depended how (privacy) sensitive and valuable information is, a secure HTTPS configuration is important for every website. Even trivial, public information could be extremely sensitive and valuable for a user. Note: for performance reasons the HTTPS test section only runs for the first available IPv6 and IPv4 address.

HTTPS redirect

Verdict:

Your web server automatically redirects visitors from HTTP to HTTPS on the same domain.

Technical details:

Web server IP address	HTTPS redirect
2600:9000:2104:1200:1a:d6d6:9e80:93a1	yes
13.227.219.60	yes

Test explanation:
We check if your web server automatically redirects visitors from HTTP to HTTPS on the same domain (through a 3xx redirect status code like 301 and 302), or if it offers support for only HTTPS and not for HTTP.

In case of redirecting, a domain should firstly upgrade itself by redirecting to its HTTPS version before it may redirect to another domain. This also ensures that the HSTS policy will be accepted by the web browser. Examples of correct redirect order:

- `http://example.nl` ⇒ `https://example.nl` ⇒ `https://www.example.nl`
- `http://www.example.nl` ⇒ `https://www.example.nl`

Note that this subtest only tests if the given domain correctly redirects from HTTP to HTTPS. An eventual further redirect to a different domain (including a subdomain of the tested domain) is not tested. You could start a seperate test to test such a domain that is being redirected to.

See 'Web application guidelines, detailed version' from NCSC-NL, guideline U/WA.05 (in Dutch).

HTTP compression

Verdict:
Your web server does not support HTTP compression.

Technical details:

Web server IP address	HTTP compression
2600:9000:2104:1200:1a:d6d6:9e80:93a1	no
13.227.219.60	no

Test explanation:
We test if your web server supports HTTP compression.

HTTP compression makes the secure connection with your webserver vulnerable for the BREACH attack. However HTTP compression is commonly used to make more efficient use of available bandwidth. Consider the trade-offs involved with HTTP compression. If you choose to use HTTP compression, verify if it is possible to mitigate related attacks at the application level. An example of such a measure is limiting the extent to which an attacker can influence the response of a server.

This subtest checks if the web server on root directory level supports HTTP compression. However it does not check additional website sources like images and scripts.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B7-1 and table 11.

Requirement level: Optional

Compression option

- Good: No compression
- Sufficient: Application-level compression (in this case HTTP compression)
- Insufficient: TLS compression

HTTS

Verdict:
Your web server does *not* offer an HSTS policy.

Technical details:

Web server IP address	HSTS policy
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if your web server supports HSTS.

Browsers remember HSTS per (sub) domain. Not adding a HSTS header to every (sub) domain (in a redirect chain) might leave users vulnerable to MITM attacks. Therefore we check for HSTS on the first contact i.e. before any redirect.

HSTS forces a web browser to connect directly via HTTPS when revisiting your website. This helps preventing man-in-the-middle attacks. We consider a HSTS cache validity period of *at least* 1 year (`max-age=31536000`) to be sufficiently secure. A long period is beneficial because it also protects infrequent visitors. However if you want to stop supporting HTTPS (which is generally a poor idea), you will have to wait longer until the validity of the HSTS policy in all browsers that visited your website, has expired.

See 'Web application guidelines, detailed version' from NCSC-NL, guideline U/WA.05 (in Dutch).

TLS

✓ TLS version

Verdict:
Your web server supports secure TLS versions only.

Technical details:

Web server IP address	Affected TLS version	Status
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None	None
13.227.219.60	None	None

Test explanation:
We check if your web server supports secure TLS versions only.

A web server may support more than one TLS version.

Note that browser makers have announced that they will stop supporting TLS 1.1 and 1.0. This will cause websites that do not support TLS 1.2 and/or 1.3 to be unreachable.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B1-1 and table 1 (in English).

Version

- Good: TLS 1.3
- Sufficient: TLS 1.2
- Phase out: TLS 1.1 and 1.0
- Insufficient: SSL 3.0, 2.0 and 1.0

✓ Ciphers (Algorithm selections)

Verdict:
Your web server supports secure ciphers only.

Technical details:

Web server IP address	Affected ciphers
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:

We check if your web server only supports secure, i.e. 'Good' and/or 'Sufficient', ciphers (also known as algorithm selections).

An algorithm selection consists of ciphers for four cryptographic functions: 1) key exchange, 2) certificate verification, 3) bulk encryption, and 4) hashing. A web server may support more than one algorithm selection.

- Since TLS 1.3, the term 'cipher suite' only comprises ciphers used for bulk encryption and hashing. When using TLS 1.3 the ciphers for key exchange and certificate verification are negotiable and not part of the naming of the cipher suite. Because this makes the term 'cipher suite' ambiguous, NCSC-NL uses the term 'algorithm selection' to comprise all four cipher functions.
- NCSC-NL uses the IANA naming convention for algorithm selections. Internet.nl uses the OpenSSL naming convention. Since TLS 1.3 OpenSSL follows the IANA naming convention. A translation between both can be found in the OpenSSL documentation.
- Note that ciphers using PSK or SRP for key exchange (which are not sufficiently secure) are not detected in this test due to a limitation related to our testing method.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B2-1 to B2-4 and table 2, 4, 6 and 7 (in English).

Below you find 'Good', 'Sufficient' and 'Phase out' algorithm selections in the by NCSC-NL prescribed order, based on appendix C of the 'IT Security Guidelines for Transport Layer Security (TLS)'. Behind every algorithm selection is the minimum TLS version (e.g. [1.2]) that supports this algorithm selection and that is at least 'Phase out'.

Good:

- `ECDHE-ECDSA-AES256-GCM-SHA384` (`TLS_AES_256_GCM_SHA384` in 1.3) [1.2]
- `ECDHE-ECDSA-CHACHA20-POLY1305` (`TLS_CHACHA20_POLY1305_SHA256` in 1.3) [1.2]
- `ECDHE-ECDSA-AES128-GCM-SHA256` (`TLS_AES_128_GCM_SHA256` in 1.3) [1.2]
- `ECDHE-RSA-AES256-GCM-SHA384` (`TLS_AES_256_GCM_SHA384` in 1.3) [1.2]
- `ECDHE-RSA-CHACHA20-POLY1305` (`TLS_CHACHA20_POLY1305_SHA256` in 1.3) [1.2]
- `ECDHE-RSA-AES128-GCM-SHA256` (`TLS_AES_128_GCM_SHA256` in 1.3) [1.2]

Sufficient:

- `ECDHE-ECDSA-AES256-SHA384` [1.2]
- `ECDHE-ECDSA-AES256-SHA` [1.0]
- `ECDHE-ECDSA-AES128-SHA256` [1.2]

- `ECDHE-ECDSA-AES128-SHA` [1.0]
- `ECDHE-RSA-AES256-SHA384` [1.2]
- `ECDHE-RSA-AES256-SHA` [1.0]
- `ECDHE-RSA-AES128-SHA256` [1.2]
- `ECDHE-RSA-AES128-SHA` [1.0]
- `DHE-RSA-AES256-GCM-SHA384` [1.2]
- `DHE-RSA-CHACHA20-POLY1305` [1.2]
- `DHE-RSA-AES128-GCM-SHA256` [1.2]
- `DHE-RSA-AES256-SHA256` [1.2]
- `DHE-RSA-AES256-SHA` [1.0]
- `DHE-RSA-AES128-SHA256` [1.2]
- `DHE-RSA-AES128-SHA` [1.0]

Phase out:

- `ECDHE-ECDSA-DES-CBC3-SHA` [1.0]
- `ECDHE-RSA-DES-CBC3-SHA` [1.0]
- `DHE-RSA-DES-CBC3-SHA` [1.0]
- `AES256-GCM-SHA384` [1.2]
- `AES128-GCM-SHA256` [1.2]
- `AES256-SHA256` [1.2]
- `AES256-SHA` [1.0]
- `AES128-SHA256` [1.2]
- `AES128-SHA` [1.0]
- `DES-CBC3-SHA` [1.0]

Cipher order

Verdict:
Your web server enforces its own cipher preference ('I'), and offers ciphers in accordance with the prescribed ordering ('II').

Technical details:

Web server IP address	First found affected cipher pair
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:

We check if your web server enforces its own cipher preference ('I'), and offers ciphers in accordance with the prescribed ordering ('II').

When your web server supports 'Good' ciphers only, this test is not applicable as the ordering has no significant security advantage.

I. *Server enforced cipher preference*: The web server enforces its own cipher preference while negotiating with a web browser, and does not accept any preference of the web browser;

II. *Prescribed ordering*: Ciphers are offered by the web server in accordance with the prescribed order where 'Good' is preferred over 'Sufficient' over 'Phase out' ciphers.

In the above table with technical details **only the first found out of prescribed order algorithm selections** are listed.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B2-5.

✓ Key exchange parameters

Verdict:
Your web server supports secure parameters for Diffie-Hellman key exchange.

Technical details:

Web server IP address	Affected parameters
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if the public parameters used in Diffie-Hellman key exchange by your web server are secure.

ECDHE: The security of elliptic curve Diffie-Hellman (ECDHE) ephemeral key exchange depends on the used elliptic curve. We check if the bit-length of the used elliptic curves is a least 224 bits. Currently we are not able to check the elliptic curve name.

DHE: The security of Diffie-Hellman Ephemeral (DHE) key exchange depends on the lengths of the public and secret keys used within the chosen finite field group. We test if your DHE public key material uses one of the predefined finite field groups that are specified in RFC 7919. Self-generated groups are 'Insufficient'.

The larger key sizes required for the use of DHE come with a performance penalty. Carefully evaluate and use ECDHE instead of DHE if you can.

RSA as an alternative: Besides ECDHE and DHE, RSA can be used for key exchange. However, it is at risk of becoming insufficiently secure (current status 'phase out'). The RSA public parameters are tested in the subtest 'Public key of certificate'. Note that RSA is considered as 'good' for certificate verification.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B5-1 and table 9 for ECDHE, and guideline B6-1 and table 10 for DHE (in English).

Elliptic curve for ECDHE

- Good: `secp384r1`, `secp256r1`, `x448`, and `x25519`
- Phase out: `secp224r1`
- Insufficient: Other curves

Finite field group for DHE

- Sufficient:
  - ffdhe4096 (RFC 7919)
    sha256 checksum:
    `64852d6890ff9e62eecd1ee89c72af9af244dfef5b853bcedea3dfd7aade22b3`
  - ffdhe3072 (RFC 7919)
    sha256 checksum:
    `c410cc9c4fd85d2c109f7ebe5930ca5304a52927c0ebcb1a11c5cf6b2386bbab`
  - Note that we also test for ffdhe8192 and ffdhe6144. However their limited gain in security rarely outweighs the loss in performance.
- Phase out:
  - ffdhe2048 (RFC 7919)
    sha256 checksum:
    `9ba6429597aeed2d8617a7705b56e96d044f64b07971659382e426675105654b`
- Insufficient: Other groups

Hash function for key exchange

Verdict:
Your web server supports a secure hash function for key exchange.

Technical details:

Web server IP address	SHA2 support for signatures
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	yes

Web server IP address	SHA2 support for signatures
13.227.219.60	yes

Test explanation:
We check if your web server supports secure hash functions to create the digital signature during key exchange.

The web server uses a digital signature during the key exchange to prove ownership of the secret key corresponding to the certificate. The web server creates this digital signature by signing the output of a hash function.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, table 5.

Requirement level: Recommended

SHA2 support for signatures

- Good: Yes (SHA-256, SHA-384 or SHA-512 supported)
- Phase out: No (SHA-256, SHA-384 of SHA-512 *not* supported)

✓ TLS compression

Verdict:
Your web server does not support TLS compression.

Technical details:

Web server IP address	TLS compression
2600:9000:2104:1200:1a:d6d6:9e80:93a1	no
13.227.219.60	no

Test explanation:
We check if your web server supports TLS compression.

The use of compression can give an attacker information about the secret parts of encrypted communication. An attacker that can determine or control parts of the data sent can reconstruct the original data by performing a large number of requests. TLS compression is used so rarely that disabling it is generally not a problem.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B7-1 and table 11 (in English).

Compression option

- Good: No compression
- Sufficient: Application-level compression (in this case HTTP compression)
- Insufficient: TLS compression

✓ Secure renegotiation

Verdict:
Your web server supports secure renegotiation.

Technical details:

Web server IP address	Secure renegotiation
2600:9000:2104:1200:1a:d6d6:9e80:93a1	yes
13.227.219.60	yes

Test explanation:
We check if your web server supports secure renegotiation.

Older versions of TLS (prior to TLS 1.3) allow forcing a new handshake. This so-called renegotiation was insecure in its original design. The standard was repaired and a safer renegotiation mechanism was added. The old version is since called insecure renegotiation and should be disabled.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B8-1 and table 12 (in English).

Insecure renegotiation

- Good: Off (or N/A for TLS 1.3)
- Insufficient: On

✓ Client-initiated renegotiation

Verdict:
Your web server does not allow for client-initiated renegotiation.

Technical details:

Web server IP address	Client-initiated renegotiation
2600:9000:2104:1200:1a:d6d6:9e80:93a1	no
13.227.219.60	no

Test explanation:
We check if a client (usually a web browser) can initiate a renegotiation with your web server.

Allowing clients to initiate renegotiation is generally not necessary and opens a web server to DoS attacks inside a TLS connection. An attacker can perform similar DoS attacks without client-initiated renegotiation by opening many parallel TLS connections, but these are easier to detect and defend against using standard mitigations. Note that client-initiated renegotiation impacts availability and not confidentiality.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B8-1 and table 13 (in English).

Requirement level: Optional

Client-initiated renegotiation

- Good: Off (or N/A for TLS 1.3)
- Sufficient: On

✓ 0-RTT

Verdict:
Your web server does not support 0-RTT.

Technical details:

Web server IP address	0-RTT
2600:9000:2104:1200:1a:d6d6:9e80:93a1	no
13.227.219.60	no

Test explanation:
We check if your web server supports Zero Round Trip Time Resumption (0-RTT).

0-RTT is an option in TLS 1.3 that transports application data during the first handshake message. 0-RTT does not provide protection against replay attacks at the TLS layer and therefore should be disabled. Although the risk can be mitigated by not allowing 0-RTT for non-idempotent requests, such a configuration is often not trivial, reliant on application logic and thus error prone.

If your web server does not support TLS 1.3, the test is not applicable. For web servers that support TLS 1.3, the index `/` page of the website is fetched using TLS 1.3 and the amount of early data support indicated by the server is checked. When more than zero, a second connection is made re-using the TLS session details of the first connection but sending the HTTP request *before* the TLS handshake (i.e. no round trips (0-RTT) needed before application data to the server). If the TLS handshake is completed and the web server responds with any non-HTTP 425 Too Early response, then the web server is considered to support 0-RTT.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B9-1 and table 14 (in English).

0-RTT

- Good: Off (or N/A prior to TLS 1.3)
- Insufficient: On

OCSP stapling

Verdict:
Your web server does *not* support OCSP stapling.

Technical details:

Web server IP address	OCSP stapling
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	`no`
`13.227.219.60`	`yes`

Test explanation:
We check if your web server supports the TLS Certificate Status extension also known as OCSP stapling.

The web browser can verify the validity of the certificate presented by the web server by contacting the certificate authority using the OCSP protocol. OCSP provides a certificate authority with information on browsers communicating to the web server: this may be a privacy risk. A web server can also provide OCSP responses to web browsers itself through OCSP stapling. This solves this privacy risk, does not require connectivity between web browser and certificate authority, and is faster.

When connecting to your web server we use the TLS Certificate Status extension to request OCSP data be included in the server response. If your web server includes OCSP data in the response we then verify that the OCSP data is valid i.e. correctly signed by a known certificate authority. Note: we do not use the OCSP data to evaluate the validity of the certificate.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, table 15 (in English).

Requirement level: Optional

OCSP stapling

- Good: On

- Sufficient: Off

Certificate

✓ Trust chain of certificate

Verdict:
The trust chain of your website certificate is complete and signed by a trusted root certificate authority.

Technical details:

Web server IP address	Untrusted certificate chain
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if we are able to build a valid chain of trust for your website certificate.

For a valid chain of trust, your certificate must be published by a publicly trusted certificate authority, and your web server must present all necessary intermediate certificates.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B3-4 (in English).

✓ Public key of certificate

Verdict:
The digital signature of your website certificate uses secure parameters.

Technical details:

Web server IP address	Affected signature parameters
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if the (ECDSA or RSA) digital signature of your website certificate uses secure parameters.

The verification of certificates makes use of digital signatures. To guarantee the authenticity of a connection, a trustworthy algorithm for certificate verification must be used. The algorithm that is used to sign a certificate is selected by its supplier. The certificate specifies the algorithm for digital signatures that is used by its owner during the key exchange. It is possible to configure multiple certificates to support more than one algorithm.

The security of ECDSA digital signatures depends on the chosen curve. The security of RSA for encryption and digital signatures is tied to the key length of the public key.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B5-1 and table 9 for ECDSA, and guideline B3-3 and table 8 for RSA (in English).

Elliptic curves for ECDSA

- Good: `secp384r1`, `secp256r1`, `x448`, and `x25519`
- Phase out: `secp224r1`
- Insufficient: Other curves

Length of RSA-keys

- Good: At least 3072 bit
- Sufficient: 2048 – 3071 bit
- Insufficient: Less than 2048 bit

✓ Signature of certificate

Verdict:
Your website certificate is signed using a secure hash algorithm.

Technical details:

Web server IP address	Affected hash algorithm
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if the signed fingerprint of the website certificate was created with a secure hashing algorithm.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B3-2 and table 3 (in English).

Hash functions for certificate verification

- Good: SHA-512, SHA-384, SHA-256
- Insufficient: SHA-1, MD5

✓ Domain name on certificate

Verdict:

The domain name of your website matches the domain name on your website certificate.

Technical details:

Web server IP address	Unmatched domains on certificate
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	None
`13.227.219.60`	None

Test explanation:
We check if the domain name of your website matches the domain name on the certificate.

It could be useful to include more than one domain (e.g. the domain with and without www) as Subject Alternative Name on the certificate.

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, guideline B3-1 (in English).

DANE

DANE existence

Verdict:
Your website domain does *not* contain a TLSA record for DANE.

Technical details:

Web server IP address	DANE TLSA record existent
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	no
`13.227.219.60`	no

Test explanation:
We check if the name servers of your website domain contain a correctly signed TLSA record for DANE.

As DNSSEC is preconditional for DANE, this test will fail in case DNSSEC is missing on the website domain, or if there are DANE related DNSSEC issues (e.g. no proof of 'Denial of Existence').

See 'IT Security Guidelines for Transport Layer Security (TLS) v2.1' from NCSC-NL, Appendix A, under 'Certificate pinning and DANE' (in English).

Requirement level: Optional

DANE validity

Verdict:

This subtest did not run, because either a parent test that this subtest depends on gave a negative result, or not enough information was available to run this subtest.

Technical details:

Web server IP address	DANE TLSA record valid
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	`not tested`
`13.227.219.60`	`not tested`

Test explanation:
We check if the DANE fingerprint presented by your domain is valid for your web certificate.

DANE allows you to publish information about your website certificate in a special DNS record, called TLSA record. Clients, like web browsers, can check the authenticity of your certificate not only through the certificate authority but also through the TLSA record. A client can also use the TLSA record as a signal to only use HTTPS (and not HTTP). DNSSEC is preconditional for DANE. Unfortunately not much web browsers are supporting DANE validation yet.

Requirement level: Recommended (only if subtest for 'DANE existence' is passed)

Security options

Warning: *Not* all recommended application security options are set for your website (Security options). With these options you can activate browser mechanisms to protect visitors against attacks with e.g. cross-site scripting (XSS) or framing. Note that we consider HTTPS as a requirement for this test category, and that these security options are *not* relevant for domains that redirect (through 301/302 redirect).

HTTP security headers

X-Frame-Options

Verdict:
Your web server does *not* offer securely configured X-Frame-Options.

Technical details:

Web server IP address	X-Frame-Options value
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	`None`
`13.227.219.60`	`None`

Test explanation:

We check if your web server provides an HTTP header for `X-Frame-Options` that has a sufficiently secure policy. With this HTTP header you let web browsers know whether you want to allow your website to be framed or not. Prevention of framing defends visitors against attacks like clickjacking. We consider the following values to be sufficiently secure:

- `DENY` (framing not allowed); or
- `SAMEORIGIN` (only framing by your own website allowed).

Although the value `ALLOW-FROM` (only allow specific websites to frame your website) is part of the `X-Frame-Options` specification (RFC 7034), it is *not* supported by most modern web browsers. Therefore we do *not* consider this value as sufficiently secure.

Note that `Content-Security-Policy` (CSP) offers similar protection through `frame-ancestors` and besides much more for visitors with modern web browsers. However `X-Frame-Options` could still protect visitors of older web browsers that do not support CSP. Furthermore `X-Frame-Options` could offer valuable protection for all visitors when CSP is not (properly) configured for the website concerned.

Also see 'Web application guidelines from NCSC-NL', guideline U/PW.03 (in Dutch).

Requirement level: Optional

⚠ X-Content-Type-Options

Verdict:
Your web server does *not* offer X-Content-Type-Options.

Technical details:

Web server IP address	X-Content-Type value
`2600:9000:2104:1200:1a:d6d6:9e80:93a1`	None
`13.227.219.60`	None

Test explanation:
We check if your web server provides an HTTP header for `X-Content-Type`. With this HTTP header you let web browsers know that they must not do 'MIME type sniffing' and always follow the `Content-Type` as declared by your web server. The only valid value for this HTTP header is `nosniff`. When enabled, a browser will block requests for `style` and `script` when they do not have a corresponding `Content-Type` (i.e. `text/css` or a 'JavaScript MIME type' like `application/javascript`).

'MIME type sniffing' is a technique where the browser scans the content of a file to detect the format of a file regardless of the declared `Content-Type` by the web server.

This technique is vulnerable to the so-called 'MIME confusion attack' in which the attacker manipulates the content of a file in a way that it is treated by the browser as a different `Content-Type`, like an executable.

Requirement level: Recommended

⚠ Content-Security-Policy

Verdict:
Your web server does *not* offer Content-Security-Policy (CSP), or does offer CSP with certain *insecure* settings.

Technical details:

Web server IP address	CSP value
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if your web server provides an HTTP header for `Content-Security-Policy` (CSP). Furthermore we check for several (in)secure CSP settings, although we do not exhaustively test the effectiveness of your CSP configuration.

CSP guards a website against content injection attacks including cross-site scripting (XSS). By using CSP to configure an 'allowlist' with sources of approved content, you prevent browsers from loading malicious content of attackers.

We test for the settings below that are the fundament of a secure CSP configuration.

1. `default-src` should be defined in order to have a restrictive default fallback policy for source types that are used in your website for which no specific policy is defined. The value should be either `'none'` or `'self'`. Additionally, the domain itself, or a subdomain or superdomain could also be defined. Next to these values `'report-sample'` could be there in case you want code samples to be sent together with the 'violation reports'. Furthermore we allow for `https:`, although you do not need it if you adhere to the below settings.
2. `frame-src` should be used (either by definition or by relying on the fallback to `child-src` and `default-src`) and have the value `'none'`, `'self'` or a specific URL, in order to prevent other sources that are loaded into your website from framing or embedding external sources with HTML elements such as `<frame>` and `<iframe>`.
3. `frame-ancestors` should be defined and have the value `'none'`, `'self'` or a specific URL, in order to prevent other websites from framing or embedding your

website using HTML elements `<frame>`, `<iframe>`, `<object>`, `<embed>`, or `<applet>`.

4. `unsafe-inline`, `unsafe-eval` and `unsafe-hashes` should *not* be used, because these enable XSS attacks.
5. `data:` should *not* be used in `default-src`, `script-src` and `object-src`, because this enables XSS attacks.
6. `http:` should *not* be used as a scheme, because this enables resources to be downloaded over an insecure connection. Use `https:` instead.
7. `*` (wildcard for the scheme and host part of any URL) should *not* be used in any directive, because this allows for any external source.
8. `127.0.0.1` should *not* be used in any directive, because it enables content injection attacks in a compromised system.

Note that in general we recommend you to allow domains in CSP directives as specific as possible, which is tested automatically to some extent in this subtest for CSP. Moreover we do not recommend to allow all domains under a TLD (`*.nl`) or SLD (`*.gov.uk`), although we currently do not test for this. We also not recommend to allow a different domain under an SLD in `default-src` (for example allow `example1.gov.uk` for `example2.gov.uk`), although we currently do not test for this either.

Furthermore we advise you to firstly use the HTTP header for `Content-Security-Policy-Report-Only` to experiment with your CSP configuration by monitoring its effect without enforcing it.

Also see 'Web application guidelines from NCSC-NL', guideline U/PW.03 (in Dutch).

Requirement level: Recommended

⚠ Referrer-Policy existence

Verdict:
Your web server does *not* offer Referrer-Policy.

Technical details:

Web server IP address	Referrer-Policy value
2600:9000:2104:1200:1a:d6d6:9e80:93a1	None
13.227.219.60	None

Test explanation:
We check if your web server provides an HTTP header for `Referrer-Policy`. With this HTTP header you let browsers know which referrer information, that is sent in the `Referer` header, should be part of the website request. The `Referer` header contains

the address of the previous web page from which the visitor followed a link to the requested page.

The information in the `Referer` header is mostly used for analytics and logging. However there can be privacy and security risks. The information could be used e.g. for user tracking and the information could leak to third parties who eavesdrop the connection. With the HTTP header for `Referrer-Policy` you can mitigate these risks.

Currently we do *not* evaluate the effectiveness of the configured `Referrer-Policy` value. However we suggest to make an informed decision, with privacy and security risks in mind, on using one of the policy values from the first two categories below.

Recommended policy values:

1. No sensitive data to third parties
    - `no-referrer`
    - `same-origin`
2. Sensitive data to third parties only via secure connections (HTTPS)
    - `strict-origin`
    - `strict-origin-when-cross-origin`

Not recommended policy values:

1. Sensitive data to third parties possibly via insecure connections (HTTP)
    - `no-referrer-when-downgrade` (browsers' default policy)
    - `origin-when-cross-origin`
    - `origin`
    - `unsafe-url`

Requirement level: Recommended