# Website test: a3.tuyaeu.com


- *Not* reachable via modern internet address, or improvement possible (IPv6)
- Domain name *not* signed (DNSSEC)
- Connection *not* or insufficiently secured (HTTPS)
- One or more recommended security options *not* set (Security options)
- Authorised route announcement (RPKI)

## Modern address (IPv6)

Too bad! Your website is *not* reachable for visitors using a modern internet address (IPv6), or improvement is possible. Therefore your website is not part of the modern Internet yet. You should ask your hosting provider to fully enable IPv6.

### Name servers of domain

#### IPv6 addresses for name servers

#### IPv6 reachability of name servers

### Web server

#### IPv6 addresses for web server

**Verdict:**
*None* of your web servers has an IPv6 address.

**Technical details:**

| Web server | IPv6 address | IPv4 address |
|---|---|---|
| a3.tuyaeu.com | None | 18.185.182.159 |
| ... | - | 3.121.131.36 |
| ... | - | 18.195.249.137 |

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

'IPv4-mapped IPv6 addresses' (RFC 4291, beginning with ::ffff:) will fail in this subtest, as they do *not* provide IPv6 connectivity.

### IPv6 reachability of web server

**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.

**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:**
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.

**Test explanation:**
We compare the response and content received from your web server over IPv6 with that received over IPv4.

We check:

1. if HTTP (port 80) and/or HTTPS (port 443) over IPv4 are also available over IPv6;
2. if HTTP headers (such as a redirect header) over IPv4 are also received over IPv6;
3. if HTML content over IPv4 is the same over IPv6. After stripping eventual nonces we do a comparison of the received content. The content difference must not be higher than 10% to pass this subtest. The treshold makes sure that small differences (for example due to changing ads) do not lead to a fail of this subtest as well. An observed difference may be intended or due to a measurement error. Therefore, in this case, a warning is given and it is not weighed into the score.

Note that in case there are multiple IPv6 and IPv4 addresses, we pick one IPv6 address and one IPv4 address to perform the subtest. If only an IPv6 address and no IPv4 address is available, the subtest is not applicable because no comparison can be made.

## 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 |
|---|---|
| a3.tuyaeu.com | GoDaddy.com, LLC |

**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 |
|---|---|
| a3.tuyaeu.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 name redirects to another signed domain name via `CNAME`, then we also check if the signature of the CNAME domain name is valid (which is conformant with the DNSSEC standard). If the signature of the CNAME domain name is not valid, the result of this subtest will be negative.

Note that we only test the name server that responds first. If name servers of a domain name have inconsistent configurations, this can lead to varying test results. In that case, for example, the result for this subtest may be 'secure' one time and 'bogus' the next.

## 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 |
|---|---|
| 18.195.249.137 | 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 tests in the HTTPS test section only run for the first available IPv6 and IPv4 address.

### HTTPS redirect

**Verdict:**
Your web server only offers support for HTTPS and not for HTTP.

**Technical details:**

| Web server IP address | HTTPS redirect |
|---|---|
| 18.195.249.137 | not applicable |

**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 separate 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 supports HTTP compression, which could be a security risk.

**Technical details:**

| Web server IP address | HTTP compression |
|---|---|
| 18.195.249.137 | yes |

**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

---

## HSTS

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

**Technical details:**

| Web server IP address | HSTS policy |
|---|---|
| 18.195.249.137 | 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.

The test does **not** check whether `preload` is used and whether the domain is included in the HSTS Preload List.

---

**Deployment recommendation**

HSTS requires your website to fully work over HTTPS. This also includes having a valid certificate from a publicly trusted certificate authority. So when your website does not properly support HTTPS, note that it will *not be reachable* by browsers that have contacted your website before. A HSTS policy change to revert effects will not have immediate effect for these browsers since they have cached your previous HSTS policy.

Therefore we advise you to follow the implementation steps below:

1. Make sure that the website on your domain fully works over HTTPS now and in the future (e.g. by having a solid certificate rollover procedure);
2. Increase the HSTS cache validity period in the stages below. During each stage, carefully check for reachability issues and broken pages. Fix any issues that come up and then wait the full max-age of the stage before moving to the next stage.
3. Start with 5 minutes (`max-age=300`);
4. Increase it to 1 week (`max-age=604800`);
5. Increase it to 1 month (`max-age=2592000`);
6. Increase it to 1 year (`max-age=31536000`) or more.

7. Repeat step 1 and 2 for every single subdomain that you want to secure with HSTS. Only use `includeSubDomains` if you are fully sure that **all** the (nested) subdomains of your domain properly support HTTPS now and in the future.

8. Use `preload` **only** if you are very sure that you want your domain to be included in the HSTS Preload List. HSTS preloading is mostly interesting for highly sensitive websites. Administrators who want to enable HSTS preloading for a particular domain should do so with extreme caution. It can affect the reachability of the domain and of any subdomains, and is not easily reversed.

---

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 | TLS version | Status |
|---|---|---|
| 18.195.249.137 | TLS 1.3 | good |
| ... | TLS 1.2 | sufficient |

**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 |
|---|---|
| 18.195.249.137 | 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:**
This subtest is not applicable as your web server supports 'Good' ciphers only.

**Technical details:**

| Web server IP address | First found affected cipher pair |
|---|---|
| 18.195.249.137 | 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 subtest 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 |
|---|---|
| 18.195.249.137 | 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

---

Note: the above names are based on the IANA naming conventions. Sometimes alternative names are used to refer to the same curves, like `prime256v1` (ANSI) and `NIST P-256` for `secp256r1`.

---

### 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 |
|---|---|
| 18.195.249.137 | 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

---

### Secure renegotiation

---

### Client-initiated renegotiation

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

**Technical details:**

| Web server IP address | Client-initiated renegotiation |
|---|---|
| 18.195.249.137 | 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 |
|---|---|
| 18.195.249.137 | 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 |
|---|---|
| 18.195.249.137 | no |

**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 *not* complete and/or *not* signed by a trusted root certificate authority.

**Technical details:**

| Web server IP address | Untrusted certificate chain |
|---|---|
| 18.195.249.137 | *.tuyacn.com |

**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 |
|---|---|
| 18.195.249.137 | 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 |
|---|---|
| 18.195.249.137 | 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 does *not* match the domain name on your website certificate.

**Technical details:**

| Web server IP address | Unmatched domains on certificate |
|---|---|
| 18.195.249.137 | *.tuyacn.com |

**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 |
|---|---|
| 18.195.249.137 | 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 |
|---|---|
| 18.195.249.137 | 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 security options, , i.e. security headers and security.txt, are set for your website (Security options). With security headers you can activate browser mechanisms to protect visitors against attacks involving, for example, cross-site scripting (XSS) or framing. Through a security.txt file you can provide researchers who have found a vulnerability on your systems, with your contact infomation and your Coordinated Vulnerability Disclosure policy. Note that HTTPS is a prerequisite for all tested security options. Security headers (except for Referrer-Policy) are *not* relevant for domains that redirect (through a 301/302 HTTP Status Code).

### 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 |
|---|---|
| 18.195.249.137 | 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-Options value |
|---|---|
| 18.195.249.137 | None |

**Test explanation:**
We check if your web server provides an HTTP header for `X-Content-Type-Options`. 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 | Findings |
|---|---|
| 18.195.249.137 | No CSP found. |

**Test explanation:**
We check if your web server provides an HTTP header for `Content-Security-Policy` (CSP). Furthermore we check for several 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 rules below that are based on good practices for a secure CSP configuration.

1. The `default-src` directive should be defined and its value should be be either `'none'`, or `'self'` and/or one or more URLs of the domain itself (including its subdomain or superdomain). This is to have a restrictive default fallback policy for source types that are used in your website for which no specific policy is defined. Next to the mentioned values `'report-sample'` could be there in case you want code samples to be sent together with the 'violation reports'.
2. The `base-uri` directive should be defined and its value should be either `'none'`, or `'self'` and/or one or more URLs of the domain itself (including its subdomain or superdomain). This restricts the `baseURI` in order to block the injection of a manipulated `<base>` element. This limits the ability of attackers to manipulate the locations of sources (e.g. scripts) that are loaded from relative URLs.
3. The `frame-src` directive should be used (either by definition or by relying on the fallback to sequentially `child-src` and `default-src`) and its value should be either `'none'`, or `'self'` and/or one or more specific URLs. This prevents content from unauthorized locations to be framed or embedded in your website (with HTML elements such as `<frame>` and `<iframe>`).

4. The `frame-ancestors` directive should be defined and its value should be either `'none'`, or `'self'` and/or one or more specific URLs. This prevents unauthorized websites from framing or embedding your website (with HTML elements `<frame>`, `<iframe>`, `<object>`, `<embed>`, or `<applet>`).
5. The `form-action` directive should be defined and its value should be either `'none'`, or `'self'` and/or one or more specific URLs. This restricts the URLs which can be used as the target of form submissions.
6. The source values `unsafe-eval`, `unsafe-inline` and `unsafe-hashes` should *not* be used, because these enable XSS attacks.
7. The `data:` scheme should *not* be used in the `default-src`, `script-src` and `object-src` directives, because this enables XSS attacks.
8. A domain with the `http://` scheme or a domain without a scheme should *not* be used, because this enables sources to be downloaded over an insecure connection. Use the `https://` scheme instead.
9. A wildcard (i.e. `*`) for the full host part of a URL should *not* be used in any directive, because this allows for any external source. Furthermore do *not* use just `https:` as this is equivalent to `https://*`. Always specify the main domain as well (e.g. `https://scripts.example.nl` or `https://*.example.nl`).
10. `127.0.0.1` should *not* be used in any directive, because it enables content injection attacks in a compromised system.

Additional recommendations:

1. Only use domains in CSP directives as specific as possible, which is tested automatically to some extent in this subtest for CSP.
    - Do *not* allow all domains under a TLD (`*.nl`) or SLD (`*.gov.uk`), although we currently do *not* test for this.
    - Do *not* allow a different main domain under a SLD in `default-src` (for example `example2.gov.nl` for `example1.gov.nl`), although we currently do *not* test for this either.
2. Ideally do *not* use inline scripts or styles. For cases where you cannot avoid using inline scripts or styles, do *not* use `unsafe-inline` (as mentioned above) but preferably use CSP hashes or otherwise CSP nonces.
3. Use the HTTP header as a mechanism for serving your CSP policy. Do *not* use the HTML `<meta>` element to serve your CSP policy. The latter is less secure and does not support all CSP directives (e.g. the required `frame-ancestors`). Therfore we do *not* check for CSP that is offered via the HTML `<meta>` element.
4. Use the HTTP header for `Content-Security-Policy-Report-Only` to experiment with your CSP configuration by monitoring its effect without enforcing it.

5. Note that an empty source list (i.e. a CSP directive without a value) is equivalent to a source list containing `none`. Besides, note that if other sources are listed in a source list next to `none`, then the attribute `none` is ignored.

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

*Requirement level: Recommended*

---

### Referrer-Policy

**Verdict:**
Your web server either does *not* offer a Referrer-Policy and thus relies on the default browser policy, or your web server offers a Referrer-Policy with a policy value that should be used with caution because of its potential impact on security and privacy.

**Technical details:**

| Web server IP address | Findings |
|---|---|
| 18.195.249.137 | No Referrer-Policy found. |

**Test explanation:**
We check if your web server provides an HTTP header for `Referrer-Policy` with a sufficiently secure and privacy-protecting policy value. With this policy your webserver instructs browsers if, what and when referrer data should be sent to the website the user is navigating to from your website. This referrer data is sent in the `Referer` header by the browser to the website the user is navigating to. This navigation does not necessarily have to be cross-domain.

The data in the `Referer` header is usually used for analytics and logging. However there can be privacy and security risks. The data could be used e.g. for user tracking and the data could leak to third parties who eavesdrop the connection. The HTTP header for `Referrer-Policy` allows you to mitigate these risks by controlling and even minimizing the sending of referrer data.

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.

**1. Good**

- `no-referrer`
- `same-origin`

With these policy values no sensitive referrer data is sent to third parties.

**2. Warning**

- `strict-origin`

- `strict-origin-when-cross-origin` (browser's default value; see also under additional note 2)

With these policy values basic referrer data, which may be sensitive, is sent to third parties only via secure connections (HTTPS). Therefore these values should only to be used where necessary and permitted by law.

**3. Bad**

- `no-referrer-when-downgrade`
- `origin-when-cross-origin`
- `origin`
- `unsafe-url`

With these policy values any or basic referrer data, which may be sensitive, is sent to third parties possibly via insecure connections (HTTP). Therefore these values must not be used.

**Additional notes:**

1. To prevent leaking of sensitive data through URLs, please make sure URLs of your website do not contain personal or otherwise sensitive data (like personal names or passwords).
2. `strict-origin-when-cross-origin` is the default policy value when no `Referrer-Policy` is published as prescribed in the current "Editor's Draft" of the `Referrer-Policy` standard and this default is used by all latest major browsers. Note that some users may still have a different default value configured in their browser.
3. Use the HTTP `Referrer-Policy` header as a mechanism for serving your referrer policy. We do *not* recommend to use the HTML `<meta>` element or the HTML `referrerpolicy` content attribute to publish your referrer policy, especially because the sections that describe these methods in the standard are marked as "not normative". Therefore we do not check for a referrer policy that is offered via the latter two methods.
4. Note that a web server may send multiple `Referrer-Policy` headers, and a `Referrer-Policy` header may contain multiple values. Unknown values will be ignored by the browser and only the last known value will be used. This makes it possible to use future standardised policy values that are not supported by all browsers yet. However, currently all latest major browsers support the above mentioned policy values under "Good" and "Warning".

*Requirement level: Recommended*

---

**Other security options**

⚠ security.txt

## Route authorisation (RPKI)

Well done! All IP addresses of your web server and associated name servers have a route announcement that is matched by the published route authorisation (RPKI). As a result, visitors with enabled route validation are better protected against various unintentional or malicious route configuration errors, that can lead to the unreachability of your servers or the interception of Internet traffic to your servers.

### Name servers of domain

#### Route Origin Authorisation existence

**Verdict:**
For all IP addresses of your name servers an RPKI Route Origin Authorisation (ROA) is published.

**Technical details:**

| Name server | IP address | RPKI Route Origin Authorization |
|---|---|---|
| ns-1043.awsdns-02.org. | 205.251.196.19 | yes |
| ... | 2600:9000:5304:1300::1 | yes |
| ns-1566.awsdns-03.co.uk. | 205.251.198.30 | yes |
| ... | 2600:9000:5306:1e00::1 | yes |
| ns-276.awsdns-34.com. | 205.251.193.20 | yes |
| ... | 2600:9000:5301:1400::1 | yes |
| ns-834.awsdns-40.net. | 205.251.195.66 | yes |
| ... | 2600:9000:5303:4200::1 | yes |

**Test explanation:**
We check if an RPKI Route Origin Authorization (ROA) has been published for all IP addresses of the name servers of your domain.

Your hoster (or its network provider) announces through the Border Gateway Protocol (BGP) for which of its IP address blocks it accepts incoming Internet traffic. Other network providers use these route announcements to determine via which route to send traffic for your server's IP addresses.

However, a route announcement can be faked. In fact, another network provider may be able to connect the IP address block of your IP address to its network and thus potentially receive Internet traffic that is actually intended for your network provider. The cause may be accidental or malicious. In either case, this can result in your server becoming unreachable or in Internet traffic to your server being intercepted.

Resource Public Key Infrastructure (RPKI) significantly improves protection against this. With RPKI, the rightful holder of a block of IP addresses can publish a digitally signed statement with route authorization (Route Origin Authorisation; ROA for short). Another network provider that wants to send Internet traffic to a particular IP address, can use the corresponding statement to filter out `Invalid` routes. In this way, the network provider prevents Internet traffic from its network from being sent to unauthorized provider networks.

*Requirement level: Recommended*

---

### Route announcement validity

**Verdict:**
All IP addresses of your name servers have a `Valid` validation state. The route announcement of these IP addresses is matched by the published RPKI Route Origin Authorisation (ROA).

**Technical details:**

| Name server | BGP Route Prefix | BGP Route Origin ASN | RPKI Origin Validation state |
|---|---|---|---|
| ns-1043.awsdns-02.org. | 205.251.196.0/24 | AS16509 | valid |
| ... | 2600:9000:5304::/48 | AS16509 | valid |
| ns-1566.awsdns-03.co.uk. | 205.251.198.0/24 | AS16509 | valid |
| ... | 2600:9000:5306::/48 | AS16509 | valid |
| ns-276.awsdns-34.com. | 205.251.193.0/24 | AS16509 | valid |
| ... | 2600:9000:5301::/48 | AS16509 | valid |
| ns-834.awsdns-40.net. | 205.251.195.0/24 | AS16509 | valid |
| ... | 2600:9000:5303::/48 | AS16509 | valid |

**Test explanation:**
We check if the validation status for all IP addresses of the name servers of your domain is `Valid`. If there are multiple overlapping route announcements for the IP address, we test that they are all `Valid`.

Your hoster (or its network provider) announces via the Border Gateway Protocol (BGP) for which of its IP address blocks it accepts incoming Internet traffic. It does this by associating (parts of) its IP address blocks (Route Prefix) with the unique number of its provider network (Route Origin ASN), and then distributing this routing information. Other network providers use these route announcements to determine via which route to send traffic for the IP addresses.

Additionally, for each route announcement, your hoster (or its network provider) should publish a digitally signed statement with route authorisation (Route Origin

Authorisation; abbreviated: ROA) statement using Resource Public Key Infrastructure (RPKI).

Another network provider that is asked to send Internet traffic to your server's IP address can cryptographically verify the associated statement. The verified content (Validated ROA Payload; abbreviated: VRP) includes which provider networks (VRP ASN) are authorised to receive Internet traffic for each recorded IP address block (VRP Prefix).

The network provider can then use this content to filter BGP routes. For example, he can reject any `Invalid` routes, to prevent Internet traffic from its network is sent to unauthorised provider networks.

Checking route announcements against route authorisations (RPKI Origin Validation; abbreviated: ROV) has the following three possible outcomes.

- `Valid`: The route announcement of the considered IP address is matched by at least one published route authorisation. A route authorisation is also matched for more specific route announcements (i.e., for a part of the IP address block contained in the route authorisation), unless they are more specific than the limit specified in the route authorisation (via the `maxLength` attribute).

- `Invalid`: The route announcement of the considered IP address is covered by a route authorisation, but it does *not* match. There are two possible causes:

- the IP address block (Route Prefix) is announced from a provider network (Route Origin ASN) that is *not* authorised in the route authorisation (result in the table below: "invalid (as)");

- the IP address block (Route Prefix) that is announced is *smaller than is allowed* in the route authorisation, i.e. exceeding `maxLength` value (result in table below: "invalid (length)").

Note: The status `Invalid` indicates an unintentional or malicious route configuration error, that can be caused by your hoster or its network provider (internal error) or by a third party (external error). In either case, it can lead to the unreachability of your server or the interception of Internet traffic to your server. Contact your hoster as soon as possible. In the case of an internal error, your hoster should ask his network provider that owns your server's IP addresses to fix the route configuration error.

- `NotFound`: No statement with route-authorisation has been found that covers the route announcement of the considered IP address. This makes the routing of Internet traffic to your server less secure. Please contact your hoster about this. He can ask his network provider that owns your server's IP addresses to publish route authorisations.

*Requirement level: Recommended*

---

## Web server

---

### Route Origin Authorisation existence

**Verdict:**
For all IP addresses of your web server an RPKI Route Origin Authorisation (ROA) is published.

**Technical details:**

| Webserver | IP address | RPKI Route Origin Authorization |
|---|---|---|
| a3.tuyaeu.com | 18.195.249.137 | yes |

**Test explanation:**
We check if an RPKI Route Origin Authorization (ROA) has been published for all IP addresses of your web server.

Your hoster (or its network provider) announces through the Border Gateway Protocol (BGP) for which of its IP address blocks it accepts incoming Internet traffic. Other network providers use these route announcements to determine via which route to send traffic for your server's IP addresses.

However, a route announcement can be faked. In fact, another network provider may be able to connect the IP address block of your IP address to its network and thus potentially receive Internet traffic that is actually intended for your network provider. The cause may be accidental or malicious. In either case, this can result in your server becoming unreachable or in Internet traffic to your server being intercepted.

Resource Public Key Infrastructure (RPKI) significantly improves protection against this. With RPKI, the rightful holder of a block of IP addresses can publish a digitally signed statement with route authorization (Route Origin Authorisation; ROA for short). Another network provider that wants to send Internet traffic to a particular IP address, can use the corresponding statement to filter out `Invalid` routes. In this way, the network provider prevents Internet traffic from its network from being sent to unauthorized provider networks.

*Requirement level: Recommended*

---

### Route announcement validity

**Verdict:**
All IP addresses of your web server have a `Valid` validation state. The route announcement of these IP addresses is matched by the published RPKI Route Origin Authorisation (ROA).

**Technical details:**

| Web server | BGP Route Prefix | BGP Route Origin ASN | RPKI Origin Validation state |
|---|---|---|---|
| a3.tuyaeu.com | 18.194.0.0/15 | AS16509 | valid |

**Test explanation:**
We check if the validation status for all IP addresses of your web server is `Valid`. If there are multiple overlapping route announcements for the IP address, we test that they are all `Valid`.

Your hoster (or its network provider) announces via the Border Gateway Protocol (BGP) for which of its IP address blocks it accepts incoming Internet traffic. It does this by associating (parts of) its IP address blocks (Route Prefix) with the unique number of its provider network (Route Origin ASN), and then distributing this routing information. Other network providers use these route announcements to determine via which route to send traffic for the IP addresses.

Additionally, for each route announcement, your hoster (or its network provider) should publish a digitally signed statement with route authorisation (Route Origin Authorisation; abbreviated: ROA) statement using Resource Public Key Infrastructure (RPKI).

Another network provider that is asked to send Internet traffic to your server's IP address can cryptographically verify the associated statement. The verified content (Validated ROA Payload; abbreviated: VRP) includes which provider networks (VRP ASN) are authorised to receive Internet traffic for each recorded IP address block (VRP Prefix).

The network provider can then use this content to filter BGP routes. For example, he can reject any `Invalid` routes, to prevent Internet traffic from its network is sent to unauthorised provider networks.

Checking route announcements against route authorisations (RPKI Origin Validation; abbreviated: ROV) has the following three possible outcomes.

- `Valid`: The route announcement of the considered IP address is matched by at least one published route authorisation. A route authorisation is also matched for more specific route announcements (i.e., for a part of the IP address block contained in the route authorisation), unless they are more specific than the limit specified in the route authorisation (via the `maxLength` attribute).

- `Invalid`: The route announcement of the considered IP address is covered by a route authorisation, but it does *not* match. There are two possible causes:

- the IP address block (Route Prefix) is announced from a provider network (Route Origin ASN) that is *not* authorised in the route authorisation (result in the table below: "invalid (as)");

- the IP address block (Route Prefix) that is announced is *smaller than is allowed* in the route authorisation, i.e. exceeding `maxLength` value (result in table below: "invalid (length)").

- `NotFound`: No statement with route-authorisation has been found that covers the route announcement of the considered IP address. This makes the routing of Internet traffic to your server less secure. Please contact your hoster about this. He can ask his network provider that owns your server's IP addresses to publish route authorisations.

**What to do if the test result shows the status Invalid?**

The status `Invalid` indicates an unintentional or malicious route configuration error, that can be caused by your hoster or its network provider (internal error) or by a third party (external error). In either case, it can lead to the unreachability of your server or the interception of Internet traffic to your server. Contact your hoster as soon as possible. In the case of an internal error, your hoster should ask his network provider that owns your server's IP addresses to fix the route configuration error.

*Requirement level: Recommended*

---